TÁCTICAS ARQUITECTÓNICAS DE SEGURIDAD PARA *E-VOTING*

DANIEL ESTEBAN BURBANO
FRANCISCO JAVIER ZAMBRANO SANTACRUZ

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2020

TÁCTICAS ARQUITECTÓNICAS DE SEGURIDAD PARA *E-VOTING*

DANIEL ESTEBAN BURBANO
FRANCISCO JAVIER ZAMBRANO SANTACRUZ

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero de Sistemas

ASESOR

GIOVANNI HERNÁNDEZ PANTOJA, Mg.

CO-ASESOR

SANDRA MARLENI VALLEJO CHAMORRO, Mg.

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2020

NOTA DE RESPONSABILIDAD

“Las ideas y conclusiones aportadas en la tesis de grado son responsabilidad exclusiva del autor”.

Artículo 1 del acuerdo No.324 de octubre 11 de 1966, emanado por el Honorable Consejo Directivo de la universidad de Nariño.

“La Universidad de Nariño no se hace responsable de las opiniones y resultados obtenidas en el presente trabajo y para su publicación priman las normas sobre el derecho de autor”.

Artículo 13. No.005 de 2010 emanado del Honorable Consejo Académico.

NOTA DE ACEPTACIÓN:

_____

_____

_____

_____

_____

_____

Firma Presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

San Juan de Pasto, febrero del 2020

AGRADECIMIENTOS

Agradezco a Dios por guiarme a lo largo de mi vida y darme fortaleza y sabiduría para que fuera posible cumplir esta meta.

A mi familia por su apoyo incondicional en todo momento y especialmente a mis padres quienes con gran esfuerzo han permitido que esto sea posible.

Agradecer a mi asesor, Giovanni Albeiro Hernández Pantoja, por la orientación y el gran aporte con su experiencia a este proyecto, gracias por compartir sus conocimientos, apoyo, consejos y su amistad.

Gracias a mi compañero de tesis Francisco Javier Zambrano Santacruz, por su compromiso y responsabilidad, pero principalmente por su amistad, por la gran calidad de ser humano que es.

Finalmente agradecer a la universidad de Nariño, al grupo de investigación Galeras.Net y al aula de informática por todo el apoyo.

Daniel Burbano.

Agradezco a Dios por todas las bendiciones que me ha dado y guiarme por el camino correcto.

Agradezco a mi familia por su amor, trabajo y sacrificio durante todos estos años, gracias a ustedes he logrado llegar hasta aquí y convertirme en lo que soy.

Mi profundo agradecimiento a nuestro asesor Giovanni Hernández, por brindarnos su tiempo, paciencia, experiencia y conocimientos para cumplir esta meta, pero, ante todo, por la gran de persona que es.

También quiero agradecer a mis compañeros, amigos con los que compartí y aprendí valores tan importantes como la amistad y el trabajo en equipo, y en general, a todas las personas que me han prestado su ayuda durante el proceso de investigación de este trabajo.

De manera especial a Daniel Burbano, por todo su apoyo, responsabilidad, por compartir sus conocimientos y especialmente, por brindarme su incondicional amistad y convertirse en un hermano para mí.

Por último, agradezco a la Universidad de Nariño, el grupo de investigación Galeras.net, el aula de informática y a todas las autoridades, por permitirme concluir una etapa en mi vida.

Francisco Zambrano.

RESUMEN

Los procesos en la democracia colombiana, no brindan las garantías necesarias a los electores y elegidos, permitiendo que se generen alteraciones o errores en los resultados. En los últimos años, el voto electrónico (*e-voting*) se ha presentado como una alternativa al fraude electoral; sin embargo, naciones con democracias consolidadas, argumentan que no lo utilizan principalmente por problemas en la seguridad. La motivación por esta investigación nace del interés por indagar sobre tácticas arquitectónicas de seguridad que se pueden implementar en el desarrollo de un producto software para *e-voting*. El objetivo de esta investigación fue identificar tácticas arquitectónicas de seguridad utilizadas en la construcción de software para *e-voting* e implementarlas en un producto software. La metodología utilizada para identificar las tácticas fue la revisión sistemática de literatura recurriendo a las fases: formular preguntas, planear y ejecutar la búsqueda, seleccionar estudios relevantes, evaluar la calidad de los estudios, extraer los datos y sintetizar los resultados. A partir de las tácticas identificadas, se procedió a construir una solución software que implemente las tácticas pertinentes para *e-voting*. Los principales resultados indican que las tácticas arquitectónicas de seguridad se orientan principalmente a detectar, resistir, reaccionar y recuperarse, a ataques; junto con auditar, autenticar y establecer sesiones seguras. Los campos de mayor aplicación para estas tácticas son la industria, la academia y el gobierno. Además, se logra sistematizar las tácticas, identificando las principales prácticas, efectos y los recursos utilizados para implementarlas. A partir de las tácticas identificadas, se construye la herramienta Kybernan utilizando Scrum como método para la gestión del proyecto. Esta investigación permite concluir que las principales tácticas arquitectónicas de seguridad para *e-voting* se centran en detectar, resistir, reaccionar y recuperarse, a los ataques. Kybernan es una herramienta que soporta un proceso electoral democrático que implementa las tácticas arquitectónicas de seguridad permitentes.

ABSTRACT

The process in Colombian democracy does not provide the necessary guarantees to voters and the elected, allowing alterations or errors to be generated in the results. In recent years, electronic voting (*e-voting*) has been presented as an alternative to electoral fraud; however, nations with consolidated democracies argue that they do not use it mainly due to security problems. The motivation for this research stems from the interest in investigating architectural security tactics that can be implemented in the development of a software product for *e-voting*. The objective of this research was to identify security architectural tactics used in the construction of *e-voting* software and implement them in a software product. The methodology used to identify the tactics was a systematic review of the literature, resorting to the phases: formulating questions, planning and executing the search, selecting relevant studies, evaluating the quality of the studies, extracting the data and synthesizing the results. From the identified tactics, a software solution was built that implements the pertinent tactics for *e-voting*. The main results indicate that the architectural security tactics are mainly oriented to detect, resist, react and recover from attacks; along with auditing, authenticating, and establishing *Secure* sessions. The fields of greatest application for these tactics are industry, academia, and government. In addition, it is possible to systematize the tactics, identifying the main practices, effects and the resources used to implement them. From the identified tactics, the tool Kybernan is built using Scrum as a method for project management. This research allows us to conclude that the main architectural security tactics for *e-voting* focus on detecting, resisting, reacting and recovering from attacks. Kybernan is a tool that supports a democratic electoral process that implements the permitting security architectural tactics.

CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

GLOSARIO

- ***E-voting:*** sistema de votación electrónico.
- ***Cracker:*** persona con grandes conocimientos de informática que se dedica a acceder ilegalmente a sistemas informáticos ajenos y a manipularlos.
- ***Sprint:*** intervalo prefijado de tiempo (no inferior a una semana ni superior a un mes) durante el cual se crea un incremento de producto "Hecho o Terminado" utilizable, potencialmente entregable
- ***Sprint backlog:*** planificación táctica del trabajo a realizar en la iteración actual. Esta lista permite ver las tareas donde el equipo está teniendo problemas y no avanza, con lo que le permite tomar decisiones al respecto.
- ***Product backlog:*** Es una lista de Historias de Usuario, ordenadas según el valor de negocio que establece el Dueño del Producto, y que trata de cubrir todas las funcionalidades necesarias.
- ***Review:*** reunión de colaboración donde se busca "*feedback*" de todos los presentes fundamentalmente para crear transparencia sobre el incremento de producto y permitir la adaptación del "*Product Backlog*" el "*Release Plan*" si fuera el caso.

INTRODUCCIÓN

Partiendo de las necesidades de la sociedad, una exigencia que ésta demanda, es que los procesos electorales democráticos se realicen de manera transparente e inequívoca. Actualmente, este tipo de procesos en la democracia colombiana, no brindan al elector una garantía de veracidad, dejándolo con una sensación de incertidumbre.

La seguridad es una característica que garantiza la calidad de un proceso electoral, puesto que permiten la protección de la información y los datos de manera que personas no autorizadas puedan leerlos o modificarlos. Del mismo modo estas características inspiran confianza a los electores, promoviendo así, la participación de la sociedad a la democracia.

Por otra parte, la tecnología ha venido evolucionando y transformándose en un recurso importante para el desarrollo de todo proceso, además, la importancia que ha tenido el software es muy contundente hoy en día, debido a que la sociedad requiere de herramientas que sirvan de apoyo a la realización de tareas complejas. Los sistemas *e-voting*, son productos software que apoyan a los procesos electorales, permitiendo a los electores vivir una experiencia diferente a la que están acostumbrados con los procesos tradicionales de elección, además que producen en el elector una sensación de confianza. Existen tácticas de arquitectura de software que permiten enfocar el producto hacia un fin, por ejemplo, hacia la seguridad. Por esta razón los sistemas *e-voting* deben contar con esta característica para enfrentar los desafíos que conlleva esta actividad.

Por lo anteriormente descrito, cobra relevancia identificar tácticas de arquitectura de seguridad para sistemas *e-voting*, con el fin de diseñar y conformar un producto software que las implemente, obteniendo como resultado un sistema seguro, transparente y auditable por el elector. Para ello se realizó un estudio preliminar que permitió determinar las tácticas de arquitectura de seguridad de software existentes, a continuación, se estableció cuáles de estas tácticas de seguridad son pertinentes para sistemas *e-voting*. Posteriormente se desarrolló un producto software de *e-voting* que implementó las tácticas de seguridad, en la cual se utilizó elementos de *Scrum* como método para la gestión del proceso de software.

Problema de investigación

Objeto o tema de investigación

TÁCTICAS ARQUITECTÓNICAS DE SEGURIDAD PARA *E-VOTING*.

Línea de investigación

Línea de investigación de software y manejo de información.

Alcance y delimitación

Se construyó un sistema que implementa tácticas arquitectónicas de seguridad para *e-voting*, con el fin de garantizar transparencia en los procesos de elección democráticos.

Para lograr la construcción del sistema de *e-voting*, se partió de una revisión sistemática de literatura relacionado con las tácticas arquitectónicas de seguridad utilizadas en la construcción de productos software. Como resultado, se obtuvo una matriz que clasifica las tácticas arquitectónicas de seguridad que se han propuesto para garantizar transparencia en los procesos de elección. Además, se estableció cuáles de las tácticas encontradas, dan respuesta a los principales problemas que se presentan en relación con el atributo de calidad de seguridad, en un producto software. Con base en las tácticas identificadas, se construyó un producto software para *e-voting*, que las implemente.

Modalidad

El proyecto se enmarcó dentro de la modalidad de trabajo de investigación.

Descripción del problema

Planteamiento del problema

En la actualidad las instituciones y organizaciones colombianas no cuentan con una herramienta que permita hacer un proceso de elecciones democráticas mediante un sistema seguro, transparente y auditable por el elector （Revista Semana，2018）. Lo que implica que se continúe usando los procesos tradicionales de votación, es decir votaciones democráticas manuales usando las boletas tradicionales de papel, las cuales deben ser marcadas y depositadas en una urna y posteriormente contabilizadas por los jurados de votación.

El proceso electoral convencional no brinda las garantías necesarias y siempre ha existido incertidumbre en el elector en cuestiones de alteración, privacidad y errores de conteo.

Existen organizaciones que tratan de garantizar y velar la transparencia de dichos procesos electorales (tradicionales) en Colombia, pero la historia ha mostrado que los resultados de las elecciones no son los más confiables independientemente de la organización encargada de velar y dar garantías en las elecciones democráticas y esto radica en que el problema está enfocado en el sistema y no en las organizaciones encargadas de dar las garantías, puesto que el sistema tradicional es difícilmente auditable, lo cual genera que pueda presentarse alteración en los resultados electorales.

Otro de los inconvenientes que presenta el sistema actual de votación es que el reporte de los resultados suele requerir mucho tiempo y el escrutinio de los votos lo hacen los jurados de votación, esto genera aún más desconfianza para el sistema dado que el ser humano no está exento de cometer errores y las actividades que se desarrollan son completamente manuales. Además, actualmente no existe un proceso de verificación al momento en que el votante ingresa a la urna, dejando una posibilidad para suplantación de identidad.

Cabe resaltar que estos inconvenientes se concluyen en una deserción de votantes masiva, por ejemplo （ELTIEMPO，2018），en las elecciones presidenciales celebradas el 28 de mayo del 2018 el 46,62% de la población colombiana no ejerció su derecho al voto, según, el 100 por ciento de los votos escrutados por la Registraduría Nacional, solo ejercieron su derecho al voto 19'636.714 personas de las 36’783.940 habilitadas. Además, en el año 2016 la abstención al voto del plebiscito por el proceso de paz fue de un 62%, una cifra bastante alta que no se presentaba en los últimos 24 años en Colombia （BBC Mundo，2016）.

Por otra parte, la tasa de votos nulos en Colombia en el año 2014 para las elecciones del senado fue del 10.38% y del 12.23% para la cámara de representantes （Registraduría Nacional del Estado civil，2014），estos índices se deben a la complejidad a la hora de votar en los tarjetones tradicionales de elección porque hay saturación de información que confunde al elector.

Estas cifras también se producen debido a cómo se maneja el proceso electoral actualmente, puesto que no usan herramientas software que garanticen la integridad, seguridad y transparencia de los procesos electorales generando desconfianza en la comunidad.

Otra razón importante es la que tiene que ver con los adultos mayores y discapacitados, la poblacion de adultos mayores que superan los 60 años en Colombia comprende el 11% de la poblacion （Ministerio de Salud, COLCIENCIAS，2015）. Este grupo de personas se abstiene de votar, debido a la dificultad que les representa este proceso de elección y como se puede ver siempre las personas con diferentes discapacidades se han mirado afectadas por el contexto social ya sea por discriminación o por falta de conocimiento en el tema de elección.

Teniendo en cuenta que todos los ciudadanos tienen el derecho a la democracia y a elegir sus representantes, se debe contar con un sistema que resuelva estas problemáticas e incentive a la comunidad a ejercer este derecho.

En los últimos años, el voto electrónico (*e-voting*) ha tomado fuerza en diferentes países como Brasil, Bélgica, India, Venezuela y Argentina （Estudiantes de Comunicación Social y Periodismo – Universidad Los Libertadores，2018），sin embargo, en Colombia han fracasado los intentos de implementación debido a que no existen las garantías necesarias en los tópicos de seguridad en sistemas *e-voting*.

Mientras que en muchos países se quiere implementar el voto electrónico, en naciones con democracias consolidadas como: Alemania, Finlandia, Holanda, Irlanda y el Reino Unido; argumentan que no lo hacen por problemas de seguridad （Estudiantes de Comunicación Social y Periodismo – Universidad Los Libertadores，2018）; es decir sostienen que el porcentaje de fraude puede ser muy alto, cabe destacar que Holanda tuvo un sistema de votación electrónica, pero decidió volver al sistema de votación tradicional. El problema de base en Holanda no es utilizar tecnología en entornos electorales, sino la falta de mantenimiento y actualización de los sistemas que utiliza para las elecciones （Ramos，2017）. En una auditoría de seguridad, las autoridades vieron que tanto el software como los ordenadores que se están utilizando para enviar esa información utilizan un sistema operativo Windows XP y un software que no se han actualizado desde hace casi 10 años. Tampoco implementaba medidas criptográficas para proteger los datos que se transmitían, como firmas electrónicas o cifradas. Por lo tanto, sin mantenimiento de seguridad desde hace años y sin medidas de protección de los datos transmitidos, existe un riesgo evidente de seguridad.

La dificultad principal que se encuentra para diseñar e implementar un producto software para *e-voting* se centra en el atributo de calidad de seguridad. El error más grave por parte de algunos de los desarrolladores e investigadores es considerar

que el nivel secreto de una votación electrónica es similar al de una entidad financiera, cuando en ésta, la operación puede ser conocida por terceros autorizados y en cambio en el voto electrónico el anonimato es parte esencial del mismo （Alonso, 2014）. Por lo tanto, este estudio pretende mostrar cómo se desarrolló un producto software *e-voting*, que permitió implementar y validar las tácticas arquitectónicas de seguridad pertinentes.

Formulación del problema

¿Cuáles son las tácticas arquitectónicas de seguridad que aportan a la construcción de un producto software para *e-voting*?

Sistematización del problema.

- ¿Cuáles son las tácticas arquitectónicas de seguridad utilizadas en la construcción de productos software?
- ¿Qué tácticas arquitectónicas de seguridad son pertinentes para un producto software de *e-voting*?
- ¿Cómo construir un producto software para *e-voting* que implemente las tácticas arquitectónicas de seguridad pertinentes para *e-voting*?

Objetivo general

Identificar las tácticas arquitectónicas de seguridad que aportan a la construcción de un producto software para *e-voting*.

Objetivos específicos

- Establecer tácticas arquitectónicas de seguridad utilizadas en la construcción de productos software a través de la revisión sistemática de literatura.
- Identificar tácticas arquitectónicas de seguridad pertinentes para un producto software de *e-voting*
- Construir un producto software para *e-voting* que implemente las tácticas arquitectónicas de seguridad pertinentes para *e-voting*.

Justificación

Este trabajo es importante, porque permitió identificar las tácticas arquitectónicas de seguridad que aportan a la construcción de un producto software para *e-voting*, para un proceso electoral totalmente auditable, transparente y seguro.

Por otra parte, con la construcción de un sistema *e-voting*, que implemente las tácticas arquitectónicas de seguridad pertinentes a estos sistemas, se eliminó el margen del error humano en las tareas que se desarrollan actualmente en los

procesos electorales tradicionales, como el conteo de votos y la autentificación de los electores.

Los resultados del proceso electoral que se realice se obtendrán más rápido en comparación a los sistemas de votación tradicionales y con más confianza ante el votante, puesto que su voto podrá verificarse al momento de terminar de votar, o también realizando una auditoria en cada fase del proceso como antes de iniciar la votación o inmediatamente al culminar la jornada.

Estas características permiten que cada persona elegible a votar pueda hacerlo de una forma segura y confiable, dado que es esencial en una democracia que la comunidad vote. El voto electrónico es la mejor opción para que los adultos mayores, discapacitados y personas con diferentes dificultades para votar, puedan hacerlo, ejerciendo así, su derecho a votar de forma independiente. Por lo tanto, se fortalece el compromiso y la participación del elector y se reduce el porcentaje de votos anulados que se obtienen actualmente en un proceso electoral.

Por lo tanto, se puede decir que el objetivo de este trabajo investigativo se encuentra dentro de un marco novedoso puesto que se crea un nuevo espacio de uso el cual se sale del esquema de proceso electoral tradicional, utilizando diversos medios y herramientas tecnológicas disponibles con el fin de que la comunidad tenga las garantías necesarias frente a los procesos electorales.

Antecedentes y estado de conocimiento

Fundamento teórico

Arquitectura de software

La arquitectura de software se comprende como una guía de desarrollo para un sistema de software, esta define y describe la estructura del sistema de software que se va a implementar, así como, funcionamiento, con interacción y coordinación entre los componentes o piezas del mismo. De acuerdo con Buschmann, Meunier, Rohnert y Sommerland （Buschmann, 1996）, las componentes de un sistema de software representan las decisiones de diseño, los resultados de la organización y configuración de las componentes de la estructura del sistema, que garantiza que se satisfaga las necesidades de los clientes y usuarios.

Dentro del desarrollo de un sistema de software, se establece la definición de la arquitectura. Según Hofmeister, Nord y Soni （Christine Hofmeister, Robert Nord y

Dilip Soni, 1999), este propósito se logra con las siguientes etapas. La primera etapa corresponde al levantamiento de requerimientos, donde se muestran qué elementos y funciones son necesarias para el desarrollo de software, así como la captura, documentación y priorización de las necesidades que influencian la arquitectura. La segunda etapa es el diseño, la cual, determina las piezas y estructuras que conformarán el producto software. En esta etapa, además, se establece un marco de trabajo para gestión y control de los componentes que integran la arquitectura con base en tácticas de diseño que involucra la toma de decisiones. La tercera etapa es la documentación, donde se abarca todo el sistema e involucra la representación de las estructuras que son representadas a través de distintas vistas las cuales especifican los distintos aspectos técnicos y funcionales. Finalmente, se realiza la etapa de evaluación, la cual establece, un estudio factible que permite identificar posibles problemas y riesgos y establecer recomendaciones. Por otro lado, evaluar el diseño permite saber si el sistema cumple con las necesidades de los interesados, con esto se busca disminuir costos de corrección de defectos, ya que este es mucho menor a tener que corregirlos cuando el sistema haya sido terminado.

Así mismo la arquitectura de software es primordial en el desarrollo de un sistema software, porque determina su estructura y como este satisface los requisitos no funcionales o de calidad establecidos por los interesados, brindando una solución a un problema. Además, el adoptar un diseño arquitectónico permite la reutilización al crear sistemas distintos, esto tiene una ventaja muy importante debido a que reduce costos, aumenta la calidad, sobre todo si el diseño ha resultado ser exitoso en otros sistemas.

Atributo de calidad

La calidad del software se comprende como el conjunto de cualidades o características implícitas que se esperan de todo software, lo caracterizan y determinan su utilidad y existencia. Ba*RBAC*ci, Klein, Longstaff y Weinstock (BARBACCI, Mario; MARK, Klein; THOMAS A; LONGSTAFF, CHARLES, B; WEINSTOCK, 1995), definen la calidad de software, como el grado en el cual el software posee un conjunto de atributos o cualidades que le permiten ser lo que dice que se es. Por lo tanto, la calidad se entiende que son las cualidades, características adicionales que posee un sistema de software, estas características o atributos se definen como las propiedades de un servicio que presta el sistema el cual al realizar un proceso cumple con los requerimientos y satisface las necesidades de los usuarios.

Bass, Clements y Kazman （BASSy otros， 2013）, clasifican los atributos de calidad en dos categorías. La primera, corresponde a los atributos observables, los cuales son atributos que se determinan en el tiempo de ejecución de un sistema. Por ejemplo, desempeño, seguridad, disponibilidad, funcionalidad, usabilidad entre otros. La segunda categoría es la de los no observables, los cuales son atributos que se determinan en tiempo de desarrollo de un sistema. Por ejemplo, configurabilidad, integralidad, interoperabilidad, modificabilidad, mantenibilidad, portabilidad, reusabilidad, escalabilidad, testeabilidad, entre otros. Por otra parte, los atributos de calidad también son considerados funciones u objetivos específicos, los cuales proporcionan los resultados esperados por un usuario en el proceso de interacción con el producto software.

Seguridad

Según la norma ISO 25010, la seguridad, se entiende como una capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos （ISO/IEC）. Esta característica se subdivide a su vez en cinco subcaracterísticas. La primera subcaracterística es la confidencialidad, se define como la capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente. La segunda subcaracterística es la integridad, consiste en la capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador. La tercera subcaracterística es el no repudio, se define como la capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente. La cuarta subcaracterística es la responsabilidad, consiste en la capacidad de rastrear de forma inequívoca las acciones de una entidad. La quinta y última subcaracterística es la autenticidad, se define como la capacidad de demostrar la identidad de un sujeto o un recurso.

Estrategia de arquitectura

Para Grant （GRANT, R.， 2004）, una estrategia es un concepto que se aplica a diferentes ámbitos como por ejemplo el militar, el empresarial, el deportivo, entre otros. Esta desea alcanzar un fin mediante el desarrollo de un conjunto de acciones que se implementan en un contexto determinado. En la arquitectura de software, una estrategia es una guía que contiene un conjunto de tácticas de arquitectura de software para satisfacer los atributos de calidad y cumplir con uno o varios objetivos. En una estrategia de arquitectura, se planea, organiza, ejecuta y evalúa; se es consciente de las acciones que se realizan y de las decisiones que se toman, con el fin último de diseñar un producto software de calidad.

Táctica de arquitectura

Una táctica se comprende que es un plan a ejecutar, para lograr un objetivo y dar una solución exitosa y satisfacer una respuesta a un atributo de calidad. Las tácticas de arquitectura tienen como objetivo lograr satisfacer atributos de calidad en un sistema de software, se considera una arquitectura de software bien diseñada aquella que integra tácticas que permiten satisfacer las necesidades del sistema.

ANTECEDENTES DEL TEMA

A continuación, se describen algunos de los trabajos más representativos en la temática de *e-voting*.

Kareem M. AboSamra, Ahmed A. AbdelHafez, Ghazy M.R. Assassa y Mona F.M. Mursi （A practical, secure, and auditable e-voting system，2017），proponen un sistema de votación electrónica criptográfico para reemplazar los métodos de votación convencionales. Para ello, se basan en el concepto de Prêt à Voter. En este sistema hace uso de canales anónimos para anonimizar los votos en los esquemas basados en Mixnet, sin embargo, proporciona un nivel comparable de seguridad y anonimato de voto con menos complejidad del sistema. Se desarrolló una prueba de implementación de concepto y simulación del esquema de votación electrónica propuesta para dilucidar su eficacia, practicidad y escalabilidad con respuestas favorables.

En el trabajo realizado por Jeff Schmidt-Peralta y Jaime Gutiérrez-Alfaro （Hacia el desarrollo de un prototipo de sistema de voto electrónico para Costa Rica，2016）, se dice que las tecnologías digitales de comunicación e información se han incorporado de una forma transversal en muchas actividades cotidianas. Aquellas tecnologías de interés público se estudian con especial atención. Los procesos electorales, por su complejidad y requerimientos particulares, han venido adoptando la tecnología digital con recelo, pero sin cerrarle las puertas porque las oportunidades de mejorar y facilitar el proceso son alentadoras. En este trabajo se presenta el desarrollo de un prototipo de voto electrónico, donde usan software para su totalidad auditoria

Como es conocido, el desarrollo para que un producto software tenga éxito, se encuentra ligado al análisis de requerimiento realizado. En el trabajo realizado por Patricia Pesado, Guillermo Feierherd y Pasini Ariel （Patricia Pesado，2015），se presenta un análisis de la especificación de requerimientos en sistemas de voto electrónico. En este proyecto trabajan especificaciones no funcionales como seguridad, se discute una especificación que supone un modelo de arquitectura

física distribuida. Además, analizan la adaptación del modelo a clases de elecciones diferentes.

Jurlind Budurushi y Roman Jöris y Melanie Volkamer, hacen referencia en su trabajo （Implementing and evaluating a softwareindependent， 2014） al principio introducido por el Federal alemán “carácter público de las elecciones”, donde evidencian que los sistemas existentes no cumplen con este principio. Para ello, especifican y concretizan los procesos que no se cumplen en los sistemas existentes e implementan un prototipo. Además, evalúan el prototipo en un estudio de usuarios que realizaron junto con las elecciones universitarias en la Technische Universtität Darmstadt obteniendo buenos resultados en la satisfacción de los participantes

En relación con la privacidad de los votantes en un sistema de *e-voting*, Pablo Garcia, Germán Montejano, Silvia Bast y Estela Fritz （Garciay otros，2016），plantean que la protección del anonimato debe ser de mayor nivel que la que se otorgue al proceso electoral, dado que este último debe ser protegido por un periodo finito de tiempo, mientras que el anonimato debe asegurarse indefinidamente. En consecuencia, consideran probado que el anonimato será seguro aun cuando un criptoanalista cuente con tiempo y recursos ilimitados. Hacen énfasis en los avances realizados en el ámbito de la protección del anonimato y enuncian las acciones futuras a desarrollar.

1. DESARROLLO DEL PROYECTO - METODOLOGÍA

Se siguieron los siguientes pasos lógicos para alcanzar el conocimiento de forma sistemática y ordenada, como parte de la metodología para el desarrollo del proyecto planteado, contemplando 3 fases que son establecer tácticas arquitectónicas de seguridad, identificar tácticas arquitectónicas de seguridad para sistemas *e-voting* y la construcción de un producto software que las implemente

1.1. PROCESO DE INVESTIGACIÓN

A continuación, se describe el orden de ejecución de las actividades planificadas en detalle.

Tabla 1. Orden proceso de investigación

ACTIVIDADES LOGICAS ANTERIORES	ACTIVIDADES PLANIFICADAS			ACTIVIDADES LOGICAS POSTERIORES
	ORDEN	DETALLE	DURACIÓN EN SEMANAS	
-	A	Definir las fuentes de información	2	B
A	B	Elaborar las preguntas orientadoras	2	C
B	C	Ejecutar las consultas de los artículos	1	D
C	D	Seleccionar estudios	2	E
D	E	Revisar estudios en profundidad	3	F
E	F	Elaborar matriz de tácticas	5	G
F	G	Elaborar capítulo del informe de investigación	2	H
G	H	Reconocer las tácticas de seguridad en sistemas *e-voting*	6	I
H	I	Elaborar la matriz de comparación entre las tácticas de seguridad en sistemas *e-voting*	2	J
I	J	Identificar las tácticas arquitectónicas pertinentes que solucionaran los problemas de seguridad para sistemas *e-voting*	2	K
J	K	Analizar las tácticas arquitectónicas pertinentes que solucionaran los problemas de seguridad para sistemas *e-voting*	3	L
K	L	Identificar las historias de usuario priorizadas	2	M
L	M	Crear el *product backlog*	1	N
M	N	Definir el tiempo y número de *Sprints* a realizar	1	O

ACTIVIDADES LOGICAS ANTERIORES	ACTIVIDADES PLANIFICADAS			ACTIVIDADES LOGICAS POSTERIORES
	ORDEN	DETALLE	DURACIÓN EN SEMANAS	
N	O	Desarrollar *Sprint* 1	2	P
O	P	Desarrollar *Sprint* 2	2	Q
P	Q	Desarrollar *Sprint* 3	2	R
Q	R	Desarrollar *Sprint* 4	2	S
R	S	Desarrollar *Sprint* 5	2	T
S	T	Desarrollar *Sprint* 6	2	U
T	U	Realizar el informe final sobre las tácticas arquitectónicas de seguridad para sistemas *e-voting*.	4	–

Fuente: Esta investigación - 2020.

1.2 RESULTADOS ESPERADOS

A continuación, se presentan los productos resultados de la investigación.

- Matriz de tácticas arquitectónicas de seguridad.
- Matriz de comparación entre tácticas arquitectónicas de seguridad para sistemas *e-voting*.
- Producto software *e-voting* que implementa las mejores tácticas arquitectónicas de seguridad.

1.3 RECURSOS

Aquí se detallan los recursos usados a lo largo del desarrollo del Proyecto. Discriminados por recursos HUMANOS, TECNOLÓGICOS, MATERIALES y FINANCIEROS.

1.4 RECURSOS HUMANOS

Tabla 2. Recursos Humanos

Persona Encargada	Horas/semana	Valor hora	Horas Totales	Total, proyecto (8 meses)
M,Sc. Giovanni Hernández. Docente Programa Ingeniería de Sistemas	2	$40.000	80	$3’200.000
M,Sc. Sandra Vallejo Docente Programa Ingeniería de Sistemas	2	$40.000	80	$3’200.000
Daniel Esteban Burbano Estudiante Ingeniería de Sistemas	20	$18.000	800	$14’400.000
Francisco Javier Zambrano Estudiante Ingeniería de Sistemas	20	$18.000	800	$14’400.000
		Total	1.760	$35’200.000

Fuente: Esta investigación - 2020.

Valores estimados ya que el trabajo se realizará en el marco de desarrollo de trabajo de grado

1.5 RECURSOS TECNOLÓGICOS

Tabla 3. Recursos Tecnológicos

Tipo	Descripción	Valor hora	Cantidad	Valor total
Hardware	2 computadores portátiles, Sistema Operativo Windows 10 Profesional, procesador Intel Core i5, 4Gb de RAM, 1Tb de disco duro	$1.000	583.233	$2’959.800
	Servidor IBM System 3650 M2, procesador Intel Xeon QuadCore 2.8GHz, 70GB RAM, 2.6 TB Disco Duro	$400	4.380 (6 meses)	$1’752.000
Software	Apache, PHP y MySQL/PostgreSQL	Gratis	Gratis	Gratis
Servicios	Internet	$1.000	$640	$640.000
Servicios	Energía Eléctrica	$700	$640	$448.000
			Total	$5’799.800

Fuente: Esta investigación - 2020.

Todo el software utilizado para esta investigación es libre por lo cual no requiere ningún costo de licencia para su uso.

1.6 RECURSOS MATERIALES

Tabla 4. Recursos Materiales

Articulo	Cantidad	Valor individual	Valor total
Memoria USB	2	$20.000	$40.000
Elementos de papelería (resmas de papel, tóner para impresión, hojas, lapiceros…)	2	$35.000	$70.000
Asistencia a congresos	2	$750.000	$1’500.000
	Total	$805.000	$1’610.000

Fuente: Esta investigación - 2020.

1.7 RECURSOS FINANCIEROS

Tabla 5. Recursos Financieros

Ítem	Valor
Recursos de personal	$35’200.000
Recursos tecnológicos	$5’799.800
Recursos materiales	$1’610.000
Imprevistos	$500.000
Total	$43’109.800

Fuente: Esta investigación - 2020.

1.8 CRONOGRAMA

A continuación, se presenta el cronograma de manera general de las actividades implementadas en el desarrollo del proyecto.

Tabla 6. Cronograma

Actividades	Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6				Mes 7				Mes 8				Mes 9				Mes 10				Mes 11				Mes 12			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Definir las fuentes de información																																															
Elaborar las preguntas orientadoras																																															
Ejecutar las consultas de los artículos																																															
Seleccionar estudios																																															
Revisar estudios en profundidad																																														
Elaborar matriz de tácticas																																														
Elaborar capítulo del informe de investigación																																															
Reconocer las tácticas de seguridad en sistemas *e-voting*																																															
Elaborar la matriz de comparación entre las tácticas de seguridad en sistemas *e-voting*																																															

Actividades	Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6				Mes 7				Mes 8				Mes 9				Mes 10				Mes 11				Mes 12			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Identificar las tácticas arquitectónicas pertinentes que solucionaran los problemas de seguridad para sistemas *e-voting*																																															
Analizar las tácticas arquitectónicas pertinentes que solucionaran los problemas de seguridad para sistemas *e-voting*																																															
Identificar las historias de usuario priorizadas																																															
Crear el *product backlog*																																															
Definir el tiempo y número de *Sprints* a realizar																																															
Desarrollar *Sprint* 1																																															
Desarrollar *Sprint* 2																																															
Desarrollar *Sprint* 3																																															
Desarrollar *Sprint* 4																																															
Desarrollar *Sprint* 5																																															

Actividades	Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6				Mes 7				Mes 8				Mes 9				Mes 10				Mes 11				Mes 12			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Desarrollar *Sprint* 6																																															
Realizar el informe final sobre las tácticas arquitectónicas de seguridad para sistemas *e-voting*.																																															

Fuente: Esta investigación - 2020.

2. RESULTADOS

En este capítulo se presentan los resultados obtenidos en el desarrollo del proyecto de investigación. El primer objetivo buscó establecer tácticas arquitectónicas de seguridad utilizadas en la construcción de productos software a través de la revisión sistemática de literatura. Luego se identificó tácticas arquitectónicas de seguridad pertinentes para un producto software de *e-voting*. Finalmente se construyó un producto software para *e-voting* que implementó las tácticas arquitectónicas de seguridad pertinentes para *e-voting*.

2.1 TÁCTICAS ARQUITECTÓNICAS DE SEGURIDAD, UNA REVISIÓN SISTEMÁTICA

En esta sección se describe como se estableció las tácticas de seguridad pertinentes en la construcción de productos software.

Para lograr este objetivo, se utilizó como técnica la revisión sistemática de literatura (LRS) descrito en （Kitchenhamy otros，2015）, que consta de las etapas de, elaborar las preguntas de investigación, realizar un proceso de búsqueda, llevar a cabo un proceso de selección y hacer un proceso de evaluación de la calidad. Finalmente se hace una síntesis de los resultados.

.

2.1.1 Preguntas de investigación

Las preguntas de investigación formuladas se basan en la pregunta principal ¿Cuáles son las tácticas arquitectónicas de seguridad que aportan a la construcción de un producto software para *e-voting*?

A partir de la pregunta principal, se derivaron las preguntas secundarias que permitieron analizar y categorizar los estudios primarios:

- **RQ1.**

  ¿Qué tácticas arquitectónicas de seguridad son utilizadas en la construcción de software?

- **RQ2.**

  ¿Qué tácticas arquitectónicas de seguridad son utilizadas en *e-voting*?

2.1.2 Proceso de búsqueda

Como parte del protocolo para buscar estudios primarios, se identificó las fuentes de información y se definió la cadena de búsqueda según la pregunta de investigación principal. Las siguientes bases de datos, se utilizaron como fuentes de información: *ACM digital library, IEEE explorer* y *Springer*.

La cadena general creada para establecer los criterios de búsqueda se configura a partir de los datos mostrados en la Tabla 7.

Tabla 7. Criterios de búsqueda

Concepto	Palabras relacionadas
Tactic	Strategy OR Technique
Architectural	Model OR Architecture
Security	Protection OR Defense OR Prevention
Software development	(Software OR System) AND (Development OR Creation OR Construction)

Fuente: Esta investigación - 2020.

La cadena de búsqueda "*architectural security tactic in software development*" fue elaborada construyendo expresiones que utilizan los operadores booleanos *OR* y *AND*. El operador *OR*, se lo utilizó para incorporar sinónimos del concepto de búsqueda, mientras que el operador *AND* permite agregar las palabras relacionadas en la cadena de búsqueda.

Para la ejecución de las búsquedas, se examinó: título, resumen y palabras clave, dentro de los resultados obtenidos a través del uso de los motores de búsqueda de cada fuente de datos seleccionada. La revisión de artículos se limitó a aquellos, que se encuentren escritos en inglés y dentro de una ventana de observación entre el 2014 y 2019.

2.1.3 Proceso de selección

Después de realizar el proceso de búsqueda, se procedió a seleccionar los estudios relevantes, es decir, aquellos artículos que permiten dar respuesta a las preguntas de investigación planteadas. Para determinar la relevancia de los estudios, se

establecieron unos criterios de inclusión y exclusión, como se pueden observar en la Tabla 8.

Tabla 8. Criterios de selección de estudios

Criterios de inclusión	Criterios de exclusión
1. Artículos que se relacionan con tácticas arquitectónicas de seguridad. 2. Artículos publicados en una ventana de observación entre 2014 y 2019 3. Artículos escritos en inglés 4. Artículos resultados de estudios primarios	1. Artículos que no se relacionen con tácticas arquitectónicas de seguridad 2. Reportes técnicos 3. Estudios duplicados

Fuente: Esta investigación - 2020.

Para la aplicación de los criterios de inclusión y exclusión, como parte de la selección, se definieron los filtros que se muestran en la Tabla 9.

Tabla 9. Estrategia de selección

Filtro	Descripción	Criterio aplicado	
		Inclusión	Exclusión
1F	Buscar los artículos aplicando la cadena de búsqueda en los motores de las fuentes seleccionadas.	3 y 4	
2F Extracción de información	Leer el título, palabras clave y resumen del	1, 2 y 5	1, 2 y 3

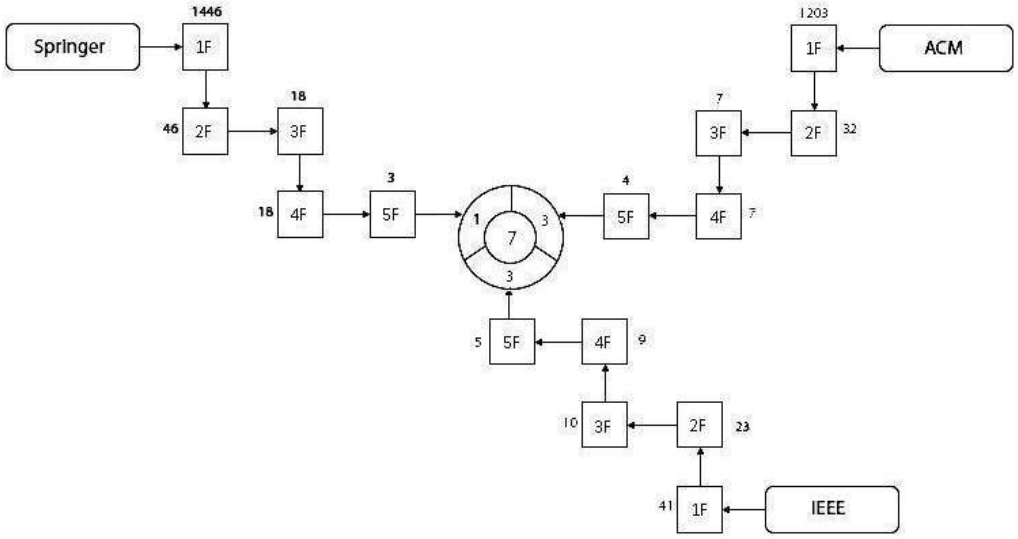
Filtro	Descripción	Criterio aplicado	
		Inclusión	Exclusión
	artículo aplicando los criterios de inclusión y exclusión.		
3F	Leer los resultados y conclusiones del artículo aplicando los criterios de inclusión y exclusión.	1, 2 y 5	1, 2 y 3
4F	Eliminar los estudios duplicados.		4
5F	Leer el artículo completo y aplicar los criterios de inclusión y exclusión.	1, 2 y 5	1, 2 y 3

Fuente: Esta investigación - 2020.

En la

Figura 1, se puede observar los resultados obtenidos de la aplicación de los filtros definidos en la Tabla 9.

Figura 1. Resultados de la revisión sistemática



Fuente: Esta investigación - 2020.

2.1.4 Proceso de evaluación de la calidad

Posterior al proceso de selección, se realizó una nueva tarea para asegurar la calidad de los artículos encontrados, que corresponde a inspeccionar en los artículos seleccionados, un conjunto de criterios que se muestran en la Tabla 10:

Tabla 10. Criterios de evaluación de la calidad

Criterio	Descripción	Categoría
C1	Los objetivos y preguntas de investigación se describen de forma explícita, son claros y relevantes.	Calidad del reporte
C2	La investigación presenta un diseño metodológico que le permite alcanzar los objetivos.	Rigor
C3	El procedimiento de recopilación de datos es coherente con el diseño metodológico.	Rigor
C4	Los resultados presentados son claros y coherentes con el diseño metodológico propuesto.	Credibilidad
C5	El estudio es valorado por otros investigadores.	Relevancia

Fuente: Esta investigación - 2020.

Para evaluar la calidad de los artículos, se estableció una escala para inspeccionar el nivel de cumplimiento de los criterios, de la siguiente manera: Alto (2 puntos), Medio (1 punto) y Bajo (0 puntos).

Tabla 11. Matriz de evaluación de la calidad de los artículos

Título del artículo	C1	C2	C3	C4	C5	$\sum_{i=1}^{5} Ci$	%Ci
Mitigating Security Threats Using Tactics and Patterns: A Controlled Experiment	2	2	2	1	0	7	70%
Security Tactics Selection Poker (TaSPeR)	2	2	2	1	0	7	70%

Título del artículo	C1	C2	C3	C4	C5	$\sum_{i=1}^{5} Ci$	%Ci
A Methodological Approach to Apply Security Tactics in Software *Architecture Design*	2	2	2	1	2	9	90%
Building Sustainable Software *by Preemptive Architectural Design Using Tactic-Equipped Patterns*	2	2	2	1	2	9	90%
Detecting, Tracing, and Monitoring Architectural Tactics in Code	2	2	2	2	2	10	100 %
Understanding Software *Vulnerabilities Related to Architectural Security Tactics*	2	2	2	2	2	10	100 %
Revisiting architectural tactics for security	2	2	2	2	1	9	90%

Fuente: Esta investigación – 2020.

Los artículos que cumplieron con un 80% o más, del total de los puntos posibles, son lo que finalmente se eligieron. Al finalizar el proceso de evaluación de la calidad de los artículos, se incluyeron 5 artículos.

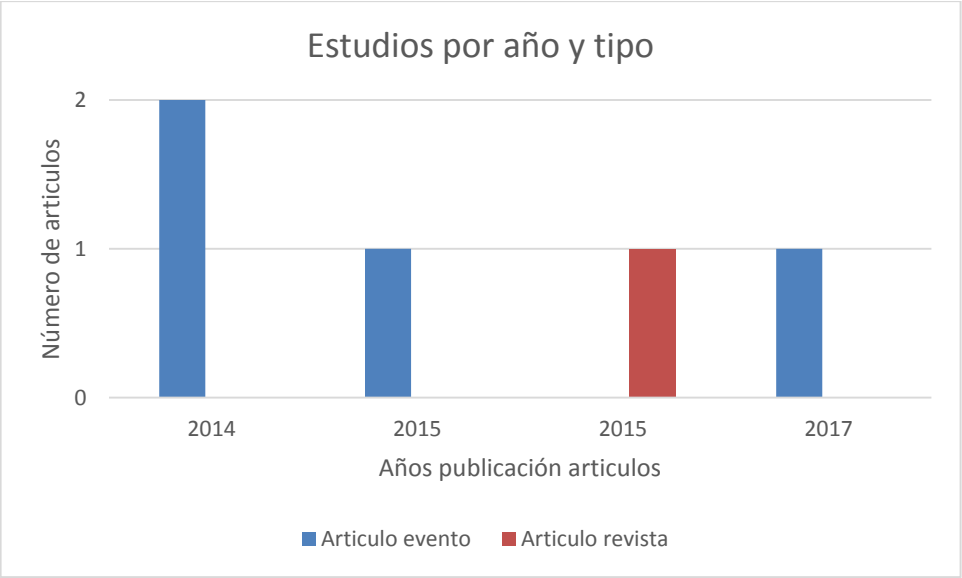
2.1.5 Resultados

A continuación, se muestra una síntesis de los resultados obtenidos en el proceso de desarrollo del primer objetivo, el cual se describe en el siguiente orden: primero se puede observar una gráfica que concluye los estudios discriminados por año y tipo, luego, se presenta una serie de tablas, en ellas se puede observar datos específicos de los artículos seleccionados en la sección anterior, finalmente, se puede observar una categorización de los ítems evaluados en las tablas de los artículos, mencionadas anteriormente.

2.1.6 Distribución de los artículos de estudio

Se analizaron cinco estudios primarios. En la Figura 2, se muestra una distribución de los documentos incluidos por año y tipo. El análisis incluye 4 artículos de conferencia (80%) y 1 artículos de revistas (20%).

Figura 2. Estudios por año y tipo



Fuente: Esta investigación – 2020.

2.1.7 Resultados preguntas de investigación

A continuación, se presentan los resultados encontrados en relación con las preguntas de investigación que se presentan en la Tabla 12.

Tabla 12. Preguntas de investigación

Categoría	Descripción	Pregunta de inv.
Tipo de táctica de seguridad	Forma como se agrupan las tácticas arquitectónicas de seguridad	RQ1
Campo de uso	Entorno donde se está utilizando la táctica arquitectónica de seguridad (Industria, gobierno, academia, otro)	RQ1
Práctica en la táctica	Buenas prácticas propuestas por la táctica arquitectónica de seguridad	RQ1
Tipo de táctica de seguridad en *e-voting*	Forma como se agrupan las tácticas arquitectónicas de seguridad en *e-voting*	RQ2
Recurso para la táctica de seguridad en *e-voting*	Elementos necesarios para poder implementar la táctica arquitectónica de seguridad en *e-voting*	RQ2
Efecto de la táctica de seguridad en *e-voting*	Resultados obtenidos de la implementación de la táctica arquitectónica de seguridad en *e-voting*	RQ2

Fuente: Esta investigación - 2020.

10.1.5.3. Resultados de las categorías definidas por artículo

Por cada artículo se extrajo la información relacionada con las categorías definidas en la Tabla 12. Los resultados se pueden observar en la Tabla 13 a la Tabla 17.

Tabla 13. *A Methodological Approach to Apply Security Tactics in* Software *Architecture Design*

Categoría	**Descripción**	**Pregunta de inv.**
Tipo de táctica de seguridad	Señal digital Autenticar remitente Autorizar actor Redundancia de servicios Rastrear acciones con *logger* Encriptación	RQ1
Campo de uso	Gobierno	RQ1
Práctica en la táctica	Evalúa casos de uso Especifica la vista funcional en capas Uso de modelo de datos Identifica recursos sensibles Define políticas de seguridad Uso de modelo de amenaza (árbol)	RQ1
Tipo de táctica de seguridad en *e-voting*	Autenticar remitente Autorizar actor Encriptación	RQ2
Recurso para la táctica de seguridad en *e-voting*	Experto seguridad Responsable de tomar decisiones Componentes de redes para interoperabilidad Bases de datos	RQ2

Categoría	Descripción	Pregunta de inv.
Efecto de la táctica de seguridad en *e-voting*	Permite llevar un control de los actores implicados y de la información del sistema, además que el sistema tiene un mejor funcionamiento gracias a las decisiones tomadas por el experto, con una red dedicada, los datos se transmiten a una velocidad mayor, permitiendo un acceso rápido.	RQ2

Fuente: Esta investigación - 2020.

Tabla 14. *Building Sustainable* Software *by Preemptive Architectural Design Using Tactic-Equipped Patterns*

Categoría	Descripción	Pregunta de inv.
Tipo de táctica de seguridad	Resistencia a ataques: Autenticación de usuarios Autorización de usuarios Confidencialidad de los datos Integridad Limitación exposición de servicios Limitar acceso Detección de ataques: Detección de intrusos Recuperación de ataques Restauración Auditoría	RQ1
Campo de uso	Academia	RQ1

Categoría	Descripción	Pregunta de inv.
Práctica en la táctica	Identificación de anti-patrones Identificación patrones compatibles Visualización de la arquitectura Administración de requerimientos Control de la seguridad Reutilización	RQ1
Tipo de táctica de seguridad en *e-voting*	Resistencia a ataques: Autenticación de usuarios Autorización de usuarios Confidencialidad de los datos Integridad Limitación/Restricción de servicios Limitar acceso Detección de ataques: Detección de intrusos Recuperación de ataques Restauración Auditoría	RQ2
Recurso para la táctica de seguridad en *e-voting*	*Firewalls* y *DMZ* Sistemas biométricos Certificados o firmas digitales Sistemas de cifrado Parámetros de acceso o niveles tipo *Grant*	RQ2

Categoría	Descripción	Pregunta de inv.
	Controles de cifrado (*DES*, *3DES*, *AES*) Algoritmos de cifrado (simétricos, cuánticos, asimétricos, de curva elíptica o híbridos) Mecanismos de *Checksum* y códigos *Hash* Copias redundantes de los datos Uso de patrones *Tunneling* y configuración de *VPN*	
Efecto de la táctica de seguridad en *e-voting*	Garantizan que un usuario tenga acceso al sistema de información mediante un conjunto de reglas de identificación y verificación de acceso, protegen la integridad de la información del usuario y su confidencialidad, permiten segmentar la red aislando los servidores, además de bloquear el acceso no autorizado, detectar posibles intrusiones, recuperación del sistema mediante copias redundantes y auditar el sistema mediante copias de respaldo de cada transacción realizada en el sistema	RQ2

Fuente: Esta investigación - 2020.

Tabla 15. *Detecting, Tracing, and Monitoring Architectural Tactics in Code*

Categoría	**Descripción**	**Pregunta de inv.**
Tipo de táctica de seguridad	Auditoría Autenticación *HMAC* Sesiones seguras Administración y *RBAC*	RQ1
Campo de uso	Industria	RQ1
Práctica en la táctica	Verificación permanente de la calidad Visualización de la arquitectura Administración de requerimientos Pruebas unitarias Control de la seguridad	RQ1
Tipo de táctica de seguridad en *e-voting*	Auditoría Autenticación *HMAC* Sesiones seguras Administración y *RBAC*	RQ2
Recurso para la táctica de seguridad en *e-voting*	Mecanismos de *Checksum* y códigos *Hash* Certificados o firmas digitales Parámetros de acceso o niveles tipo *Grant* Listas de usuarios Roles del sistema	RQ2

Categoría	Descripción	Pregunta de inv.
	Dominios Controles de cifrado (*DES*, *3DES*, *AES*) Copias redundantes de los datos *Tunneling* y configuración de *VPN* *kernels* y *shells* de seguridad	
Efecto de la táctica de seguridad en *e-voting*	Permiten auditar el sistema mediante copias de respaldo de cada transacción realizada en el sistema, ingresos al sistema seguros mediante un proceso de autorización y autenticación, la información de los usuarios está segura mediante controles de cifrado, permite establecer los permisos de los usuarios mediante el control de acceso basado en roles, se logra verificar la integridad de la información transmitida	RQ2

Fuente: Esta investigación - 2020.

Tabla 16. *Understanding* Software *Vulnerabilities Related to Architectural Security Tactics*

Categoría	Descripción	Pregunta de inv.
Tipo de táctica de seguridad	Identificar actores Validar entradas Administrar sesiones de usuario Autenticar actores Autorizar actores	RQ1

Categoría	Descripción	Pregunta de inv.
	Limitar acceso Limitar exposición Encriptar datos Separar entidades Cambiar configuración por defecto Informar actores Detectar ataque por negación de servicio Detectar intruso Verificar integridad del mensaje Auditar	
Campo de uso	Industria	RQ1
Práctica en la táctica	Visualización de la arquitectura Administración de requerimientos Pruebas unitarias Control de la seguridad	RQ1
Tipo de táctica de seguridad en *e-voting*	Identificar actores Validar entradas Administrar sesiones de usuario Autenticar actores Autorizar actores Limitar acceso Limitar exposición Encriptar datos	RQ2

Categoría	Descripción	Pregunta de inv.
	Separar entidades Cambiar configuración por defecto Informar actores Detectar ataque por negación de servicio Detectar intruso Verificar integridad del mensaje Auditar	
Recurso para la táctica de seguridad en *e-voting*	Monitores de red y registro de eventos Sistemas biométricos Certificados o firmas digitales Sistemas de cifrado Controles de cifrado (*DES*, *3DES*, *AES*) Mecanismos de *Checksum* y códigos *Hash*	RQ2
Efecto de la táctica de seguridad en *e-voting*	protegen la integridad de la información del usuario y su confidencialidad mediante la encriptación de los datos, se logra identificar intrusiones, se brinda el acceso solo a usuarios autenticados y se establecen limitaciones, se valida los datos ingresados por los usuarios y es posible auditar el sistema mediante copias de respaldo de cada transacción realizada en el sistema	RQ2

Fuente: Esta investigación - 2020.

Tabla 17. *Revisiting architectural tactics for security*

Categoría	**Descripción**	**Pregunta de inv.**
Tipo de táctica de seguridad	Detectar ataques Verificar integridad del mensaje Verificar la integridad del almacenamiento Mantener pista de auditoría Identificar intruso Por firma Por comportamiento Detener o mitigar ataques Autenticar sujetos Autorizar sujetos Detectar origen de mensajes Establecer un canal seguro Gestionar datos de seguridad Ocultar información Por encriptación Por esteganografía Reaccionar a los ataques Alertar sujetos Aplicar políticas de la institución Recuperarse de los ataques Auditar acciones	RQ1

Categoría	Descripción	Pregunta de inv.
	Aplicar políticas de la institución	
Campo de uso	Academia	RQ1
Práctica en la táctica	Uso de políticas de seguridad Jerarquía de políticas Seguridad aplicada a todo el sistema Integridad de servicios Integridad de mensajes Identificar, autenticar, autorizar autores Limitar acceso Cifrar datos Informes del sistema Auditoría Mantener disponibilidad Complementar la seguridad del sistema con patrones de seguridad	RQ1
Tipo de táctica de seguridad en *e-voting*	Detectar ataques Verificar integridad del mensaje Verificar la integridad del almacenamiento Mantener pista de auditoría Identificar intruso Por firma Por comportamiento	RQ2

Categoría	Descripción	Pregunta de inv.
	Detener o mitigar ataques Autenticar sujetos Autorizar sujetos Detectar origen de mensajes Establecer un canal seguro Gestionar datos de seguridad Ocultar información Por encriptación Por esteganografía Reaccionar a los ataques Alertar sujetos Aplicar políticas de la institución Recuperarse de los ataques Auditar acciones Aplicar políticas de la institución	
Recurso para la táctica de seguridad en *e-voting*	Sistemas de control y autenticación Sistemas de cifrado de datos Algoritmos de cifrado Red con canales seguros Patrones de seguridad de apoyo Hardware de nueva generación	RQ2
Efecto de la táctica de seguridad en *e-voting*	Permiten tener un acceso más confiable de actores al sistema, con permisos referentes a su puesto y rango, además al tener cifrados los datos, es poco probable que un	RQ2

Categoría	Descripción	Pregunta de inv.
	actor externo pueda extraer información del sistema, por otra parte, con hardware de nueva generación, el sistema será más robusto y complementado con un canal dedicado, la disponibilidad de los datos será mayor.	

Fuente: Esta investigación - 2020.

10.1.5.4. Tipo de táctica de seguridad

Los tipos de tácticas de seguridad en cada uno de los artículos mencionados en las tablas 13 a la 17, según Bass y otros (2013) se pueden clasificar en las siguientes categorías: detectar ataques, resistir ataques, reaccionar ante ataques, recuperarse de los ataques y las tácticas con mayor frecuencia mencionadas en los artículos fueron: autenticar actores, autorizar actores, auditoria.

10.1.5.5. Campo de uso

Por otro lado, se puede ver que, al estudiar las anteriores tablas, los campos de uso principales son industria y academia, a su vez, los recursos generalmente empleados para las tácticas fueron: algoritmos de cifrados, copias redundantes de datos, sistemas biométricos y certificados digitales.

10.1.5.6. Práctica en la táctica

Al analizar las tablas 13 a la 17 se logró concluir que las principales prácticas en las tácticas se enfocan en definir políticas de seguridad, uso de modelos de datos y amenazas, administración de requerimientos, visualización y pruebas unitarias.

10.1.5.7. Tipo de táctica de seguridad en *e-voting*

Cómo se mencionó en el punto 10.1.5.3., las tácticas se agrupan en categorías, que según Bass y otros (2013) son: detectar ataques, resistir ataques, reaccionar ante ataques, recuperarse de los ataques, para el caso de *e-voting*, se puede observar en las anteriores tablas que en su mayoría se nombran: la autenticación y autorización de usuarios, la detección de intrusos y la auditoria.

10.1.5.8. Recurso para la táctica de seguridad en *e-voting*

Los principales recursos para las tácticas de seguridad en *e-voting* mencionadas en las tablas 13 a la 17 se encuentra que: las copias redundantes de los datos, sistemas de cifrado y gestión de roles.

10.1.5.9. Efecto de la táctica de seguridad en *e-voting*

Al comparar las tablas 13 a la 17, se puede observar que los efectos con mayor frecuencia fueron el control de acceso, se obtiene un sistema más robusto y veloz, además, cabe resaltar la recuperación y la auditoria del sistema.

2.1.6 Tácticas arquitectónicas de seguridad utilizadas en la construcción de productos software

Los artículos que finalmente fueron seleccionados, se los puede observar en la Tabla 18. El 80% de los artículos corresponden a actas que se generan como memorias presentadas en eventos. Estos estudios están en un rango del 2014 al 2017 y, además, IEEE es la principal fuente de información.

Tabla 18. Estudios seleccionados

ID	Estudio	Tipo de artículo	Año	Fuente
S1	*A Methodological Approach to Apply Security Tactics in* Software *Architecture Design*	Acta de congreso	2014	*IEEE*
S2	*Building Sustainable* Software *by Preemptive Architectural Design Using Tactic-Equipped Patterns*	Acta de congreso	2014	*IEEE*
S3	*Detecting, Tracing, and Monitoring Architectural Tactics in Code*	Artículo publicado en revista	2015	*IEEE*
S4	*Understanding* Software *Vulnerabilities Related to Architectural Security Tactics*	Acta de congreso	2017	*IEEE*
S5	*Revisiting architectural tactics for security*	Acta de congreso	2015	*Springer*

Fuente: Esta investigación - 2020.

2.1.6.1 Tácticas arquitectónicas de seguridad

Las tácticas arquitectónicas de seguridad identificadas al inspeccionar los artículos seleccionados se pueden observar en la Tabla 19.

Tabla 19. Tácticas arquitectónicas de seguridad identificadas

Estudio	Tipo de táctica de seguridad	Campo de uso	Práctica en la táctica de seguridad
S1	Señal digital Autenticar remitente Autorizar actor Redundancia de servicios Rastrear acciones con *logger* Encriptación	Gobierno	Evalúa casos de uso Especifica la vista funcional en capas Uso de modelo de datos Identifica recursos sensibles Define políticas de seguridad Uso de modelo de amenaza (árbol)
S2	Resistencia a ataques: Autenticación de usuarios Autorización de usuarios Confidencialidad de los datos Integridad Limitación exposición de servicios Limitar acceso Detección de ataques: Detección de intrusos	Academia	Identificación de anti-patrones Identificación patrones compatibles Visualización de la arquitectura Administración de requerimientos Control de la seguridad Reutilización

Estudio	Tipo de táctica de seguridad	Campo de uso	Práctica en la táctica de seguridad
	Recuperación de ataques Restauración Auditoría		
S3	Auditoría Autenticación *HMAC* Sesiones seguras Administración y *RBAC*	Industria	Verificación permanente de la calidad Visualización de la arquitectura Administración de requerimientos Pruebas unitarias Control de la seguridad
S4	Identificar actores Validar entradas Administrar sesiones de usuario Autenticar actores Autorizar actores Limitar acceso Limitar exposición Encriptar datos Separar entidades Cambiar configuración por defecto Informar actores Detectar ataque por negación de servicio Detectar intruso Verificar integridad del mensaje	Industria	Visualización de la arquitectura Administración de requerimientos Pruebas unitarias Control de la seguridad

Estudio	Tipo de táctica de seguridad	Campo de uso	Práctica en la táctica de seguridad
	Auditar		
S5	Detectar ataques Verificar integridad del mensaje Verificar la integridad del almacenamiento Mantener pista de auditoría Identificar intruso Por firma Por comportamiento Detener o mitigar ataques Autenticar sujetos Autorizar sujetos Detectar origen de mensajes Establecer un canal seguro Gestionar datos de seguridad Ocultar información Por encriptación Por esteganografía Reaccionar a los ataques Alertar sujetos Aplicar políticas de la institución Recuperarse de los ataques Auditar acciones Aplicar políticas de la institución	Academia	Uso de políticas de seguridad Jerarquía de políticas Seguridad aplicada a todo el sistema Detectar intruso Integridad de servicios Integridad de mensajes Identificar, autenticar, autorizar autores Limitar acceso Cifrar datos Informes del sistema Auditoría Mantener disponibilidad Complementar la seguridad del sistema con patrones de seguridad

Fuente: Esta investigación - 2020.

Se puede evidenciar en la Tabla 19, que los tipos de tácticas de seguridad más frecuentes en los estudios seleccionados son: detección, resistencia y recuperación, de ataques. Por otra parte, los campos de uso principales son industria y academia, a su vez, los recursos generalmente empleados para las tácticas son: algoritmos de cifrados, copias redundantes de datos, sistemas biométricos y certificados digitales.

2.1.6.2 Tácticas arquitectónicas de seguridad en *E-voting*

Las tácticas arquitectónicas de seguridad en *e-voting* identificadas al inspeccionar los artículos seleccionados, se pueden observar en la Tabla 20.

Tabla 20. Tácticas arquitectónicas de seguridad para *e-voting* identificadas

Estudio	Tipo de táctica de seguridad en *e-voting*	Recurso para la táctica
S1	Autenticar remitente Autorizar actor Encriptación	Experto seguridad Responsable de tomar decisiones Componentes de redes para interoperabilidad Bases de datos
S2	Resistencia a ataques: Autenticación de usuarios Autorización de usuarios Confidencialidad de los datos Integridad Limitación/Restricción de servicios Limitar acceso Detección de ataques: Detección de intrusos Recuperación de ataques Restauración Auditoría	*Firewalls* y *DMZ* Sistemas biométricos Certificados o firmas digitales Sistemas de cifrado Parámetros de acceso o niveles tipo *Grant* Controles de cifrado (*DES*, *3DES*, *AES*) Algoritmos de cifrado (simétricos, cuánticos, asimétricos, de curva elíptica o híbridos) Mecanismos de *Checksum* y códigos *Hash* Copias redundantes de los datos Uso de patrones *Tunneling* y configuración de *VPN*

S3	Auditoría Autenticación *HMAC* Sesiones seguras Administración y *RBAC*	Certificados o firmas digitales Parámetros de acceso o niveles tipo *Grant* Listas de usuarios Roles del sistema Dominios Controles de cifrado (*DES*, *3DES*, *AES*) Copias redundantes de los datos *Tunneling* y configuración de *VPN* *kernels* y *shells* de seguridad
S4	Identificar actores Validar entradas Administrar sesiones de usuario Autenticar actores Autorizar actores Limitar acceso Limitar exposición Encriptar datos Separar entidades Cambiar configuración por defecto Informar actores Detectar ataque por negación de servicio Detectar intruso Verificar integridad del mensaje	Monitores de red y registro de eventos Sistemas biométricos Certificados o firmas digitales Sistemas de cifrado Controles de cifrado (*DES*, *3DES*, *AES*) Mecanismos de *Checksum* y códigos *Hash* Copias redundantes de los datos

	Auditar	
S5	Detectar ataques Verificar integridad del mensaje Verificar la integridad del almacenamiento Mantener pista de auditoría Identificar intruso Por firma Por comportamiento Detener o mitigar ataques Autenticar sujetos Autorizar sujetos Detectar origen de mensajes Establecer un canal seguro Gestionar datos de seguridad Ocultar información Por encriptación Por esteganografía Reaccionar a los ataques Alertar sujetos Aplicar políticas de la institución Recuperarse de los ataques Auditar acciones Aplicar políticas de la institución	Sistemas de control y autenticación Sistemas de cifrado de datos Algoritmos de cifrado Red con canales seguros Patrones de seguridad de apoyo Hardware de nueva generación

Fuente: Esta investigación - 2020.

De la Tabla 20, se puede deducir que entre el tipo de tácticas de seguridad en *e-voting* que más se mencionan, se relacionan con, detectar, resistir, reaccionar y recuperarse, ante ataques. Con respecto a los recursos para la táctica, entre los principales están: las copias redundantes de los datos, sistemas de cifrado y roles en el equipo de trabajo.

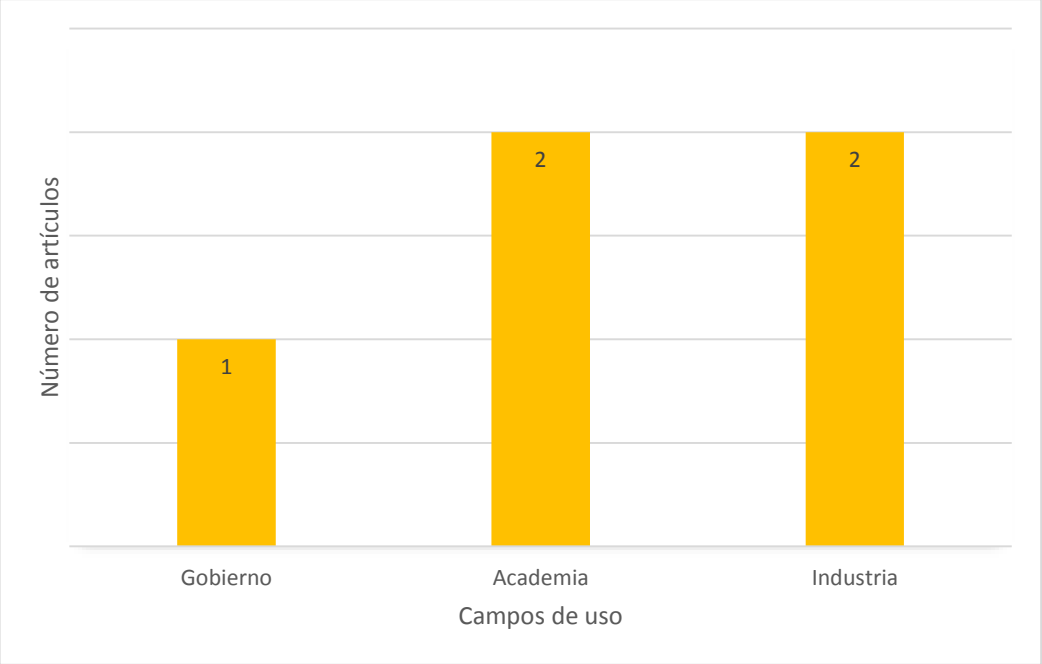
2.1.6.3 Síntesis de las tácticas arquitectónicas de seguridad de *e-voting*

Después de analizar los datos obtenidos de cada uno de los artículos objeto de estudio, se procedió a contrastar y comparar dichos resultados. Estos resultados se muestran a continuación:

- Campos de uso de los artículos seleccionados

Se puede evidenciar que, los campos de uso de los artículos seleccionados, el 20% de los artículos se enfocan en el campo de gobierno, mientras que el 40% en la academia y el 40% restante en la industria como se puede ver en la Figura 3.

Figura 3. Campos de uso de los artículos seleccionados



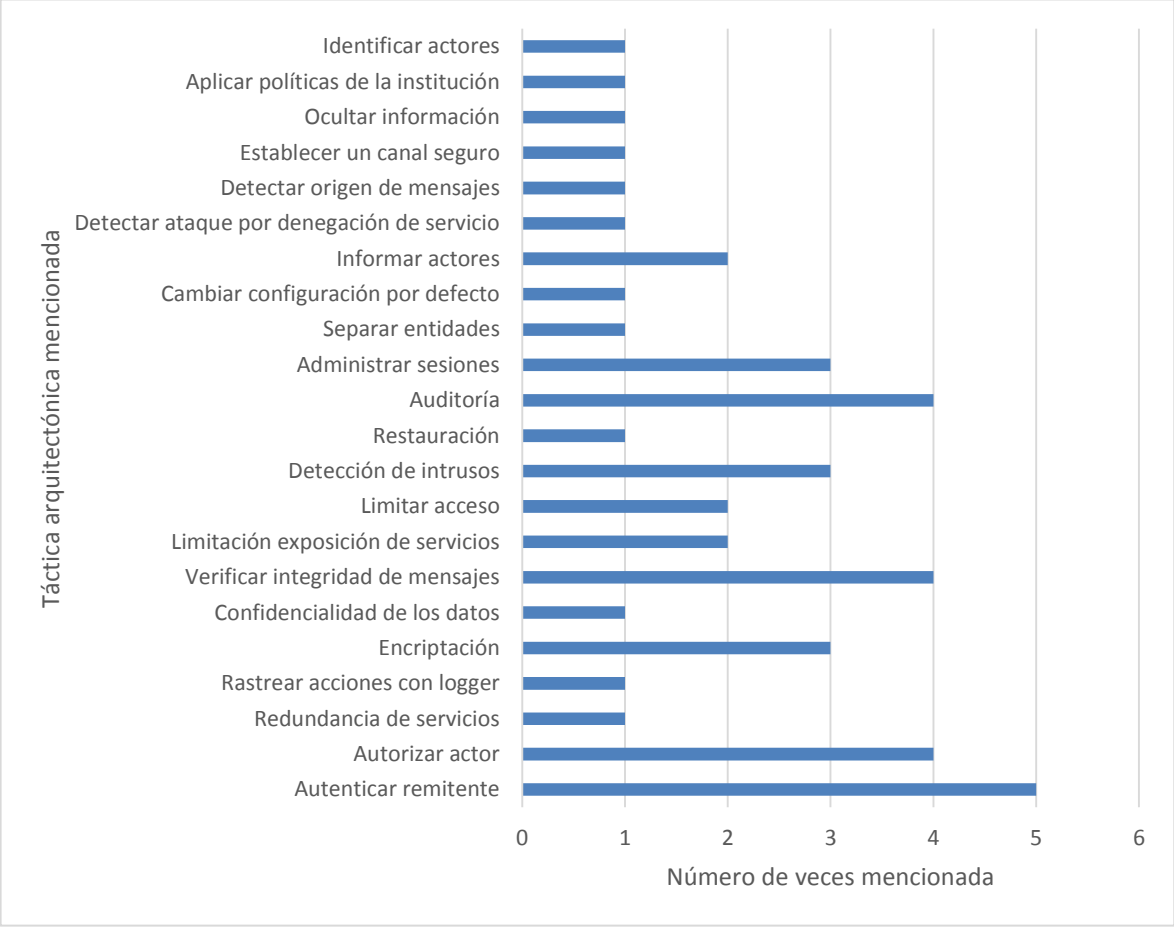
Fuente: Esta investigación - 2020.

- Tácticas arquitectónicas de seguridad

Se puede concluir que, las tácticas arquitectónicas mencionadas en los artículos seleccionados con mayor frecuencia son: autenticar remitente, autorizar actor,

verificar integridad de mensajes y la auditoria, como se puede apreciar en la **¡Error! No se encuentra el origen de la referencia.**.

Figura 4. Tácticas arquitectónicas de Seguridad mencionadas

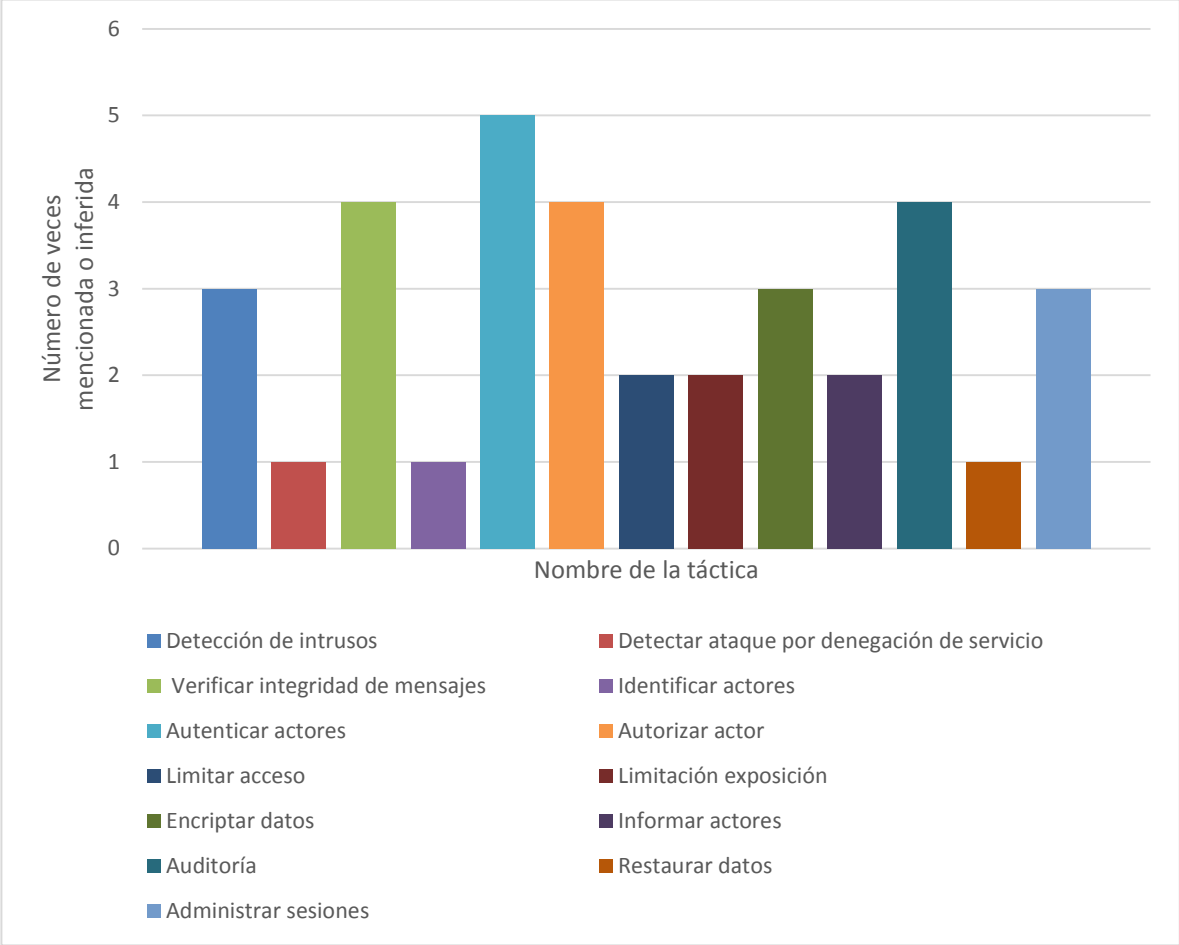


Fuente: Esta investigación - 2020.

- Tácticas arquitectónicas de seguridad para *e-voting*

Al analizar las anteriores tablas, se puede verificar en la Figura 5. Tácticas arquitectónicas de seguridad para *e-voting* mencionadas o inferidas de los artículos que la táctica arquitectónica de seguridad con mayor aparición en los artículos estudiados, es autenticar actores con 5 apariciones, luego, se tiene con 4 apariciones: verificar integridad de mensajes, autorizar actor y finalmente la auditoría.

Figura 5. Tácticas arquitectónicas de seguridad para *e-voting* mencionadas o inferidas de los artículos

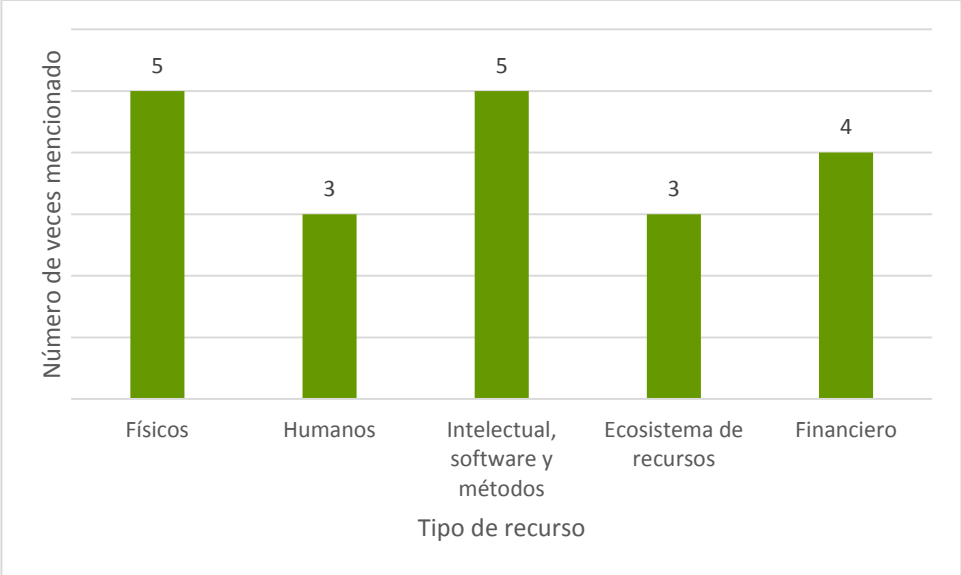


Fuente: Esta investigación - 2020.

- Recursos necesarios para la táctica de seguridad en *e-voting*

Al analizar la Figura 6. Recursos necesarios para la táctica de seguridad en *e-voting*, se puede verificar que los recursos físicos e intelectuales, software y métodos, son los más utilizados en este tipo de sistemas, los recursos financieros se nombran en cuatro oportunidades y finalmente con 3 apariciones, se tiene los recursos humanos y ecosistema de resultados.

Figura 6. Recursos necesarios para la táctica de seguridad en *e-voting*

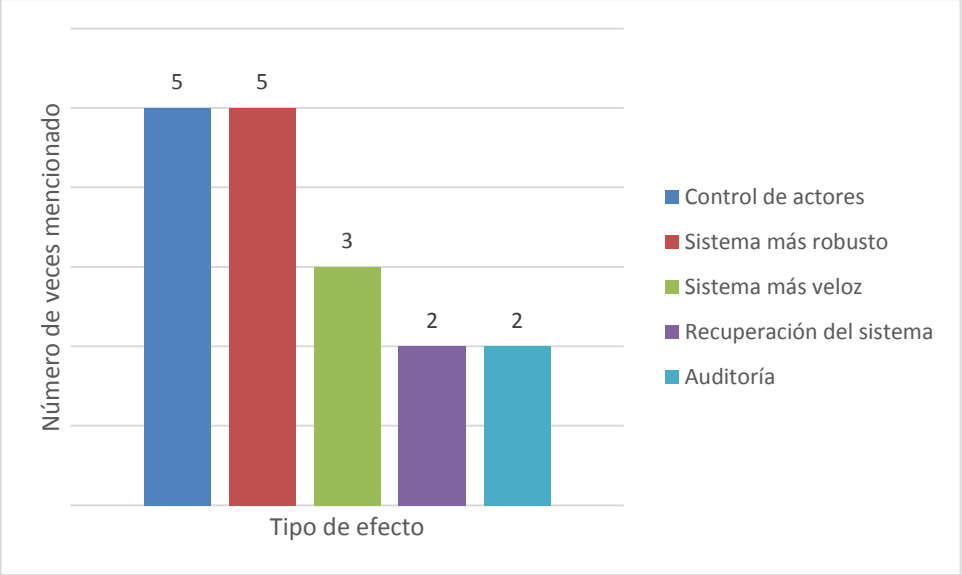


Fuente: Esta investigación - 2020.

- Efecto de la táctica de seguridad en *e-voting*

Como se puede ver en la Figura 7. Efecto de la táctica de seguridad en *e-voting*, se concluye que los efectos en la táctica de seguridad en *e-voting* son: principalmente el control de actores y el sistema más robusto, con 5 apariciones cada uno, seguido a ellos, se tiene que el sistema es más veloz, con 3 apariciones y finalmente, la recuperación del sistema y la auditoria con dos apariciones respectivamente.

Figura 7. Efecto de la táctica de seguridad en *e-voting*



Fuente: Esta investigación - 2020.

2.2 TÁCTICAS ARQUITECTÓNICAS DE SEGURIDAD PERTINENTES PARA E-VOTING

Según Bass y otros (2013), la seguridad es una medida de la capacidad del sistema para proteger los datos y la información del acceso no autorizado, pero sin dejar de proporcionar acceso a personas y sistemas autorizados. Una acción tomada contra el sistema informático con la intención de hacer daño, se llama ataque y puede adoptar varias formas. Eso puede ser un intento no autorizado de: acceder a datos o servicios, modificar datos; y puede tener la intención de negar servicios a usuarios legítimos （Bassy otros, 2013）.

2.2.1 Esquema general de las tácticas de seguridad para *e-voting*

Existen métodos que permiten pensar en cómo lograr un sistema que cumpla con los atributos de seguridad. Uno de ellos, es pensar en la seguridad física. Las

instalaciones seguras tienen acceso limitado (por ejemplo, mediante el uso de puntos de control de seguridad), tienen medios para detectar intrusos (por ejemplo, al exigir a los visitantes legítimos que usen insignias), tienen mecanismos de disuasión como guardias armados, tienen mecanismos de reacción como el bloqueo automático de puertas y tener mecanismos de recuperación como copia de seguridad fuera del sitio. Estos conducen a cuatro categorías de tácticas: detectar, resistir, reaccionar y recuperarse （Bassy otros，2013）. La Figura 8, muestra estas categorías como el objetivo de las tácticas de seguridad.

Figura 8. El Objetivo de las tácticas de seguridad

Fuente: Tomado de （Bassy otros，2013）.

Para Bass y otros (2013), **detectar ataques** consiste en 3 tácticas: detectar intrusos, detección de ataque de denegación de servicio y verificar integridad de mensajes. Igualmente, Bass y otros (2013) plantean que en la categoría para **resistir ataques** existen varios medios bien conocidos como: identificar, autenticar y autorizar actores, limitar el acceso, también la exposición, encriptar datos y administrar sesiones a usuarios. En la categoría de **reacción ante ataques** existen una táctica denominada informar actores, destinada a responder a un posible ataque （Bassy otros，2013）. Finalmente, Bass y otros (2013) plantean que en para la **recuperación de ataques,** una vez un sistema ha detectado e intentado resistir un ataque, necesita recuperarse. Parte de la recuperación es la restauración de los servicios y datos. Por ejemplo, servidores adicionales o conexiones de red pueden mantenerse en reserva para tal fin. Dado que un ataque exitoso puede considerarse un tipo de falla.

Además de las tácticas de disponibilidad que permiten la restauración de los servicios, se necesita mantener una pista de auditoría. Auditar, es decir, mantener

un registro de las acciones del usuario y del sistema, y sus efectos para ayudar a rastrear las acciones de un atacante e identificarlo （Bassy otros，2013）:

De acuerdo a lo anteriormente escrito, se plantea que una táctica corresponde a realizar auditoria y restaurar datos.

La adaptación del conjunto de tácticas de seguridad para sistemas *e-voting*, se muestra en la Figura 9.

Figura 9. Tácticas de Seguridad

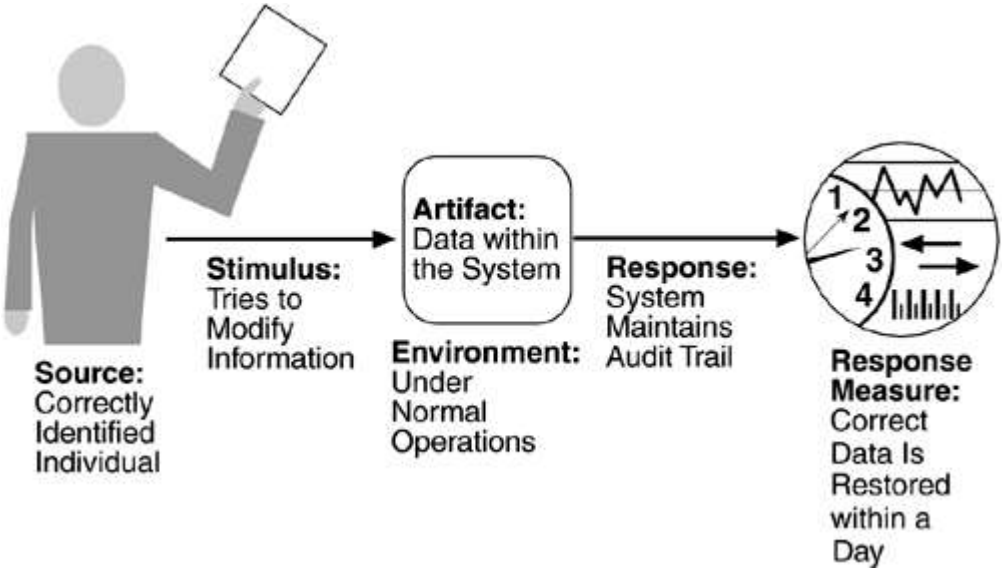
Fuente: Adaptación tomada del libro Software Architecture in Practice, （Bassy otros，2013）

2.2.2 Especificación de las tácticas

Para Bass y otros (2013), se debe plantear escenarios generales para los atributos de calidad de seguridad, utilizando una técnica denominada: modelado de amenazas y creando un árbol de ataque. La raíz del árbol es un ataque exitoso, los nodos son posibles causas directas de ese ataque y los nodos hijos descomponen las causas directas. Un ataque es un intento para romper la confidencialidad, integridad y disponibilidad (CIA) de un sistema de información, las hojas de los árboles de ataque serían el estímulo en el escenario.

Los escenarios de seguridad según Bass y otros (2013), como se observar en la Figura 3, constan de: fuente, estímulo, artefacto, ambiente, respuesta, y medida esperada de la respuesta. La *fuente* es una entidad (un humano, un sistema informático o cualquier otro actor) que generó el estímulo. El *estímulo* es una condición que requiere una respuesta cuando llega a un sistema. *El entorno corresponde con las* condiciones sobre las que opera el sistema, y puede estar en una condición de sobrecarga o en operación normal, o en algún otro estado relevante. Para muchos sistemas, la operación 'normal' puede referirse a uno de varios modos. Para este tipo de sistemas, el entorno debe especificar en qué modo se está ejecutando el sistema. *El artefacto* corresponde con una colección de sistemas, todo el sistema, o una parte o partes de él. *La respuesta* es la actividad realizada como resultado de la llegada del estímulo; y la *medida de respuesta es el grado o nivel esperado por el sistema cuando este en operación*. Cuando se produce la respuesta, debe ser medible de alguna manera para que el requisito pueda ser probado.

Figura 10. Representación gráfica de un escenario de calidad



Fuente: Tomado de （Bassy otros，2013）

Cada escenario describe algunos casos que se pueden presentar con el uso del sistema y cuál sería la respuesta esperada ante el caso.

Para establecer cada escenario, se propuso tener como base la siguiente tabla, que describe cada uno de los componentes del escenario:

Tabla 21. Componentes del escenario

Escenario	Descripción y numeración del escenario
Atributo de calidad	Atributo de calidad que se desea evaluar
Prioridad	Clasificación de prioridad en la que se encuentra el escenario
Fuente	Entidad que genera el estímulo
Estímulo	Condición que requiere el sistema
Artefacto	Artefacto del sistema que requiere el estímulo
Ambiente	Condición del entorno
Respuesta	Actividad realizada como resultado
Medida esperada de la respuesta	Medida de la actividad realizada

Fuente: Esta investigación - 2020.

2.1.3 Tácticas arquitectónicas de seguridad para *e-voting*

A continuación, se presentan las tácticas arquitectónicas de seguridad para *e-voting* de acuerdo con la categorización presentada en la figura 2.

2.1.3.1 Tácticas para detectar ataques

- Detectar intrusos

Detectar intrusión es la comparación del tráfico de red o los patrones de solicitud de servicio dentro de un sistema a un conjunto de firmas o patrones conocidos de comportamiento malicioso almacenados en una base de datos. Las firmas pueden basarse en protocolos, indicadores TCP, tamaños de carga útil, aplicaciones, fuente o dirección de destino o número de puerto （Bassy otros, 2013）. En la Tabla 22, se muestra un escenario para la táctica.

Tabla 22. Escenario para la táctica de detección de intrusos

Escenario	Escenario de Calidad # 01
Atributo de calidad	Seguridad

Prioridad	Alta
Fuente	*Cracker* desde ubicación remota.
Estímulo	Acceder a la información del sistema.
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema *IDS* implementado identifica la intrusión y la notifica.
Medida esperada de la respuesta	El sistema *IDS* identifica la intrusión, la almacena y notifica en un periodo menor a un minuto.

Fuente: Esta investigación - 2020.

- Detección de ataque de denegación de servicio

Detectar la denegación de servicio es la comparación del patrón o firma del tráfico de red que viene en un sistema de perfiles históricos de ataques conocidos de denegación de servicio （Bassy otros， 2013）. En la Tabla 23, se presenta un escenario para la táctica.

Tabla 23. Escenario para la táctica de detección de ataque de denegación de servicio

Escenario	Escenario de Calidad # 02
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	*Cracker* contratado por fuentes externas.
Estímulo	Dejar inaccesibles los servicios del sistema para impedir más votaciones.
Artefacto	Servicios del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema cuenta con validaciones captcha para verificar que el usuario sea una persona y no un robot *DDOS*

Medida esperada de la respuesta	Las validaciones captcha implementadas redireccionan al primer intento de ataques de robots *DDOS*

Fuente: Esta investigación - 2020.

- Verificar integridad de mensajes

Verificar integridad de mensajes emplea técnicas como sumas de comprobación o valores *Hash* para verificar la integridad de los mensajes, archivos de recursos, archivos de implementación y archivos de configuración. La suma de comprobación es un mecanismo de validación en el que el sistema mantiene información redundante para archivos y mensajes de configuración, y utiliza esta información redundante para verificar archivo de configuración o mensaje cuando se usa. Un valor *Hash* es una cadena única generada por una función *Hash* cuya entrada podría ser archivos de configuración o mensajes. Incluso un ligero cambio en los archivos o mensajes originales producen un cambio significativo en el valor *Hash* （ Bassy otros, 2013）. En la Tabla 24, se exhibe un escenario para la táctica.

Tabla 24. Escenario para la táctica de verificar integridad de mensajes

Escenario	Escenario de Calidad # 03
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	*Cracker* contratado por fuentes externas.
Estímulo	Modificar mensajes para realizar transacciones en el sistema de *e-voting*.
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	Al recibir el mensaje modificado el sistema detecta la alteración en el sello digital y rechazará el mensaje debido a que la suma de comprobación *Hash* no coincide.
Medida esperada de la respuesta	Al recibir el mensaje modificado el sistema detecta la alteración en el sello digital y de inmediato rechazará el mensaje.

Fuente: Esta investigación - 2020.

2.3.3.2 Tácticas para resistir ataques

- Identificar actores

Identificar "actores" se trata realmente de identificar la fuente de cualquier entrada externa al sistema. Los usuarios generalmente se identifican a través de ID de usuario. Otros sistemas pueden ser identificados a través de códigos de acceso, direcciones *IP*, protocolos, puertos, etc. （Bassy otros，2013）.En la Tabla 25, se puede observar un escenario para la táctica.

Tabla 25. Escenario para la táctica de identificar actores

Escenario	Escenario de Calidad # 04
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	*Cracker* desde ubicación remota.
Estímulo	Ingresar a la información y servicios del sistema.
Artefacto	Información y servicios del sistema *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema no permite el ingreso a usuarios no registrados desde la aplicación.
Medida esperada de la respuesta	El sistema no permite el ingreso a usuarios no registrados desde la aplicación de forma inmediata.

Fuente: Esta investigación - 2020.

- Autenticar actores

Autenticación de actores significa garantizar que un actor (un usuario o una computadora remota) es en realidad quién o qué pretende ser. Contraseñas, contraseñas de un solo uso, certificados digitales, y la identificación biométrica proporcionan un medio para la autenticación. （Bassy otros，2013）. En la Tabla 26, se muestra un escenario para la táctica.

Tabla 26. Escenario para la táctica de autenticar actores

Escenario	Escenario de Calidad # 05
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	Usuario votante del sistema *e-voting*.
Estímulo	Realizar una suplantación en el proceso de votación
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema no permite el ingreso para ejercer la votación a un usuario no registrado biométricamente.
Medida esperada de la respuesta	El sistema no permite el ingreso para ejercer la votación a un usuario no registrado biométricamente de forma inmediata.

Fuente: Esta investigación - 2020.

- Autorizar actores

Autorización de actores significa garantizar que un actor autenticado tenga los derechos para acceder y modificar datos o servicios. Este mecanismo generalmente se habilita al proporcionar Algunos mecanismos de control de acceso dentro de un sistema. El control de acceso puede ser por un actor o por una clase de actor. Las clases de actores pueden definirse por grupos de actores, por roles de actores o por listas de individuos （Bassy otros，2013）. En la Tabla 27, se exhibe un escenario para la táctica.

Tabla 27. Escenario para la táctica de autorizar actores

Escenario	Escenario de Calidad # 06
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	Funcionario del departamento de TI

Estímulo	Visualizar información confidencial del sistema.
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El uso del sistema permite a cada usuario acceder únicamente a la información y servicios que tiene autorizados en sus roles.
Medida esperada de la respuesta	El sistema informa de un intento de acceso no autorizado y se inhabilita el acceso al sistema para este usuario de forma inmediata.

Fuente: Esta investigación - 2020.

- Limitar acceso

Limitar el acceso a los recursos informáticos implica limitar el acceso a los recursos tales como memoria, conexiones de red o puntos de acceso. Esto se puede lograr usando memoria protección, bloqueo de un host, cierre de un puerto o rechazo de un protocolo. Por ejemplo, una zona desmilitarizada (*DMZ*) se usa cuando una organización quiere permitir que usuarios externos accedan a ciertos servicios y no acceder a otros servicios. Se encuentra entre Internet y un *firewall* frente a la red interna. El *firewall* es un único punto de acceso a la intranet (límite de exposición). También restringe acceso mediante una variedad de técnicas para autorizar a los usuarios (autorizar actores) （Bassy otros，2013）. En la Tabla 28, se puede observar un escenario para la táctica.

Tabla 28. Escenario para la táctica de limitar acceso

Escenario	Escenario de Calidad # 07
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	Candidato con permisos de acceso
Estímulo	Visualizar información confidencial del sistema.
Artefacto	Información del sistema de *e-voting*.

Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema usa los permisos del rol del candidato para limitar el acceso a la información confidencial del sistema *e-voting*.
Medida esperada de la respuesta	Se impide el acceso a la información confidencial automáticamente.

Fuente: Esta investigación - 2020.

- Limitar exposición

La táctica de límite de exposición minimiza la superficie de ataque del sistema. Esta táctica se enfoca en reducir la probabilidad y minimizar los efectos del daño causado por una acción hostil. Es una defensa pasiva porque no impide de manera proactiva que los atacantes hagan daño. El límite de exposición generalmente se logra al tener el menor número posible de puntos de acceso para recursos, datos o servicios y reduciendo la cantidad de conectores que pueden proporcionar exposición no anticipada （Bassy otros, 2013）. En la Tabla 29, se muestra un escenario para la táctica.

Tabla 29. Escenario para la táctica de limitar exposición

Escenario	Escenario de Calidad # 08
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	*Cracker* desde ubicación remota.
Estímulo	Visualizar información del sistema.
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	La exposición de los servidores del sistema se encuentra en una red interna, apartada de la red pública minimizando la superficie del ataque al sistema de información.
Medida esperada de la respuesta	La exposición de la información del sistema se reduce considerablemente.

Fuente: Esta investigación - 2020.

- Cifrar datos

Cifrar datos se refiere a que los datos deben estar protegidos contra el acceso no autorizado. La confidencialidad es usualmente lograda mediante la aplicación de alguna forma de cifrado a los datos y a la comunicación. El cifrado proporciona protección adicional a los datos mantenidos de forma persistente más allá de los disponibles en autorización. Los enlaces de comunicación, por otro lado, pueden no tener controles de autorización. En tales casos, el cifrado es la única protección para pasar datos a través de enlaces de acceso público. El enlace puede ser implementado por una red privada virtual (*VPN*) o por un *Secure Sockets Layer* (*SSL*) para un enlace basado en la web. El cifrado puede ser simétrico (ambas partes usan la misma clave) o asimétrica (claves públicas y privadas) （Bassy otros , 2013）. En la Tabla 30, se presenta un escenario para la táctica.

Tabla 30. Escenario para la táctica de encriptar datos

Escenario	Escenario de Calidad # 09
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	*Cracker* desde ubicación remota.
Estímulo	Visualizar información confidencial del sistema.
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema cuenta con un *SSL* que encripta las transacciones realizadas en el sistema de *e-voting*.
Medida esperada de la respuesta	El sistema cuenta con un *SSL* que encripta las transacciones realizadas en tiempo real en el sistema de *e-voting*.

Fuente: Esta investigación - 2020.

- Administración de sesiones

La administración de sesiones y una correcta implementación de esta táctica en una aplicación permitiría al sistema realizar un seguimiento de usuarios que están actualmente autenticados incluidos los permisos asignados y sus roles. （ Understanding Software Vulnerabilities Related to Architectural Security Tactics, 2017）. En la Tabla 31, se exhibe un escenario para la táctica.

Tabla 31. Escenario para la táctica de administración de sesiones

Escenario	Escenario de Calidad # 10
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	Empleado con acceso al sistema de *e-voting*
Estímulo	Acceder a servicios diferentes a su rol.
Artefacto	Servicios del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema no permite el ingreso a los servicios que el usuario no tenga asignado en sus roles.
Medida esperada de la respuesta	El sistema mantiene un registro de auditoria de cada ingreso a los módulos del sistema para cada usuario.

Fuente: Esta investigación - 2020.

2.3.3.3 Tácticas para reaccionar ante ataques

- Informar a los actores

Informar a los actores hace referencia a que los ataques en curso pueden requerir la acción de operadores, otro personal o sistemas cooperantes. Dicho personal o sistemas deben ser notificados cuando el sistema ha detectado un ataque. （Bassy otros, 2013）. En la Tabla 32, se puede observar un escenario para la táctica.

Tabla 32. Escenario para la táctica de informar actores

Escenario	Escenario de Calidad # 11
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	*Cracker* desde ubicación remota.
Estímulo	Acceder a los servicios del sistema.
Artefacto	Servicios del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	Se identifica el acceso no autorizado y se notifica dicho acceso.
Medida esperada de la respuesta	Se identifica el acceso no autorizado y se notifica dicho acceso de forma inmediata.

Fuente: Esta investigación – 2020.

2.3.3.4 Tácticas para recuperarse de ataques

- Auditoria

La auditoría quiere decir mantener un registro de las acciones del usuario, del sistema y sus efectos para ayudar a rastrear acciones y para identificar un atacante. Se puede analizar pistas de auditoría para intentar identificar a los atacantes, o para crear mejores defensas en el futuro. （Bassy otros, 2013）. En la Tabla 33, se muestra un escenario para la táctica.

Tabla 33. Escenario para la táctica de auditoria

Escenario	Escenario de Calidad # 12
Atributo de calidad	Seguridad
Prioridad	Alta

Fuente	Director del departamento de TI
Estímulo	Modificación de la información del sistema.
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema genera un reporte de auditoría de todas las transacciones realizadas.
Medida esperada de la respuesta	Se identifica a la fuente que realizó la modificación no autorizada en el registro de auditoria del sistema.

Fuente: Esta investigación - 2020.

- Restaurar datos

La restauración de los datos quiere decir que una vez que un sistema ha detectado e intentado resistir un ataque, necesita recuperarse. Parte de la recuperación es la restauración de datos y servicios. Por ejemplo, servidores adicionales o conexiones de red pueden configurarse para tal fin. （Bassy otros, 2013）. En la Tabla 34, se presenta un escenario para la táctica.

Tabla 34. Escenario para la táctica de restaurar datos

Escenario	Escenario de Calidad # 13
Atributo de calidad	Seguridad
Prioridad	Alta
Fuente	Director del departamento de TI
Estímulo	Eliminación de la información del sistema.
Artefacto	Información del sistema de *e-voting*.
Ambiente	Bajo operación normal del sistema.
Respuesta	El sistema cuenta con copias redundantes de los datos sensibles.

Medida esperada de la respuesta	Los datos sensibles se restauran correctamente sin alteraciones.

Fuente: Esta investigación - 2020.

2.3 CONSTRUCCIÓN DE UN PROTOTIPO *E-VOTING* QUE IMPLEMENTE TÁCTICAS ARQUITECTÓNICAS DE SEGURIDAD.

A continuación, se presenta el proceso de construcción del producto software Kybernan. Como parte de la arquitectura, se implementaron tácticas arquitectónicas de seguridad identificadas en la sección del documento 2.2 TÁCTICAS ARQUITECTÓNICAS DE SEGURIDAD PERTINENTES PARA E- VOTING

Para la gestión del desarrollo del producto software, se utilizaron de *Scrum* elementos que se describe de acuerdo con los artefactos, eventos, lineamientos y roles. Los artefactos utilizados fueron el *product backlog* y el *sprint backlog*. Los eventos correspondieron con el *sprint planning meeting*, *daily meeting*, *review* y las *retrospective*. En cuanto a los lineamientos, se utilizaron métricas para medir desempeño y velocidad del equipo. En cuanto a los roles, se emplearon el *producto owner*, *development team* y *scrum master*.

En total se realizaron seis (6) *sprints*, cada uno tuvo una duración de 15 días. A continuación, por cada sprint se presenta información en relación con: necesidades priorizadas (*sprint backlog*), gráfico de objetivos suaves para la definición de las decisiones de arquitectura, resultados del *review* e información relacionada con la medición de velocidad y desempeño.

2.3.1 *Sprint* 1

A continuación, se presentan los resultados obtenidos en el desarrollo del *sprint 1* (ver Anexo 1. sprint 1)

2.3.1.1 *Sprint backlog*

En la Tabla 35, se puede observar las cinco (5) historias de usuario priorizadas para el *sprint*, identificando la complejidad.

Tabla 35. Sprint backlog.

No.	Descripción de la Historia de Usuario	Prioridad	Complejidad
1	Como administrador quiero cargar los usuarios que pueden participar en la elección para que puedan votar	100	Media

2	Como administrador quiero agregar un candidato	100	Alta
3	Como administrador quiero eliminar un candidato	100	Baja
4	Como administrador quiero crear un proceso de elección	100	Muy alta
5	Como administrador quiero cancelar un proceso de elección no iniciado	100	Media

Fuente: Esta investigación - 2020.

2.3.1.2 Decisiones arquitectónicas de diseño

Para establecer las decisiones de diseño implementando tácticas arquitectónicas de seguridad, se utilizó como técnica el árbol de objetivos suaves (*SIG* por sus siglas en inglés *Softgoal Interdependency Graph*). El *SIG* se elabora informalmente, tiene objetivos y sub-objetivos de calidad representados como objetivos flexibles, y soluciones candidatas de diseño representadas como operacionalizaciones （Kobayashi y otros, 2016）. En la Figura 11, se observa las decisiones de diseño adoptadas para el primer *sprint*. Las tácticas arquitectónicas de seguridad para *e-voting* trabajadas en este *sprint* fueron el manejo de sesiones y la autorización de usuarios.

Figura 11. Árbol de objetivos suaves.

Fuente: Esta investigación - 2020.

Como se puede ver en la **¡Error! No se encuentra el origen de la referencia.** se especifica cuáles son las tecnologías que se usaron en la implementación del

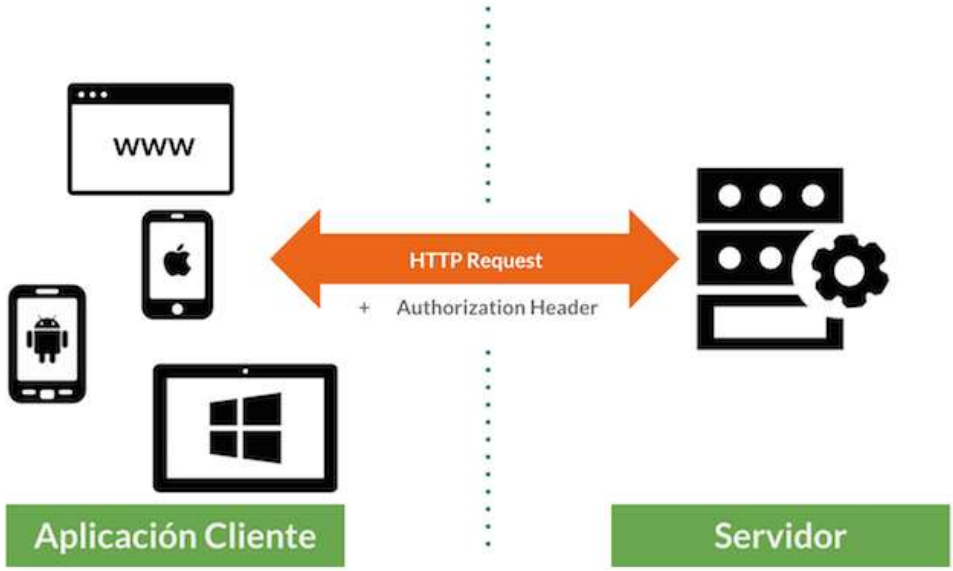
sistema con respecto a las tácticas correspondientes, a continuación, se describe brevemente cada una.

- *Token*

En palabras de Paszniuk (2016), *Json Web Token* es un conjunto de medios de seguridad para peticiones *HTTP* y así representar demandas para ser transferidos entre dos partes (cliente y servidor). Las partes de un *JWT* se codifican como un objeto *JSON* que está firmado digitalmente utilizando *JSON Web Signature*(*JWS*)（Paszniuk，2016）.

La representación gráfica de una petición *HTTP* se puede ver en la Figura 12.

Figura 12. Representación gráfica de una petición *HTTP*.



Fuente: Tomado de （Paszniuk，2016）

- *Cookies*

Una *cookie* es un archivo que se descarga en el dispositivo de cómputo al acceder a determinadas páginas web. Para la empresa RSL (2017), las *cookies* permiten a una página web, entre otras cosas, almacenar y recuperar información sobre los hábitos de navegación de un usuario o de su equipo y, dependiendo de la información que contengan y de la forma en que utilice su equipo, pueden utilizarse para reconocer al usuario. Las *cookies* se asocian únicamente a un usuario anónimo y su ordenador o dispositivo y no proporcionan referencias que permitan conocer sus datos personales （RSL，2017）.

- Roles y permisos tipo *grant*

Para la empresa Moodle (2020), un rol es una colección de permisos definida para todo el sistema que se puede asignar a usuarios específicos en contextos específicos. La combinación de roles y contexto definen la habilidad de un usuario específico para hacer algo en alguna página, los ejemplos más comunes son los roles de estudiante y maestro en el contexto de un curso （Moodle，2020）.

2.3.1.3 *Review*

El objetivo del *sprint* fue elaborar las cinco (5) historias de usuario seleccionadas y registradas en el *sprint backlog* y del requerimiento arquitectónico de seguridad incorporar el manejo de sesiones y autorizar actores.

En el *sprint* se logró realizar dos (2) historias de usuario del *sprint backlog* y se incorpora el manejo de sesiones y autorización de actores.

Los principales impedimentos encontrados fueron dar respuesta al manejo de sesiones y autorización de actores se incorpora dentro del *sprint backlog* como actividades, pero no se estiman, acción que permitió avanzar en dos (2) de las cinco (5) historias de usuarios planeadas.

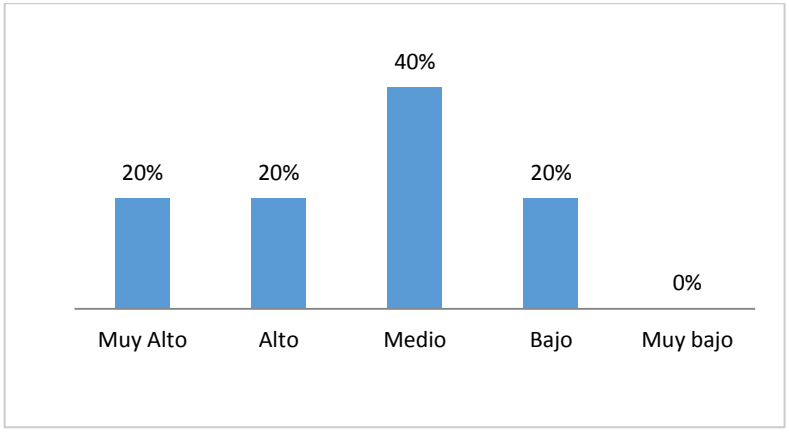
Para el siguiente *sprint* se trasladaron las historias de usuario para agregar candidatos, eliminar un candidato y cancelar un proceso de votación. Por lo tanto, estas historias se priorizan para realizar en el siguiente *sprint*.

2.3.1.4 Medición

A continuación, se presenta la medición de productividad en el proceso de software. Está medición se realizó a través de los indicadores: complejidad de las historias de usuario, tiempo de trabajo por etapa, tiempo de trabajo por semana y desempeño.

Como se puede evidenciar en la Figura 13. Nivel de complejidad de las historias de usuario de usua, la complejidad de historias de usuario con mayor frecuencia fue el nivel medio, con un 40%.

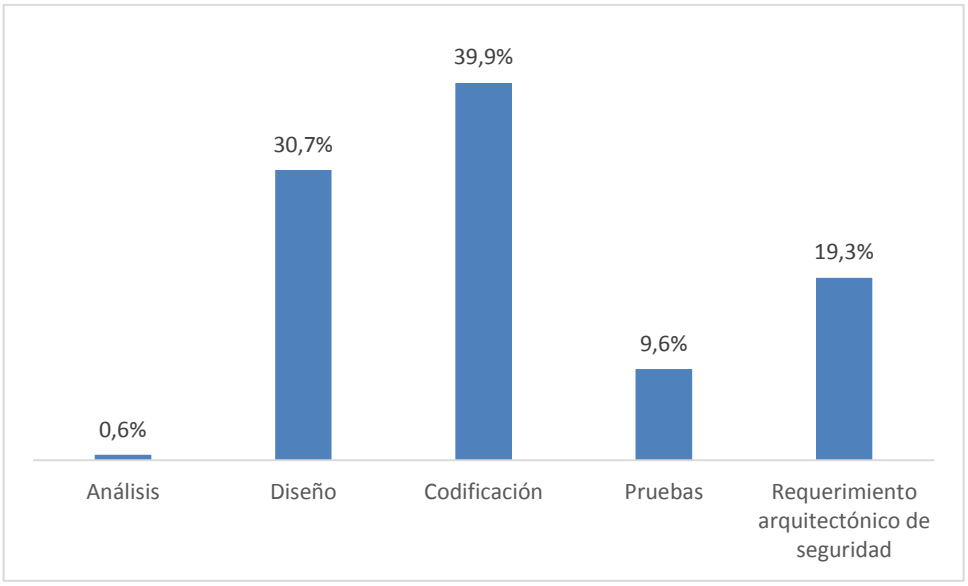
Figura 13. Nivel de complejidad de las historias de usuario de usuario.



Fuente: Esta investigación.

El tiempo estimado de trabajo para el *sprint* fue de 56 horas, pero el tiempo realmente trabajado fue de 57:05 horas. La etapa que más porcentaje de tiempo requirió fue el diseño, incluyendo las decisiones de arquitectura con un 50%, como se puede observar en la Figura 14.

Figura 14. Porcentaje de tiempo de trabajo por etapa.



Fuente: Esta investigación - 2020.

Teniendo en cuenta que se realizó una planeación para trabajar 28 horas a la semana, en la Tabla 36, se puede mirar que en la semana uno se trabajó un 19%

más del tiempo presupuestado. Sin embargo, también se puede observar que en la semana dos, el desarrollo tomó un 15% menos del tiempo que se planeó, y así, hubo una compensación con respecto al tiempo para el *sprint* 1.

Tabla 36. Tiempo de trabajo por semana.

Semanas	Tiempo trabajado	Tiempo no trabajado	% Tiempo trabajado	% Tiempo No trabajado
Semana 1	33:20:00	0:00:00	119%	0%
Semana 2	23:45:00	4:15:00	85%	15%
Total	57:05:00	4:15:00		

Fuente: Esta investigación - 2020.

Finalmente, en la Tabla 37Tabla 37. Resumen del desempeño., se presenta un resumen del desempeño para el *sprint* 1, donde se observa que se trabajó un 2% más, hubo un desperdicio de 8% con respecto al tiempo total del tiempo trabajo para el ciclo, y como se mencionó anteriormente, el desperdicio está compensado en la semana uno, donde se trabajó más del tiempo planeado.

Tabla 37. Resumen del desempeño.

Tiempo total de trabajo planeado para el ciclo	56:00:00
Tiempo total trabajado	57:05:00
Tiempo de desperdicio total	4:15:00
% de tiempo trabajado	102%
% de desperdicio total	8%

Fuente: Esta investigación - 2020.

2.3.2 *Sprint* 2

A continuación, se presentan los resultados obtenidos en el desarrollo del *sprint 2.* (ver Anexo 2. sprint 2).

2.3.2.1 *Sprint backlog*

En la Tabla 38, se puede observar las tres (3) historias de usuario priorizadas para el *sprint*, identificando la complejidad.

Tabla 38. Sprint backlog.

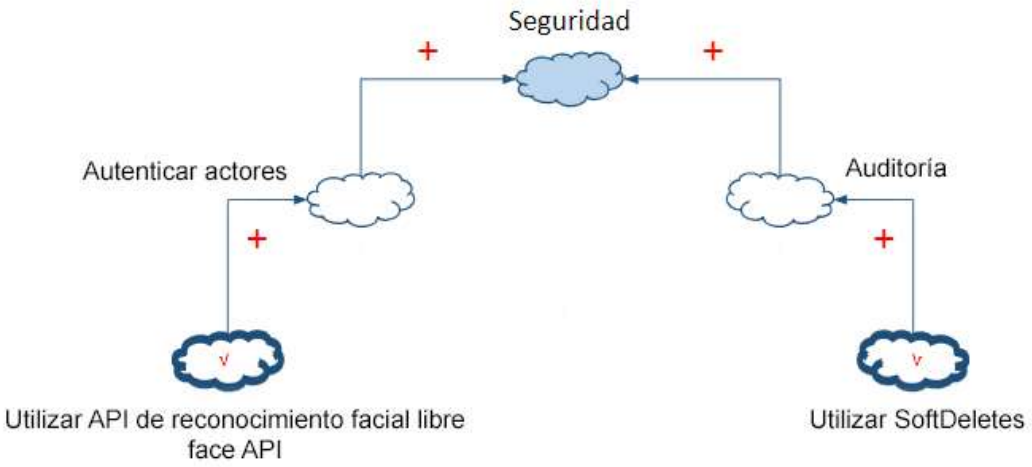
No.	Descripción de la Historia de Usuario	Prioridad	Complejidad
2	Como administrador quiero agregar un candidato	100	Alta
3	Como administrador quiero eliminar un candidato	100	Baja
5	Como administrador quiero cancelar un proceso de elección no iniciado	100	Media

Fuente: Esta investigación - 2020.

2.3.2.2. Decisiones arquitectónicas de diseño

En la Figura 15, se observa las decisiones de diseño adoptadas para el segundo *sprint*. Las tácticas arquitectónicas de seguridad para *e-voting* trabajadas en este *sprint* fueron la autenticación de actores y la auditoría.

Figura 15. Árbol de objetivos suaves.



Fuente: Esta investigación - 2020.

Como se puede ver en la Figura 15**¡Error! No se encuentra el origen de la referencia.** se especifica cuáles son las tecnologías que se usaron en la implementación del sistema con respecto a las tácticas correspondientes, a continuación, se describe brevemente cada una.

- *Face API*

En palabras de Vincent Mühler, (2018) *face-api.js* es un *API* de *JavaScript* para detección de rostros y reconocimiento de rostros en el navegador implementada sobre la *API* central de *tensorflow.js* （Mühler，2018）.

- *SoftDeletes*

Según Morales (2019), es importante no eliminar los datos de la base de datos, el método *SoftDeletes* quiere decir que al registro se le cambia su estado, pero no se lo elimina, al cambiar su estado, se elimina de forma lógica （Morales，2019）.

En la Figura 16, se puede observar el árbol de objetivos suaves completo con las decisiones de diseño adoptadas hasta el *sprint.*

Figura 16. Árbol general de objetivos suaves

Fuente: Esta investigación - 2020.

2.3.2.3. *Review*

El objetivo del *sprint* fue elaborar las tres (3) historias de usuario seleccionadas y registradas en el *sprint backlog* y del requerimiento arquitectónico de seguridad incorporar la autenticación de actores y la auditoría.

En el *sprint* se logró realizar tres (3) historias de usuario del *sprint backlog* y se incorpora la autenticación de actores y la auditoría.

Los principales impedimentos encontrados fueron implementar la autenticación de actores, debido al uso de una versión no compatible de *face API*, acción que permitió avanzar en las tres (3) historias de usuarios planeadas.

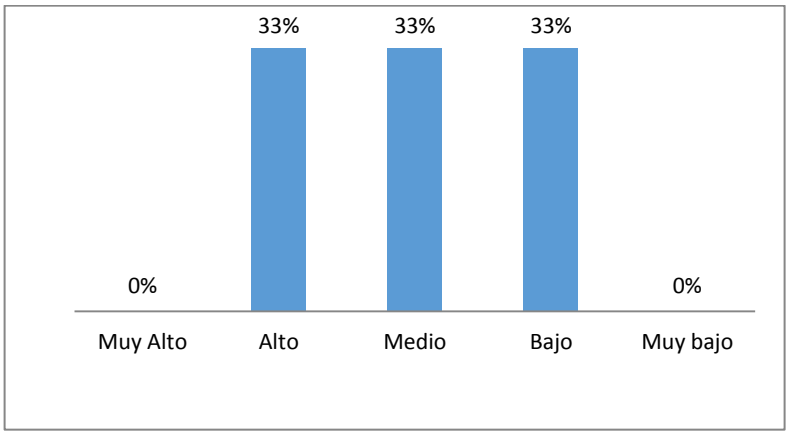
Para el siguiente *sprint* se plantearon las historias para que un usuario votante registre un voto, consulte si es apto para votar, reporte una novedad y que un administrador descargue un reporte de los resultados de un proceso de votación.

2.3.2.4. Medición

A continuación, se presenta la medición de productividad en el proceso de software. Está medición se realizó a través de los indicadores: complejidad de las historias de usuario, tiempo de trabajo por etapa, tiempo de trabajo por semana y desempeño.

Como se puede evidenciar en la Figura 17, las complejidades de historias de usuario con mayor frecuencia fueron el nivel medio, alto y bajo, con un 33% cada una.

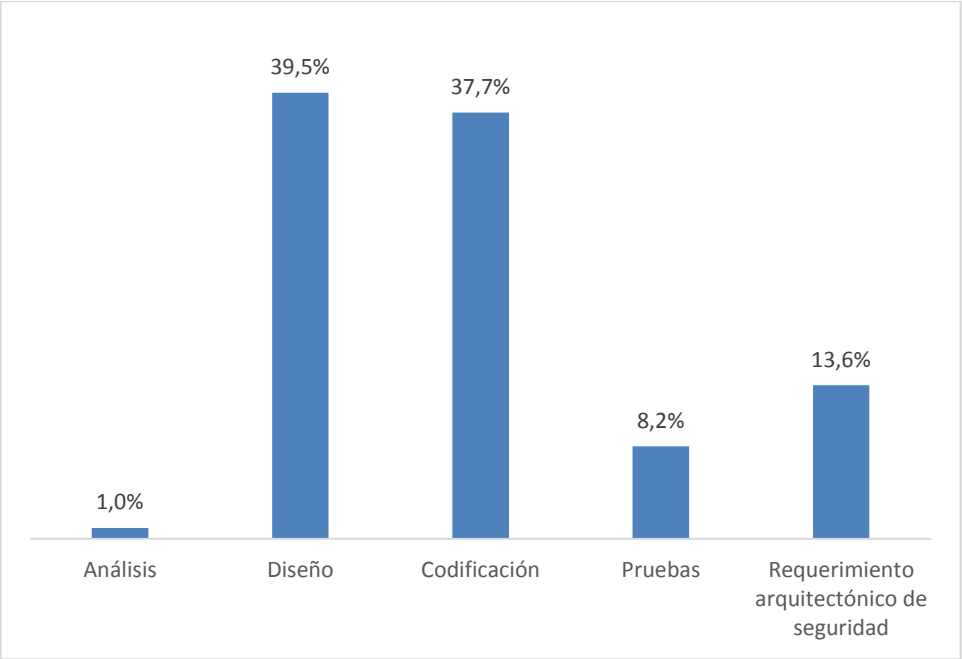
Figura 17. Nivel de complejidad de las historias de usuario de usuario.



Fuente: Esta investigación.

El tiempo estimado de trabajo para el *sprint* fue de 56 horas, pero el tiempo realmente trabajado fue de 51:28 horas. La etapa que más porcentaje de tiempo requirió fue el diseño, incluyendo las decisiones de arquitectura con un 53,1%, como se puede observar en la Figura 18Figura 14.

Figura 18. Porcentaje de tiempo de trabajo por etapa.



Fuente: Esta investigación - 2020.

Teniendo en cuenta que se realizó una planeación para trabajar 28 horas a la semana, en la Tabla 39. Tiempo de trabajo por semana.Tabla 36. Tiempo de trabajo por semana., se puede mirar que en la semana uno se trabajó un 5% menos del tiempo presupuestado. También se puede observar que en la semana dos, el desarrollo tomó un 11% menos del tiempo que se planeó.

Tabla 39. Tiempo de trabajo por semana.

Semanas	Tiempo trabajado	Tiempo no trabajado	% Tiempo trabajado	% Tiempo No trabajado
Semana 1	26:30:00	1:30:00	95%	5%
Semana 2	24:58:00	3:02:00	89%	11%
Total	51:28:00	4:32:00		

Fuente: Esta investigación - 2020.

Finalmente, en la Tabla 40, se presenta un resumen del desempeño para el *sprint* 2, donde se observa que se trabajó un 92%, hubo un desperdicio de 8% con respecto al tiempo total del tiempo trabajo para el ciclo.

Tabla 40. Resumen del desempeño.

Tiempo total de trabajo planeado para el ciclo	56:00:00
Tiempo total trabajado	51:28:00
Tiempo de desperdicio total	4:32:00
% de tiempo trabajado	92%
% de desperdicio total	8%

Fuente: Esta investigación - 2020.

2.3.3 *Sprint* 3

A continuación, se presentan los resultados obtenidos en el desarrollo del *sprint 3.* (ver Anexo 3. sprint 3).

2.3.3.1 *Sprint backlog*

En la Tabla 41, se puede observar las cuatro (4) historias de usuario priorizadas para el *sprint*, identificando la complejidad.

Tabla 41. Sprint backlog.

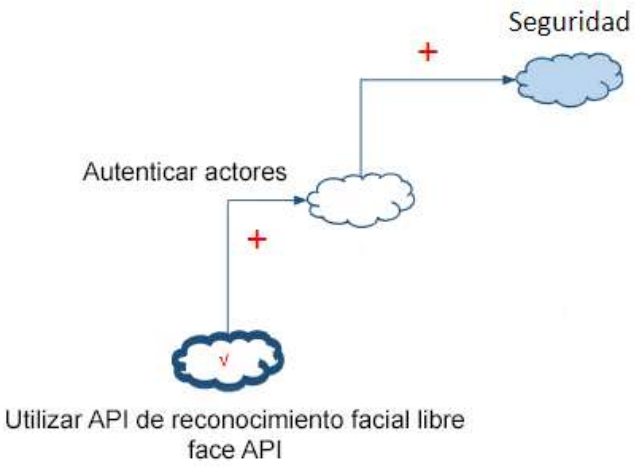
No.	Descripción de la Historia de Usuario	Prioridad	Complejidad
7	Como votante quiero registrar mi voto para elegir a mi representante	100	Muy alta
9	Como votante quiero consultar si soy apto para votar	100	Media
11	Como votante quiero reportar una novedad para poder votar	100	Media
17	como administrador quiero descargar un reporte de los resultados del proceso de elección terminado	90	Media

Fuente: Esta investigación - 2020.

2.3.3.2. Decisiones arquitectónicas de diseño

En la Figura 19, se observa la decisión de diseño que se continúa para el tercer *sprint*. La táctica arquitectónica de seguridad para *e-voting* trabajada en este *sprint* fue la autenticación de actores.

Figura 19. Árbol de objetivos suaves.



Fuente: Esta investigación - 2020.

El árbol de objetivos suaves completo con las decisiones de diseño adoptadas hasta el *sprint* se mantiene sin cambios, como se muestra en la Figura 16.

2.3.3.3. *Review*

El objetivo del *sprint* fue elaborar las cuatro (4) historias de usuario seleccionadas y registradas en el *sprint backlog* y del requerimiento arquitectónico de seguridad incorporar la autenticación de actores.

En el *sprint* se logró realizar cuatro (4) historias de usuario del *sprint backlog* y se incorporó la autenticación de actores.

Los principales impedimentos encontrados fueron implementar la autenticación de actores, debido al uso de una versión no compatible de *face API*, acción que permitió avanzar en las cuatro (4) historias de usuarios planeadas.

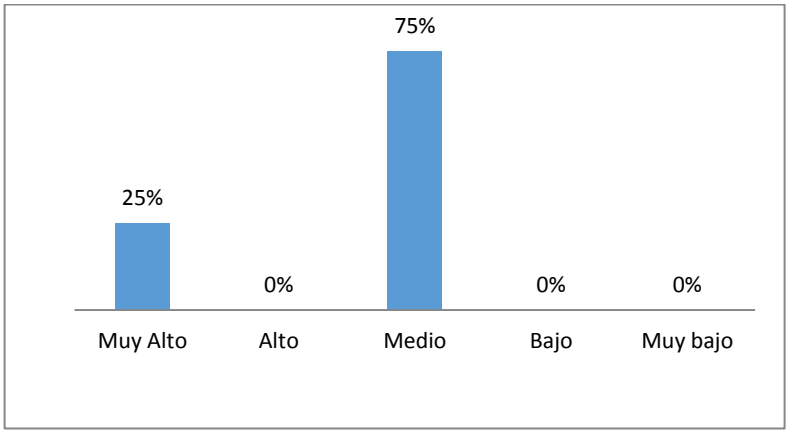
Para el siguiente *sprint* se plantearon las historias para que un usuario administrador inicie y finalice automáticamente un proceso de votación, modifique la fecha, hora de inicio y fin, el nombre y descripción de un proceso de votación, también para que un usuario votante reciba su certificado de voto y consulte los resultados de los escrutinios.

2.3.3.4. Medición

A continuación, se presenta la medición de productividad en el proceso de software. Está medición se realizó a través de los indicadores: complejidad de las historias de usuario, tiempo de trabajo por etapa, tiempo de trabajo por semana y desempeño.

Como se puede evidenciar en la Figura 20, la complejidad de historias de usuario con mayor frecuencia fue la media con un 75%.

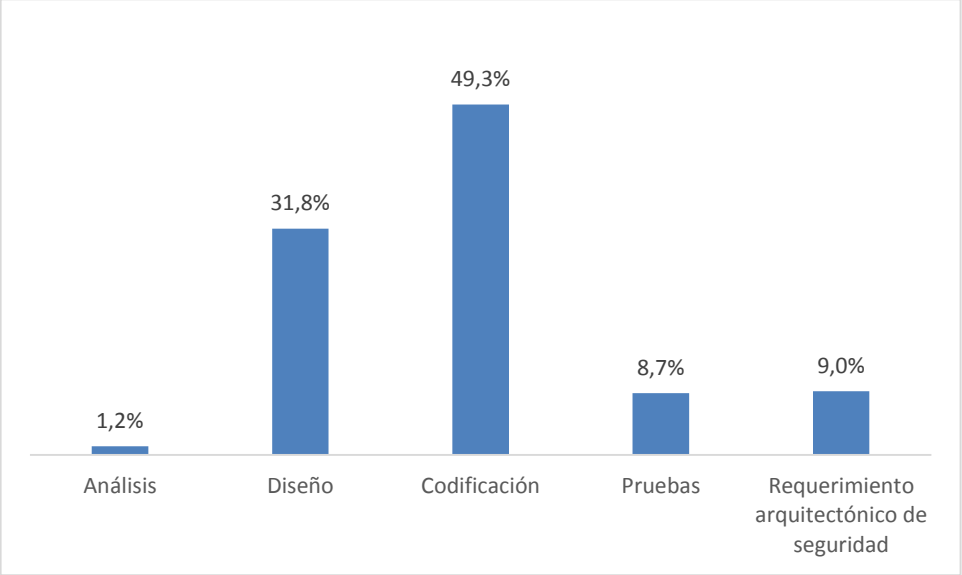
Figura 20. Nivel de complejidad de las historias de usuario de usuario.



Fuente: Esta investigación.

El tiempo estimado de trabajo para el *sprint* fue de 56 horas, pero el tiempo realmente trabajado fue de 55:45 horas. La etapa que más porcentaje de tiempo requirió fue la codificación con un 49,3%, como se puede observar en la Figura 21Figura 14.

Figura 21. Porcentaje de tiempo de trabajo por etapa.



Fuente: Esta investigación - 2020.

Teniendo en cuenta que se realizó una planeación para trabajar 28 horas a la semana, en la Tabla 42Tabla 36. Tiempo de trabajo por semana., se puede mirar que en la semana uno se trabajó un 7% menos del tiempo presupuestado. También se puede observar que en la semana dos, el desarrollo tomó un 6% más del tiempo que se planeó, por esta razón, hubo una compensación con respecto a la semana uno.

Tabla 42. Tiempo de trabajo por semana.

Semanas	Tiempo trabajado	Tiempo no trabajado	% Tiempo trabajado	% Tiempo No trabajado
Semana 1	26:10:00	1:50:00	93%	7%
Semana 2	29:35:00	0:00:00	106%	0%
Total	55:45:00	1:50:00		

Fuente: Esta investigación - 2020.

Finalmente, en la Tabla 43, se presenta un resumen del desempeño para el *sprint* 3, donde se observa que se trabajó un 100%, hubo un desperdicio de 3% con respecto al tiempo total del tiempo trabajo para el ciclo, sin embargo, como se

mencionó anteriormente, hubo una compensación en la semana dos con respecto a la primera.

Tabla 43. Resumen del desempeño.

Tiempo total de trabajo planeado para el ciclo	56:00:00
Tiempo total trabajado	55:45:00
Tiempo de desperdicio total	1:50:00
% de tiempo trabajado	100%
% de desperdicio total	3%

Fuente: Esta investigación - 2020.

2.3.4 *Sprint* 4

A continuación, se presentan los resultados obtenidos en el desarrollo del *sprint 4.* (*v*er Anexo 4. sprint 4).

2.3.4.1. *Sprint backlog*

En la Tabla 44, se puede observar las ocho (8) historias de usuario priorizadas para el *sprint*, identificando la complejidad.

Tabla 44. Sprint backlog.

No.	Descripción de la Historia de Usuario	Prioridad	Complejidad
11	Como administrador quiero que se inicie y finalice automáticamente un proceso de votación	90	Media
12	Como votante quiero consultar los resultados de los escrutinios para conocer quién me representa	81	Media
13	Como votante quiero recibir mi certificado de votación	81	Media
14	Como administrador quiero modificar la fecha y hora de inicio y fin de un proceso de elección no iniciado	81	Media
15	Como administrador quiero modificar el nombre y descripción de un proceso de elección no iniciado	81	Baja
21	Como administrador quiero suspender un proceso de votación	80	Media

No.	Descripción de la Historia de Usuario	Prioridad	Complejidad
24	Como usuario quiero consultar los candidatos registrados y sus propuestas	36	Media
25	Como administrador quiero ingresar al sistema	20	Baja

Fuente: Esta investigación - 2020.

2.3.4.2. Decisiones arquitectónicas de diseño

Se continuó con la decisión de diseño adoptada para el tercer *sprint*. La táctica arquitectónica de seguridad para *e-voting* trabajada en este *sprint* fue la autenticación de actores como se puede ver en la Figura 19.

2.3.4.3. Review

El objetivo del *sprint* fue elaborar las ocho (8) historias de usuario seleccionadas y registradas en el *sprint backlog* y del requerimiento arquitectónico de seguridad incorporar la autenticación de actores.

En el *sprint* se logró realizar ocho (8) historias de usuario del *sprint backlog* y se incorporó la autenticación de actores.

En este *sprint* no se presentaron impedimentos y se avanzó en las ocho (8) historias de usuarios planeadas.

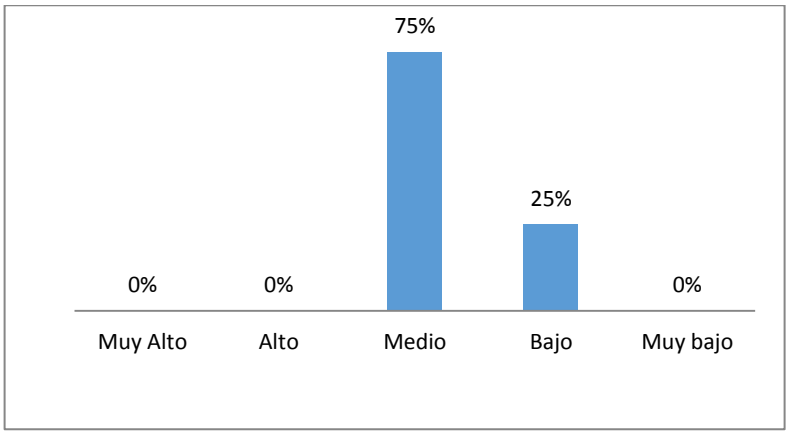
Para el siguiente *sprint* se plantearon las historias para que un usuario administrador pueda ver verificar las novedades registradas por los usuarios votantes pueda responder sus solicitudes, conozca los estados de reconocimiento facial en un proceso de votación, descargue un reporte de los candidatos que se inscribieron a un proceso de votación, y para que un usuario votante pueda verificar el estado de una solicitud reportada de una novedad y verifique si su certificado de votación es válido.

2.3.4.4. Medición

A continuación, se presenta la medición de productividad en el proceso de software. Esta medición se realizó a través de los indicadores: complejidad de las historias de usuario, tiempo de trabajo por etapa, tiempo de trabajo por semana y desempeño.

Como se puede evidenciar en la Figura 22, la complejidad de historias de usuario con mayor frecuencia fue la media con un 75%.

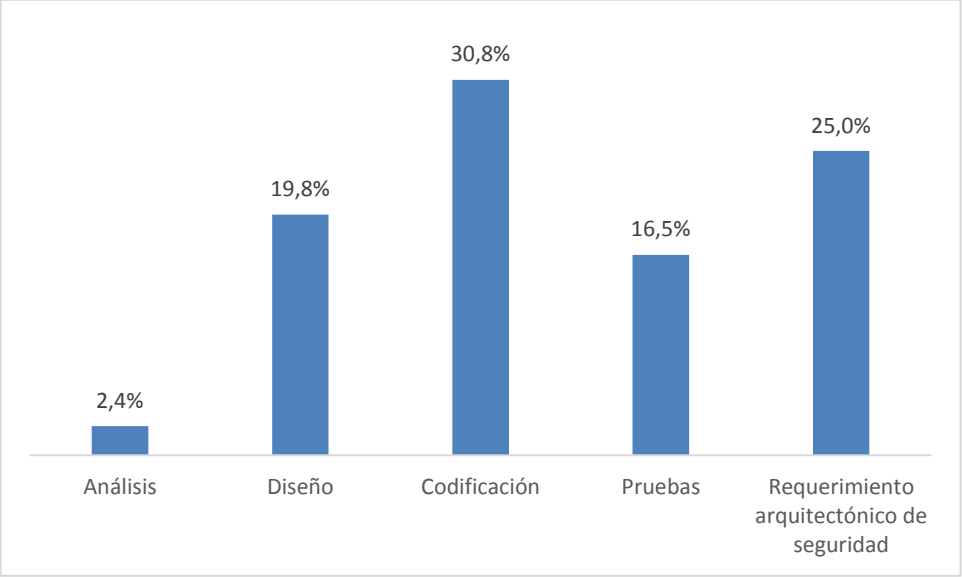
Figura 22. Nivel de complejidad de las historias de usuario de usuario.



Fuente: Esta investigación.

El tiempo estimado de trabajo para el *sprint* fue de 56 horas, y así mismo, el tiempo trabajado fue de 56 horas. La etapa que más porcentaje de tiempo requirió fue la de diseño, incluyendo el requerimiento arquitectónico de seguridad con un 44,8%, como se puede observar en la Figura 23.

Figura 23. Porcentaje de tiempo de trabajo por etapa.



Fuente: Esta investigación - 2020.

Teniendo en cuenta que se realizó una planeación para trabajar 28 horas a la semana, en la Tabla 45, se puede mirar que en la semana uno y dos se trabajó un 100% del tiempo presupuestado.

Tabla 45. Tiempo de trabajo por semana.

Semanas	Tiempo trabajado	Tiempo no trabajado	% Tiempo trabajado	% Tiempo No trabajado
Semana 1	28:00:00	0:00:00	100%	0%
Semana 2	28:00:00	0:00:00	100%	0%
Total	56:00:00	0:00:00		

Fuente: Esta investigación - 2020.

Finalmente, en la Tabla 46, se presenta un resumen del desempeño para el *sprint* 4, donde se observa que se trabajó un 100% y hubo un desperdicio de 0% con respecto al tiempo total del tiempo trabajado para el ciclo.

Tabla 46. Resumen del desempeño.

Tiempo total de trabajo planeado para el ciclo	56:00:00
Tiempo total trabajado	55:45:00
Tiempo de desperdicio total	1:50:00
% de tiempo trabajado	100%
% de desperdicio total	0%

Fuente: Esta investigación - 2020.

2.3.5. *Sprint* 5

A continuación, se presentan los resultados obtenidos en el desarrollo del *sprint 5.* (ver Anexo 5. sprint 5)

2.3.5.1. *Sprint backlog*

En la Tabla 47, se puede observar las cinco (5) historias de usuario priorizadas para el *sprint*, identificando la complejidad.

Tabla 47. Sprint backlog.

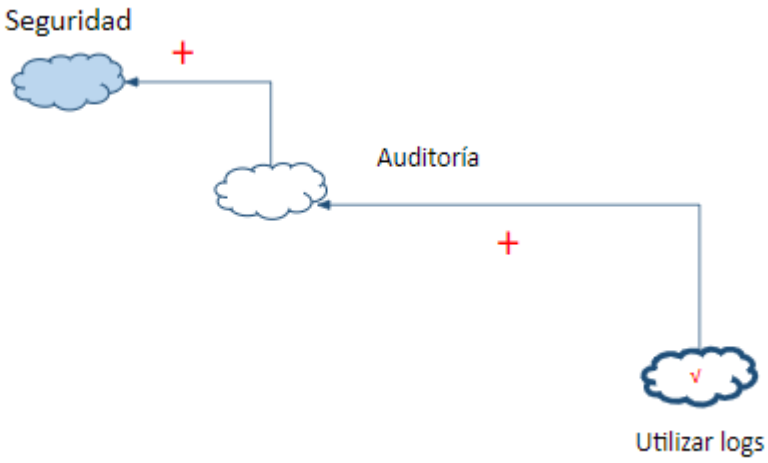
No.	Descripción de la Historia de Usuario	Prioridad	Complejidad
9	Como funcionario encargado de verificar las novedades registradas por los usuarios votantes quiero responder a sus solicitudes	100	Media
12	Como votante quiero verificar el estado de una solicitud reportada de una novedad para poder votar	81	Media
17	Como administrador quiero conocer los estados de reconocimiento facial en un proceso de votación	81	Media
18	Como votante quiero consultar si mi certificado de votación es valido	81	Media
19	Como administrador quiero descargar un reporte de los candidatos que se inscribieron a un proceso de votación	80	Media

Fuente: Esta investigación - 2020.

2.3.5.2. Decisiones arquitectónicas de diseño

En la Figura 24, se observa la decisión de diseño adoptada para el quinto *sprint*. La táctica arquitectónica de seguridad para *e-voting* trabajada en este *sprint* fue complementar la auditoria con el uso de *logs*.

Figura 24. Árbol de objetivos suaves.



Fuente: Esta investigación - 2020.

Como se puede ver en la Figura 24,**¡Error! No se encuentra el origen de la referencia.** se especifica cuáles son las tecnologías que se usaron en la implementación del sistema con respecto a las tácticas correspondientes, a continuación, se describe brevemente.

- ***Log***

Para la Superintendencia de sociedades (2017), un *Log* es el registro de las acciones y de los acontecimientos que ocurren en un sistema computacional cuando un usuario o proceso está activo y sucede un evento que está configurado para reportar （Superintendencia de sociedades，2017）.

En la Figura 25, se puede observar el árbol de objetivos suaves completo con las decisiones de diseño adoptadas hasta el *sprint.*

Figura 25. Árbol general de objetivos suaves

Fuente: Esta investigación - 2020

2.3.5.3. Review

El objetivo del *sprint* fue elaborar las cinco (5) historias de usuario seleccionadas y registradas en el *sprint backlog* y del requerimiento arquitectónico de seguridad de auditoria.

En el *sprint* se logró realizar cinco (5) historias de usuario del *sprint backlog* y se complementa la táctica de auditoria con el uso de *logs.*

En este *sprint* no se presentaron impedimentos y se avanzó en las cinco (5) historias de usuarios planeadas.

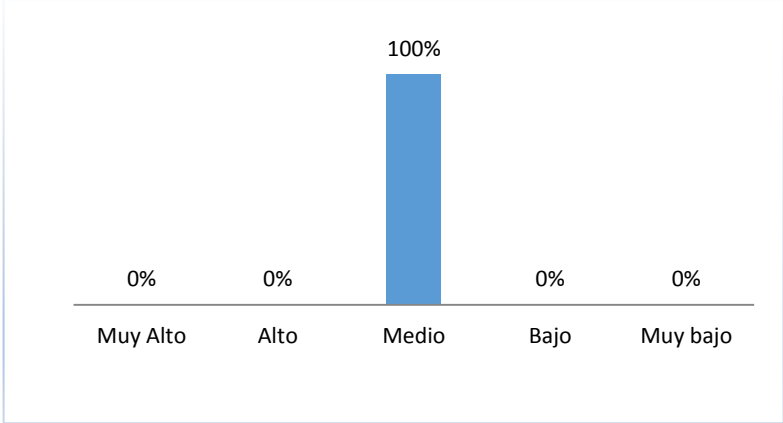
Para el siguiente *sprint* se plantearon las historias para que un usuario administrador descargue un reporte de los votantes de un proceso de elección, actualice los datos de los candidatos en caso de ser erróneos, envíe un *email* invitando a votar a las personas activas para el proceso de elección y registre otro usuario administrador y para que un usuario candidato renuncie a la candidatura antes de que el proceso de elección comience.

2.3.5.4. Medición

A continuación, se presenta la medición de productividad en el proceso de software. Está medición se realizó a través de los indicadores: complejidad de las historias de usuario, tiempo de trabajo por etapa, tiempo de trabajo por semana y desempeño.

Como se puede evidenciar en la Figura 26, la complejidad de historias de usuario con el total de la frecuencia fue la media.

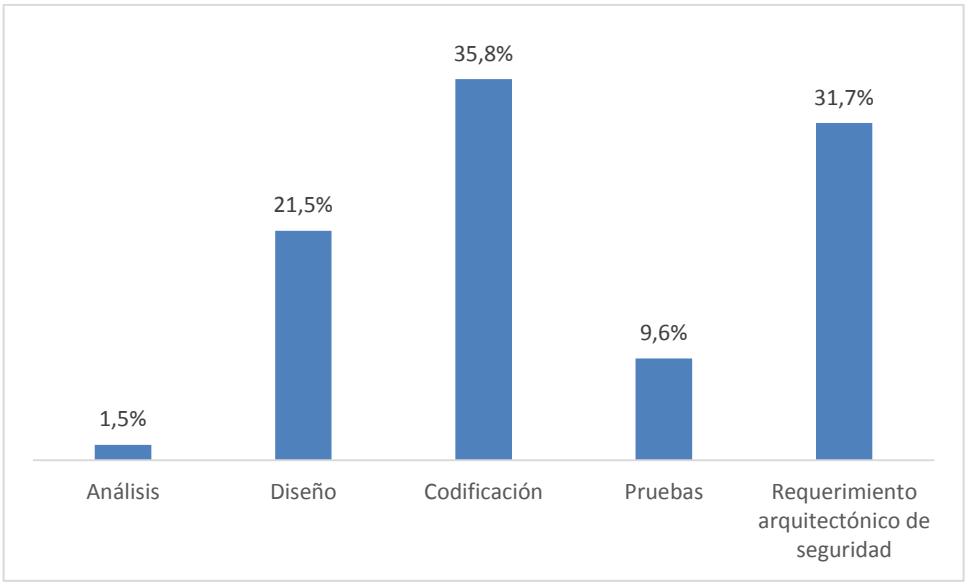
Figura 26. Nivel de complejidad de las historias de usuario de usuario.



Fuente: Esta investigación.

El tiempo estimado de trabajo para el *sprint* fue de 56 horas, pero el tiempo realmente trabajado fue de 56:51 horas. La etapa que más porcentaje de tiempo requirió fue la de diseño incluyendo el requerimiento arquitectónico de seguridad con un 53,2%, como se puede observar en la Figura 27Figura 14.

Figura 27. Porcentaje de tiempo de trabajo por etapa.



Fuente: Esta investigación - 2020.

Teniendo en cuenta que se realizó una planeación para trabajar 28 horas a la semana, en la Tabla 48Tabla 36. Tiempo de trabajo por semana., se puede mirar

que en la semana uno se trabajó un 22% más del tiempo presupuestado. También se puede observar que en la semana uno, el desarrollo tomó un 19% menos del tiempo que se planeó, por esta razón, hubo una compensación con respecto a la semana dos.

Tabla 48. Tiempo de trabajo por semana.

Semanas	Tiempo trabajado	Tiempo no trabajado	% Tiempo trabajado	% Tiempo No trabajado
Semana 1	34:05:00	0:00:00	122%	0%
Semana 2	22:46:00	5:14:00	81%	19%
Total	56:51:00	5:14:00		

Fuente: Esta investigación - 2020.

Finalmente, en la Tabla 49, se presenta un resumen del desempeño para el *sprint* 5, donde se observa que se trabajó un 2% más, hubo un desperdicio de 9% con respecto al tiempo total del tiempo trabajo para el ciclo, sin embargo, como se mencionó anteriormente, hubo una compensación en la primera semana con respecto a la segunda.

Tabla 49. Resumen del desempeño.

Tiempo total de trabajo planeado para el ciclo	56:00:00
Tiempo total trabajado	56:51:00
Tiempo de desperdicio total	5:14:00
% de tiempo trabajado	102%
% de desperdicio total	9%

Fuente: Esta investigación - 2020.

2.3.6. *Sprint* 6

A continuación, se presentan los resultados obtenidos en el desarrollo del *sprint 6.* (ver Anexo 6. sprint 6)

2.3.6.1. *Sprint backlog*

En la Tabla 50, se puede observar las cinco (5) historias de usuario priorizadas para el *sprint*, identificando la prioridad y complejidad.

Tabla 50. Sprint backlog.

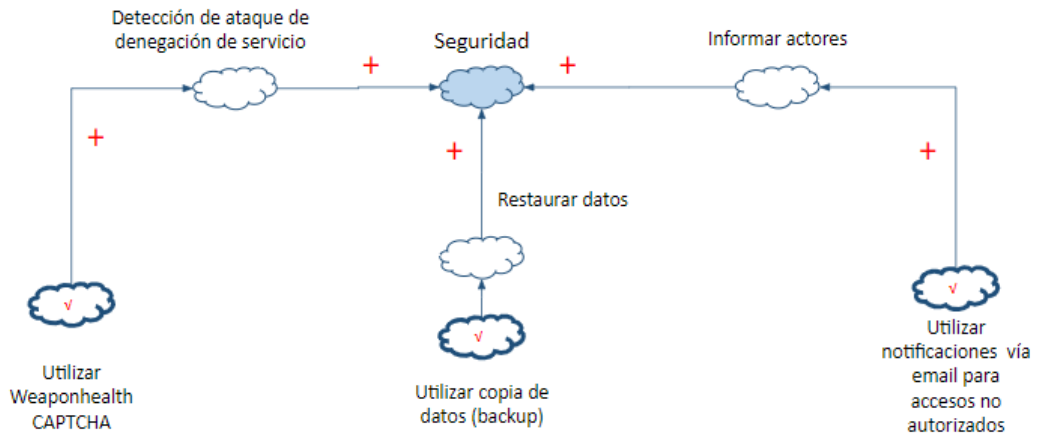
No.	Descripción de la Historia de Usuario	Prioridad	Complejidad
20	Como administrador quiero descargar un reporte de los votantes de un proceso de elección	80	Media
21	Como candidato quiero renunciar a la candidatura antes de que el proceso de elección comience	80	Media
23	Como administrador quiero actualizar los datos de los candidatos en caso de ser erróneos	56	Media
26	Como administrador quiero enviar un e-mail invitando a votar a las personas activas para el proceso de elección	8	Media
27	Como administrador quiero registrar otro usuario administrador	5	Baja

Fuente: Esta investigación - 2020.

2.3.6.2. Decisiones arquitectónicas de diseño

En la Figura 28Figura 19, se observa las decisiones de diseño adoptadas para el sexto *sprint*. Las tácticas arquitectónicas de seguridad para *e-voting* trabajadas en este *sprint* fueron: detección de ataque de denegación de servicio, restaurar datos e informar actores.

Figura 28. Árbol de objetivos suaves.



Fuente: Esta investigación – 2020.

Como se puede ver en la Figura 28, se especifica cuáles son las tecnologías que se usaron en la implementación del sistema con respecto a las tácticas correspondientes, a continuación, se describe brevemente cada una.

- *Weaponhealth CAPTCHA*

Para Google (2020), un *CAPTCHA* (*test* de *Turing* público y automático para distinguir a los ordenadores de los humanos, del inglés "*Completely Automated Public Turing test to tell Computers and Humans Apart*") es un tipo de medida de seguridad conocido como autenticación pregunta-respuesta. Un *CAPTCHA* te ayuda a protegerte del *spam* y del descifrado de contraseñas pidiéndote que completes una simple prueba que demuestre que eres humano y no un ordenador que intenta acceder a una cuenta protegida con contraseña （Google，2020）.

- Copia de datos (*backup*)

Según Rafinno (2020), en informática, se entiende por un *backup* (del inglés: *back up*, “respaldo”, “refuerzo”), respaldo, copia de seguridad o copia de reserva a una copia de los datos originales de un sistema de información o de un conjunto de software (archivos, documentos, etc.) que se almacena en un lugar seguro o una región segura de la memoria del sistema, con el fin de poder volver a disponer de su información en caso de que alguna eventualidad, accidente o desastre ocurra y ocasione su pérdida del sistema. En otras palabras, se trata de una copia por si

acaso que, usualmente, se actualiza cada cierto tiempo como medida de seguridad （Raffino，2020）.

En la Figura 29, se puede observar el árbol de objetivos suaves completo con las decisiones de diseño adoptadas hasta el *sprint.*

Figura 29. Árbol de objetivos suaves.

Fuente: Esta investigación - 2020

2.3.6.3. *Review*

El objetivo del *sprint* fue elaborar las cinco (5) historias de usuario seleccionadas y registradas en el *sprint backlog*, además del requerimiento arquitectónico de seguridad de restaurar datos, informar actores y detección de ataque de denegación de servicio.

En este *sprint* no se presentaron impedimentos y se avanzó en las cinco (5) historias de usuarios planeadas.

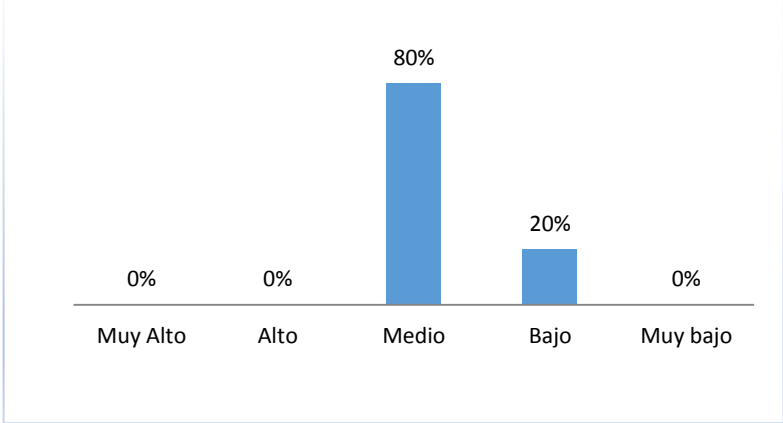
En el *sprint* se logró realizar cinco (5) historias de usuario del *sprint backlog* y se incorporó la copia de datos (*backup*), informar actores y validaciones *CAPTCHA*.

2.3.6.4. Medición

A continuación, se presenta la medición de productividad en el proceso de software. Está medición se realizó a través de los indicadores: complejidad de las historias de usuario, tiempo de trabajo por etapa, tiempo de trabajo por semana y desempeño.

Como se puede evidenciar en la Figura 30, la complejidad de historias de usuario con mayor frecuencia fue la media con un 80%.

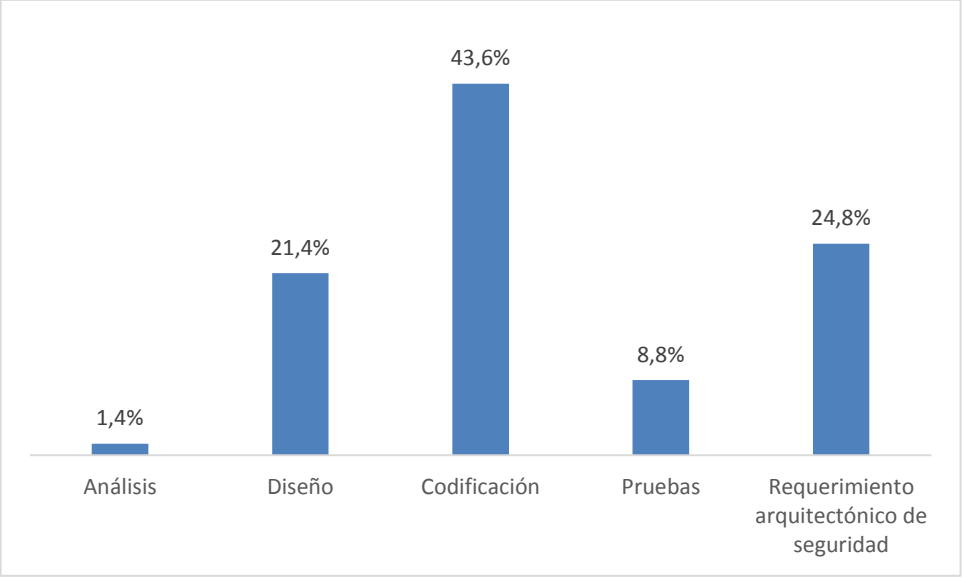
Figura 30. Nivel de complejidad de las historias de usuario de usuario.



Fuente: Esta investigación.

El tiempo estimado de trabajo para el *sprint* fue de 56 horas, pero el tiempo realmente trabajado fue de 60:25 horas. La etapa que más porcentaje de tiempo requirió fue la de diseño, incluyendo el requerimiento arquitectónico de seguridad con un 46,2%, como se puede observar en la Figura 31Figura 14.

Figura 31. Porcentaje de tiempo de trabajo por etapa.



Fuente: Esta investigación - 2020.

Teniendo en cuenta que se realizó una planeación para trabajar 28 horas a la semana, en la Tabla 51Tabla 42Tabla 36. Tiempo de trabajo por semana., se puede

mirar que en la semana uno se trabajó un 35% más del tiempo presupuestado. También se puede observar que en la semana dos, el desarrollo tomó un 19% menos del tiempo que se planeó, por esta razón, hubo una compensación con respecto a la semana dos.

Tabla 51. Tiempo de trabajo por semana.

Semanas	Tiempo trabajado	Tiempo no trabajado	% Tiempo trabajado	% Tiempo No trabajado
Semana 1	37:40:00	0:00:00	135%	0%
Semana 2	22:45:00	5:15:00	81%	19%
Total	60:25:00	5:15:00		

Fuente: Esta investigación - 2020.

Finalmente, en la Tabla 52, se presenta un resumen del desempeño para el *sprint* 6, donde se observa que se trabajó un 8% más, hubo un desperdicio de 9% con respecto al tiempo total del tiempo trabajo para el ciclo, sin embargo, como se mencionó anteriormente, hubo una compensación en la primera semana con respecto a la segunda.

Tabla 52. Resumen del desempeño.

Tiempo total de trabajo planeado para el ciclo	56:00:00
Tiempo total trabajado	60:25:00
Tiempo de desperdicio total	5:15:00
% de tiempo trabajado	108%
% de desperdicio total	9%

Fuente: Esta investigación - 2020.

2.3.7. Síntesis de la construcción del prototipo

Se logró construir un producto software *e-voting* funcional llamado “Kybernan”, que implementa tácticas arquitectónicas de seguridad para la gestión de procesos democráticos mediante la realización de veintisiete (27) historias de usuario, con un porcentaje de 74,07% según el nivel de dificultad Media.

De acuerdo con el requerimiento arquitectónico de seguridad, se implementaron 8 tácticas arquitectónicas. Dentro de las decisiones de diseño más importantes se

encontró que la implementación de las tecnologías *JWT* y el uso de roles y permisos tipo *grant*, fueron las que aportaron con mayor frecuencia al sistema.

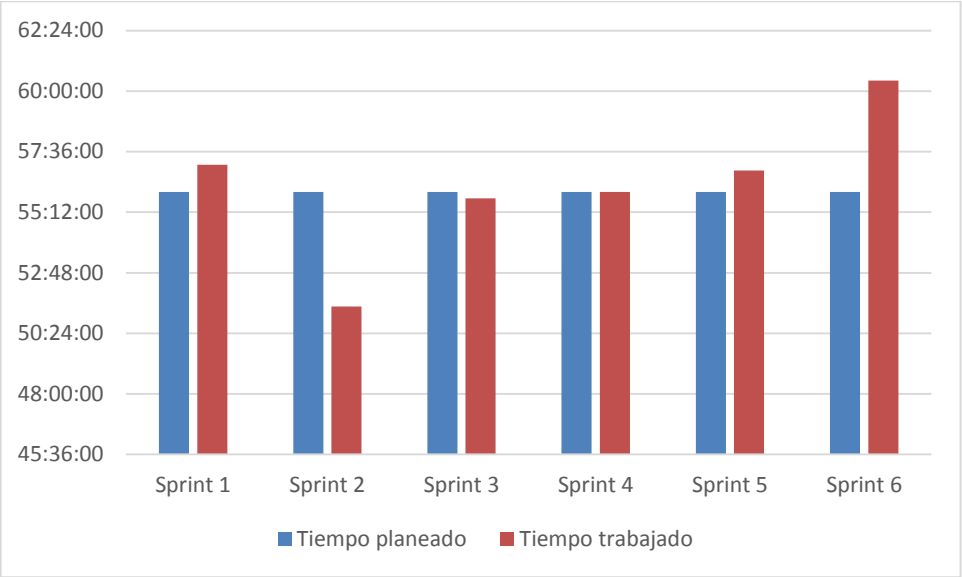
De los seis (6) *sprints,* en un 83,3% se logró cumplir con el objetivo planeado. Entre los principales impedimentos para no cumplir con los objetivos definidos para los *sprints* están: dar respuesta al manejo de sesiones y autorización de actores, estas actividades se incorporaron en el *sprint backlog*, pero se estimaron, por esta razón, se avanzaron dos de las cinco historias de usuario para el sprint 1, adicionalmente se extendió la implementación de la táctica de autenticar actores, debido a un inconveniente de compatibilidad entre las versiones de angular y *face API*.

En relación con la productividad en el desarrollo del proyecto, se puede afirmar que la etapa que mayor porcentaje de tiempo requirió fue la de diseño porque el desarrollo del sistema estuvo centrado en implementar tácticas arquitectónicas de diseño. En total se trabajaron 337:34 horas en el proyecto y de acuerdo con la planeación, se presentó un porcentaje de desperdicio de 6%.

Durante el inicio del desarrollo del sistema, la eficiencia no fue la mejor, por ejemplo, en el primer *sprint* se planearon desarrollar cinco (5) historias de usuario, de las cuales se desarrollaron dos (2), sin embargo, a medida que el desarrollo fue avanzando, el desempeño fue mejorando, al punto de que se logró implementar hasta ocho (8) historias de usuario y el requerimiento arquitectónico de seguridad en un mismo *sprint*.

Como se puede observar en la Figura 32. Tiempo planeado en comparación con el tiempo trabajado., a partir del tercer *sprint*, el tiempo trabajado se incrementa cada vez más con respecto al anterior sprint, por lo tanto, se puede decir que, en la mayoría de los *sprints*, se trabajó más del tiempo planeado.

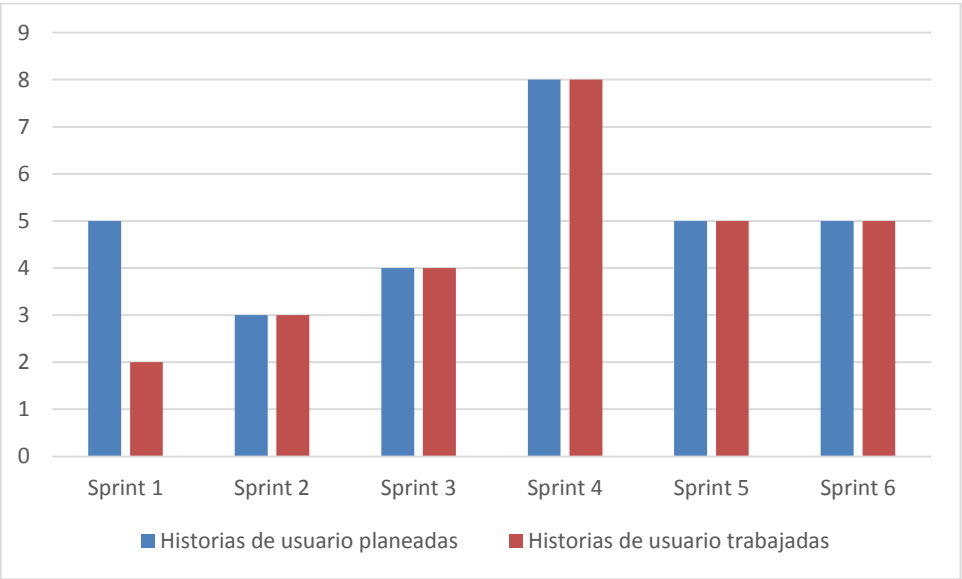
Figura 32. Tiempo planeado en comparación con el tiempo trabajado.



Fuente: Esta investigación – 2020.

En la Figura 33, se puede mirar que se logró cumplir con todas las historias de usuario planeadas a excepción del *sprint* 1, donde solo se logró cumplir con dos (2) de las cinco (5) historias de usuario planeadas.

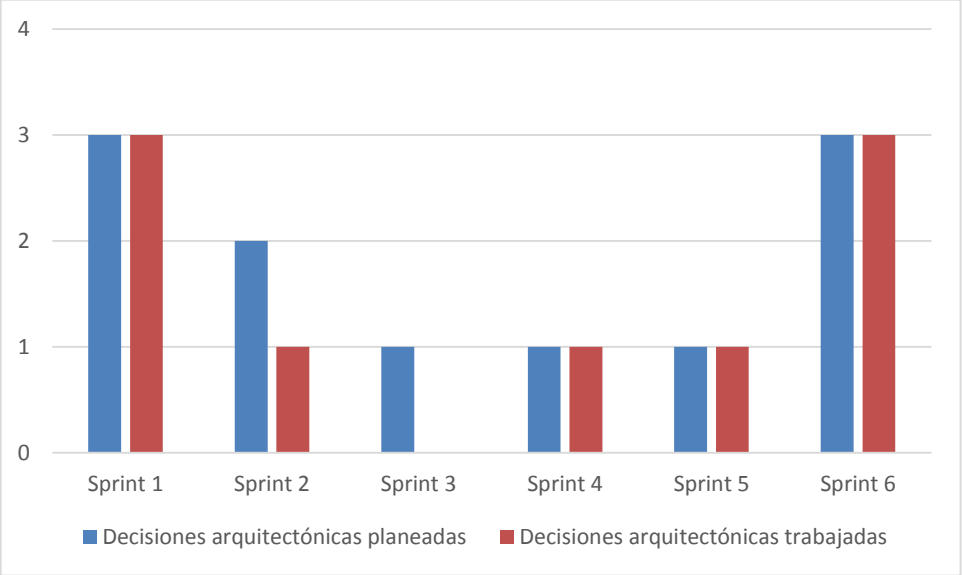
Figura 33. Historias de usuario planeadas en comparación con las historias de usuario trabajadas



Fuente: Esta investigación – 2020.

De la Figura 34, se puede concluir que, para el segundo y tercer *sprint*, no se logró completar las decisiones arquitectónicas planeadas, sin embargo, para los demás *sprints* si se logró implementarlas satisfactoriamente.

Figura 34. Decisiones arquitectónicas planeadas en comparación con las decisiones arquitectónicas trabajadas.



Fuente: Esta investigación - 2020.

2.4 APROPIACIÓN SOCIAL DE CONOCIMIENTO

El trabajo investigativo realizado ha permitido divulgar los resultados en el 4to Congreso Andino en Computación, Informática y Educación – 2019, y en el 1er Congreso Internacional en Ciencia e Ingeniería – 2020. Además, se cuenta con un artículo que será publicado como capítulo en la segunda versión del libro “Nuevas tendencias investigativas en computación, informática y educación en ingeniería”. Los certificados y artículo se pueden observar en el Anexo 7. Artículo y certificados.

3. CONCLUSIONES

Al finalizar el proceso de investigación, teniendo en cuenta la fundamentación teórica, las experiencias y resultados logrados, se obtuvieron las siguientes conclusiones:

Las tácticas arquitectónicas de seguridad se orientaron principalmente a detectar, resistir, reaccionar y recuperarse, a ataques; junto con auditar, autenticar y establecer sesiones seguras. Los campos de mayor aplicación para estas tácticas fueron la industria, la academia y el gobierno. Además, se logró sistematizar las tácticas, identificando las principales prácticas, efectos y los recursos utilizados para implementarlas.

Las tácticas arquitectónicas de seguridad pertinentes para un producto software de *e-voting,* identificadas de manera sistemática son aquellas que permitieron detectar, resistir, reaccionar y recuperarse ante ataques.

Se construyó Kybernan, un producto software para *e-voting,* que implementa ocho tácticas arquitectónicas de seguridad para la gestión de procesos democráticos. Dentro de las decisiones de diseño más importantes incorporadas está la implementación de tecnologías *JWT (Por sus siglas en inglés Json Web Token.)* y el uso de roles y permisos tipo *grant*.

El sistema en un ambiente experimental no permitió exponer en su totalidad las pruebas de seguridad que se pueden presentar en un ambiente de producción, esto se debe a que, bajo este entorno de desarrollo, se expuso al sistema en escenarios puntuales y controlados.

Usando la técnica de revisión sistemática de literatura (LRS), no se encontró ninguna táctica arquitectónica de seguridad enfocada directamente a un sistema *e-voting* en todos los estudios seleccionados.

En el desarrollo del producto software Kybernan, al estar centrado en la implementación de tácticas arquitectónicas, la etapa que conllevó mayor porcentaje de tiempo es la de diseño con un 48% del total trabajado.

4. RECOMENDACIONES

Una de las tácticas arquitectónicas de seguridad pertinentes para un producto software de *e-voting* es la detección de ataques. Por esta razón, se sugiere que para la implementación de un producto software se tomen decisiones de diseño en relación con la detección de intrusos, denegación de servicio y verificación de la integridad de los mensajes.

Se construye Kybernan, un producto software para *e-voting,* que implementa ocho tácticas arquitectónicas de seguridad para la gestión de procesos democráticos. Las pruebas de Kybernan se realizan en un ambiente experimental, por esta razón se recomienda en futuros proyectos desplegarlo en una infraestructura tecnológica donde se pueda analizar decisiones de diseño orientadas a tácticas para la detección de intrusos, limitar exposición, encriptar datos y verificar integridad del mensaje. Además, para autenticar actores Kybernan utiliza un componente de software libre denominado *face API*. En este orden de ideas, para futuros trabajos se recomienda realizar un análisis comparativo incorporando componentes propietarios y elementos de autenticación biométrica.

BIBLIOGRAFÍA

A practical, secure, and auditable e-voting system. **Benha University**Banha, Egipto, Elsevier, 2017.

AlonsoPanizo *Desarrollo de una metodología para el análisis y la clasificación de los sistemas de voto electrónico:*. León, s.n., 2014.

BARBACCI, Mario; MARK, Klein; THOMAS A; LONGSTAFF, CHARLES, B; WEINSTOCK *Technical report, CMU/SEI-95-TR-021, ESC-TR-95-021, Quality Attributes*. 1995.

BASSLen, CLEMENTS y KAZMAN *Software architecture in practice, Third Edition,Capítulo 4: Understanding quality attributes.*2013.

BassLen, KazmanRick, ClementsP *Software architecture in practice*. s.l., Addison-Wesley Professional, 2013.

BBC Mundowww.bbc.com. *Qué dice de Colombia que haya habido 62% de abstención en el histórico plebiscito por el proceso de paz*. [En línea]3 Octubre 2016. [Citado el: 2018 Mayo 28.]http://www.bbc.com/mundo/noticias-america-latina-37539590.

BuschmannF.,Meunier, R., Rohnert, H., Sommerland, P. *Pattern oriented software architecture: a system of patterns*. s.l., John Wiley & Sons, 1996.

Christine Hofmeister, Robert Nord y Dilip Soni Describing software architecture with uml. San Antonio, s.n., 1999, págs.145-160.

ELTIEMPOEl 46,6 por ciento de los colombianos no votaron. [En línea]28 05 2018. [Citado el: 28 05 2018.]http://www.eltiempo.com/elecciones-colombia-2018/presidenciales/colombianos-que-no-votaron-a-la-presidencia-223064.

Estudiantes de Comunicación Social y Periodismo – Universidad Los Libertadores Las 2 Orillas. *¿Es viable y necesario el voto electrónico en*

Colombia?[En línea]3 Abril 2018. https://www.las2orillas.co/es-viable-y-necesario-el-voto-electronico-en-colombia/.

GarciaPabloy otros *Anonimato en sistemas de e-voting: últimos avances.* La Pampa - Argentina, s.n., 2016.

GoogleAdministrador de Google Workspace . [En línea]2020 . https://support.google.com/a/answer/1217728?hl=es.

GRANT, R. *Dirección estratégica. Conceptos, técnicas y aplicaciones .* s.l. , THOMSON – CIVITAS., 2004 .

Hacia el desarrollo de un prototipo de sistema de voto electrónico para Costa Rica . **Schmidth-PeraltaJ., Gutiérrez-AlfaroJ.**3, Costa Rica, Editorial tecnológica de Costa Rica, 2016, Vol. 29.

IBM27754_IBM_WP_Native_Web_or_hybrid_2846853 El desarrollo de aplicaciones móviles nativas,. *FTP IBM software.* [En línea]12 Abril 2012. [Citado el: 4 Mayo 2017 .]ftp://ftp.software.ibm.com/la/documents/gb/commons/27754_IBM_WP_Native_Web_or_hybrid_2846853.pdf.

Implementing and evaluating a softwareindependent . **Jurlind BudurushiRomanJöris, Melanie Volkamer**Darmstadt, Elsevier, 2014.

ISO/IECISO 25010 . *ISO/IEC 25010 .* [En línea][Citado el: 27 Mayo 2018 .]http://iso25000.com/index.php/normas-iso-25000/iso-25010.

KitchenhamBarbara , BudgenDavid , BreretonPearl *Evidence-Based Software Engineering and Systematic Reviews.* s.l., Boca Raton, FL : CRC Press, 2015.

Kobayashi y otros *Quantitative Non Functional Requirements evaluation using softgoal weight. J. Internet Serv. Inf. Secur., 6(1), 37-46.*2016.

Ministerio de Salud, COLCIENCIAS *SABE Colombia 2015: estudio nacional de salud, bienestar y envejecimiento .* Bogota (Colombia) , Ministerio de Salud, COLCIENCIAS, 2015.

MoodleRoles y permisos . *Moodle .* [En línea]11 Septiembre 2020 . https://docs.moodle.org/all/es/Roles_y_permisos.

MoralesItaloBorrado Lógico, método softDeletes en Laravel . *rimorsoft .* [En línea]5 Agosto 2019. https://rimorsoft.com/borrado-logico-metodo-softdeletes-en-laravel#:~:text=As%C3%AD%20que%20podemos%20traducir%20lo,est%C3%A1ndar%2C%20entenderemos%20mejor%20este%20tema..

MühlerVincentface-api.js . *justadudewhohacks .* [En línea]24 Octubre 2018 . https://justadudewhohacks.github.io/face-api.js/docs/index.html.

PaszniukRodrigo¿Qué es Json Web Token (JWT)?*Programación .* [En línea]4 Julio 2016 . https://www.programacion.com.py/varios/que-es-json-web-token-jwt#:~:text=Json%20Web%20Token%20es%20un,JSON%20Web%20Signature(%20JWS%20)..

Patricia PesadoGuillermoFeierherd, Ariel Pasini *Especificación de Requerimientos para Sistemas de Voto Electrónico .* Ushuaia, s.n., 2015.

RaffinoMaríaEstelaConcepto.de . [En línea]27 06 2020 . https://concepto.de/backup/.

RamosDavidA fondo: ¿Es seguro el e-voting? (I). [En línea]14 Marzo 2017. [Citado el: 2018 Marzo 28.]https://www.silicon.es/seguridad-e-voting-i-2331029.

Registraduría Nacional del Estado civilASÍ PARTICIPAN LOS COLOMBIANOS EN LAS ELECCIONES PRESIDENCIALES. [En línea]2014. [Citado el: 2018 05 28.]https://wsr.registraduria.gov.co/Asi-participan-los-colombianos-en.html.

Revista Semana"En Colombia nunca se ha hecho una auditoría independiente al software de la Registraduría" . *Revista Semana .* [En línea]22 Mayo 2018 . https://www.semana.com/elecciones-presidenciales2018/candidatos-elecciones-presidencia-2018/articulo/polemica-por-auditoria-al-software-de-la-registraduria/568155/.

RSLPolítica de Cookies . *RSL Programación y Diseño Web .* [En línea]19 Septiembre 2017 . https://www.rslprogramacion.es/politica-de-

cookies/#:~:text=Las%20cookies%20permiten%20a%20una,utilizarse%20para%20reconocer%20al%20usuario..

Superintendencia de sociedadesPROCEDIMIENTO DE GESTION DE LOGS Y REGISTROS DE AUDITORÍA . [En línea]04 12 2017 . https://www.supersociedades.gov.co/superintendencia/oficina-asesora-de-planeacion/polinemanu/sgi/Documents/Documentos%20Infraestructura%20Tecnologica/Documentos/GINT-PR-007%20%20Gestion%20de%20logs_y_registros_de_auditoria.pdf.

Understanding Software Vulnerabilities Related to Architectural Security Tactics. **J. C. Santos, A. Peruma, M. Mirakhorli, M.**2017, IEEE International Conference on Software Architecture.

VittoneJosé, CuelloJavier Las aplicaciones. *Basics to design native apps.*[En línea]1 Febrero 2014 . [Citado el: 4 Mayo 2017 .]http://appdesignbook.com/es/contenidos/las-aplicaciones/.