Anexo 15: Modelo DevOps

PROCEDIMIENTOS Y HERRAMIENTAS DEVOPS

Como se mencionó en el capítulo anterior, a partir de aquí vamos a presentar los procesos, herramientas y ejemplos del modelo DevOps para el proyecto Cuenta Única. Se empieza por los conceptos de infraestructura como código, después tenemos integración continua, entrega continua, implantación continua y monitoreo.

Las herramientas no son restrictivas, es factible proponer el cambio por tecnologías similares u homologas (siempre y cuando cuenten con soporte del Proveedor de Plataforma de **LA SUNAT** y estén basados en productos open source).

Para el caso de aprovisionamiento en la Plataforma de **LA SUNAT**, esta asume el costo de consumo de estas herramientas.

Para el caso de aprovisionamiento en la Plataforma de **LA FIRMA CONSULTORA**, el costo es por cuenta de esta última.

INFRAESTRUCTURA COMO CÓDIGO

La propuesta para la etapa de Operaciones es la Infraestructura como código y será escalada en la nube (*cloud*). En ella se puede crear una infraestructura completa fuera del área física de la empresa, resultando en mayor seguridad y posibilidad de acceso a cualquier lugar que se haga necesario, necesitando sólo tener un dispositivo con acceso a internet. Además de proporcionar el escalonamiento de recursos, o sea, posibilitar la expansión de los recursos solamente cuando hay real necesidad de utilización y reducción de estos cuando la demanda es baja, ahorrando dinero.

El enfoque de Infraestructura como código es la entrega de una estructura ágil (administrar la configuración y automatizar el aprovisionamiento e implementaciones), utilizando codificación simple y objetiva sin la necesidad de diversos pasos y procesos para prepararse un ambiente cuando se configuran las máquinas manualmente. Otra ventaja es la posibilidad de aplicar el mismo código cada vez que sea necesaria la creación de un nuevo ambiente o si una función no trabaja correctamente.

Otro lenguaje bastante popular es Ansible:

Ansible es una herramienta de automatización de infraestructuras. Se pueden configurar sistemas, implementar software y orquestar tareas más avanzadas, como implementaciones continuas o actualizaciones continuas. Es adecuado para administrar todos los entornos, desde pequeñas configuraciones con un puñado de instancias hasta entornos corporativos con muchos miles de instancias.

Los principales objetivos de Ansible son simplicidad y facilidad de uso. También tiene un fuerte foco en seguridad y confiabilidad, presentando un mínimo de partes móviles, uso de OpenSSH para transporte (con otros transportes y modos *pull* como alternativas).

Otro punto, que no es característica solamente de ambientes en la nube, es la necesidad de monitoreo, tanto de los recursos de la infraestructura, como de los servicios dentro del Kubernetes. Con eso, garantizamos que, cuando algún recurso/servicio falla, tengamos una acción de recuperación y el problema sea resuelto en el menor tiempo posible.

La estructura elegida para la nube del proyecto Cuenta Única es la siguiente:

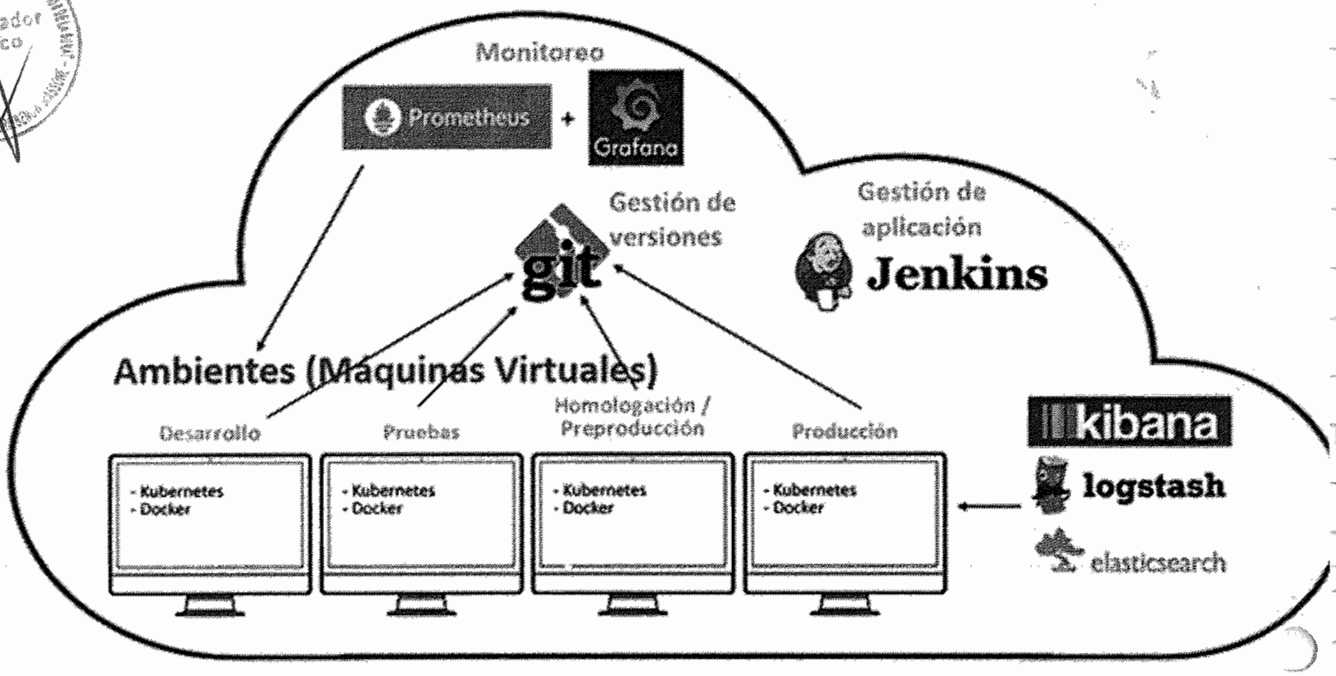

Ilustración 1 - Modelo de Operaciones para el proyecto Cuenta Única

En el modelo propuesto, se tienen los ambientes de aplicación: Pruebas, Homologación y Producción, donde los tres tendrán los mismos *softwares*.

El ambiente de Desarrollo es donde los desarrolladores pueden guardar y validar la versión antes de la implantación en ambiente de Pruebas.

El ambiente de Pruebas es para la implantación del código de desarrollo, en este ambiente serán ejecutadas las pruebas automatizadas.

El ambiente de Homologación (Pre-Producción) es para validación de la próxima versión que entrará en producción. También puede ser llamado de ambiente preproducción. En él serán ejecutadas pruebas funcionales (no automatizadas) y no funcionales. Las implantaciones serán aplicadas por el Gestor de acuerdo con su planificación.

El ambiente de Producción es donde estará la última versión de la aplicación disponible para la interacción con el usuario final.

Otra exigencia del modelo es que la configuración de *hardware* de las máquinas virtuales de Homologación y Producción deben ser las mismas.

Con el ambiente configurado, utilizaremos los archivos *JenkinsFile* y *DockerFile* para hacer la creación de las imágenes Docker, *tagging* y el Despliegue automatizado.

MONITOREO DE INFRAESTRUCTURA Y MICROSERVICIOS

El monitoreo de los servicios tendremos el *software* Prometheus con la *interface* del *software* Grafana.

Prometheus es un conjunto de herramientas de monitoreo y alerta de sistemas *open source* originalmente creado en SoundCloud.

Sus características principales son:

- Un modelo de datos multidimensional con datos de series de tiempo identificados por nombre de métrica y pares clave / valor.

- Un lenguaje de consulta flexible para aprovechar esta escalabilidad.
- No confiar en el almacenamiento distribuido: los nodos de servidor único son autónomos.
- La colección de series de tiempo ocurre a través de un modelo de extracción a través de HTTP.
- La serie de tiempo de empuje es compatible a través de una puerta de enlace intermediaria.
- "Targets" se descubren mediante servicios de descubrimiento o configuración estática.
- Múltiples modos de soporte gráfico y de tablero

El ecosistema Prometheus consta de múltiples componentes, muchos de los cuales son opcionales:

- El servidor Prometheus principal que raspa y almacena datos de series de tiempo.
- Bibliotecas cliente para instrumentar código de aplicación.
- Una puerta de enlace para apoyar trabajos de corta duración.
- Exportadores de propósito específico para servicios como HAProxy, StatsD, Graphite, etc.
- Un gestor de alerta para manejar alertas.
- Varias herramientas de soporte.

Grafana, conforme descrito en Grafana Labs, es una aplicación de código abierto de propósito general y compositor gráfico, que se ejecuta como una aplicación web. Permite consultar, visualizar, alertar y comprender sus métricas, sin importar dónde estén almacenadas. Es compatible con Graphite, InfluxDB o OpenTSDB como *backends*.

Para el monitoreo de los recursos de la nube, proponemos los *softwares* Elasticsearch, Logstash y Kibana de la empresa Elasticsearch BV, que describe las herramientas como:

El Elasticsearch es un motor de búsqueda RESTful distribuido creado para la nube. Él captura las informaciones de los recursos para generar los logs en LogStach.

Sus características incluyen:

- Motor de búsqueda distribuido y altamente disponible:
  - Cada índice está completamente fragmentado con una cantidad configurable de fragmentos.
  - Cada fragmento puede tener una o más réplicas.
  - Operaciones de lectura / búsqueda realizadas en cualquiera de los fragmentos de réplica.
- *Multi Tenant*:
  - Soporte para más de un índice.
  - Configuración de nivel de índice.
- Varios conjuntos de API:
  - API HTTP RESTful.
  - API nativa de Java.
  - Todas las API realizan un redireccionamiento automático de la operación del nodo.
- *Document oriented.*

- Sin necesidad de una definición de esquema inicial.
- El esquema se puede definir para la personalización del proceso de indexación.

- Escritura asíncrona confiable para la persistencia a largo plazo.
- *Near Real-Time Search.*
- Construido sobre Lucene:
  - Cada fragmento es un índice Lucene completamente funcional.
  - Todo el poder de Lucene se expone fácilmente a través de configuraciones / complementos simples.
- *Per operation consistency*:
  - Las operaciones de nivel de documento único son atómicas, consistentes, aisladas y duraderas.

Logstash es una fuente de procesamiento de datos de código abierto, del lado del servidor que ingiere datos de una multitud de fuentes simultáneamente, las transforma y luego las envía a Elasticsearch.

Los datos a menudo se encuentran dispersos o en silos en muchos sistemas en muchos formatos. Logstash admite una variedad de entradas que extraen eventos de múltiples fuentes comunes, todas al mismo tiempo. Ingesta fácilmente desde los registros, métricas, aplicaciones web, *data stores* y varios servicios de nube, todo en forma continua.

A la medida en que los datos viajan de la fuente al repositorio, los filtros de Logstash analizan cada evento, identifican campos con nombre para construir la estructura y los transforman en un formato común para un análisis más rápido y fácil.

Logstash transforma dinámicamente y prepara sus datos independientemente del formato o la complejidad:

- Descifrar las coordenadas geográficas de las direcciones IP.
- Anonimizar los datos PII, excluir completamente los campos sensibles.
- Facilite el procesamiento general independientemente de la fuente de datos, del formato o del esquema.

Para la creación de gráficos y *dashboards*, tenemos la herramienta Kibana, que es un plugin de visualización de datos de fuente abierta para el Elasticsearch. Ella proporciona recursos de visualización sobre el contenido indexado en un *cluster* Elasticsearch. Los usuarios pueden crear gráficos de barra, línea y dispersión, o otros gráficos sobre grandes volúmenes de datos.

INTEGRACIÓN CONTINUA

NOMENCLATURA DE ARTEFACTOS

Antes de presentar los procesos y herramientas, es necesario implementar una nomenclatura estándar para los archivos .JAR, directorios, paquetes, versiones y tags del proyecto.

El estándar más indicado para los JAR, directorios, paquetes, es el estándar de SUN, donde los nombres de los archivos son relativos al nombre de la empresa que desarrolló la clase.

Comenzando con el prefijo del país del proyecto: "pe", seguida por la palabra "com", después el nombre de la empresa, nombre del proyecto y, por final, el nombre del paquete:

pe.com.nombredelaempresa.nombredelproyecto.paquete

Donde tenemos: **pe.gob.sunat.<megaproceso.macroproceso.>.cuentaunica.primerpaquete**

Donde **<megaproceso.macroproceso.>** se definirá en la etapa de desarrollo por el Líder de desarrollo.

Los paquetes sólo tienen letras minúsculas, no importa cuántas palabras estén contenidas en él. Este estándar existe para evitar al máximo el conflicto de paquetes de empresas diferentes.

El nombre del archivo .JAR debe contener el nombre del proyecto y su versión, como: **cuentaunica_v1-0-0.jar**.

Sobre el nombre de las imágenes Docker, no existe un patrón definido, pero debemos utilizar algo

que haga referencia al proyecto y al contenido de la imagen generada.

Como buena práctica, utilizaremos el nombre de la imagen compuesto de:

Entorno en el que se utilizará + Fecha de creación + Versión

Es decir, si la primera imagen del entorno de desarrollo se creó el 15/06/2018, el nombre de la imagen debe ser: **development_150618_v01**

Para los nombres de versiones y *tags*, utilizaremos el concepto de *Semantic Versioning*, donde las palabras claves "DEBE", "NO DEBE", "OBLIGATORIO", "DEBERÁ", "NO DEBERÁ", "PUEDEN", "NO PUEDEN", "RECOMENDADO", "PUEDE" y "OPCIONAL", en este documento, se interpreta como se describe en RFC 2119:

- El *software* que utiliza la versión semántica DEBE declarar una API pública. Esta API puede ser declarada en el propio código o existir estrictamente en la documentación, siempre que sea precisa y comprensiva.
- Un número de versión normal DEBE tener el formato de X.Y.Z, donde X, Y, y Z son enteros no negativos, y NO DEBE contener ceros a la izquierda. X es la versión Mayor, Y es la versión Menor, y Z es la versión de Corrección. Cada elemento DEBE aumentar numéricamente. Por ejemplo: 1.9.0 -> 1.10.0 -> 1.11.0.
- Una vez que un paquete versionado ha sido liberado, el contenido de esta versión NO DEBE modificarse. Cualquier modificación DEBE ser lanzado como una nueva versión.
- Al principio del desarrollo, la versión Mayor DEBE ser cero (0.y.z). Cualquier cosa puede cambiar en cualquier momento. El API pública no debe considerarse estable.
- La versión 1.0.0 define el API como pública. La forma en que el número de versión se incrementa después de que esta contabilización es dependiente del API pública y cómo se cambia.
- Versión de corrección Z (x.y.Z | x> 0) DEBE incrementarse sólo si mantiene compatibilidad e introducir corrección de errores. Una corrección de errores se define como un cambio interno que corrige un comportamiento incorrecto.
- Versión Menor Y (x.Y.z | x> 0) DEBE incrementarse si se introduce una funcionalidad nueva y compatible en el API pública. DEBE ser incrementada si cualquier funcionalidad del API pública se establece en descontinuada. PUEDE ser incrementada si una nueva funcionalidad o mejora sustancial se introduce dentro del código privado. PUEDE incluir cambios a nivel de corrección. La versión de corrección se debe restablecer a 0 (cero) cuando se incrementa la versión Menor.
- (X.y.z | X> 0) DEBE ser incrementada si se introducen cambios incompatibles en el API pública. Puede incluir cambios a nivel de versión menor y de versión de corrección. La versión

de corrección y la versión menor se deben restablecer a 0 (cero) cuando se incrementa la versión más grande.

- Una versión preliminar (pre-release) se puede identificar añadiendo un guion (*dash*) y una serie de identificadores separados por punto (punto) inmediatamente después de la versión de corrección. Identificador DEBE incluir sólo caracteres alfanuméricos y guiones [0-9A-Za-z-]. El identificador NO DEBE ser vacío. Indicador numérico NO DEBE incluir ceros a la izquierda. La versión de prelanzamiento tiene una prioridad inferior a la versión normal a la que está asociada. Una versión preliminar (pre-release) indica que la versión es inestable y puede que no cumpla los requisitos de compatibilidad deseados, según lo indicado por su versión normal asociada. Ejemplos: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92.

- Los metadatos de construcción (*Build*) se pueden identificar añadiendo un signo más (+) y una serie de identificadores separados por punto inmediatamente después de la corrección o prelanzamiento. Identificador DEBE ser compuesto sólo por caracteres alfanuméricos y guiones [0-9A-Za-z-]. El identificador NO DEBE ser vacío. Los metadatos de construcción PUEDEN ser ignorados cuando se determina la versión de precedencia. Así, dos versiones que difieren sólo en los metadatos de construcción tienen la misma precedencia.

Ejemplos: 1.0.0-alpha + 001, 1.0.0 + 20130313144700, 1.0.0-beta + exp.sha.5114f85.

- La precedencia se refiere a cómo las versiones se comparan con cada otra cuando se solicita. La precedencia DEBE ser calculada separando identificadores de versión en Mayor, Menor, Corrección y Prelanzamiento, en este orden (Metadatos de construcción no figuran en la precedencia). La precedencia se determina por la primera diferencia cuando se compara cada identificador de izquierda a derecha, como sigue: Versiones Mayor, Menor y Corrección siempre se comparan numéricamente. Ejemplo: 1.0.0 <2.0.0 <2.1.0 <2.1.1. Cuando Mayor, Menor y Corrección son iguales, la versión de Pre-Lanzamiento tiene precedencia menor que la versión normal. Ejemplo: 1.0.0-alpha <1.0.0. La precedencia entre dos versiones de Prelanzamiento con la misma versión Mayor, Menor y Corrección DEBE ser determinada comparando cada identificador separado por punto de izquierda a derecha hasta que se encuentre diferencia de la siguiente forma: identificadores consistentes sólo dígitos se comparan numéricamente e identificadores con letras o guion se comparan lexicalmente en el orden de clasificación ASCII. Los identificadores numéricos siempre tienen menos precedencia que los no numéricos. Un conjunto mayor de campos de pre-lanzamiento tiene una precedencia mayor que un conjunto más pequeño, si todos los identificadores anteriores son iguales: 1.0.0-alpha <1.0.0-alpha.1 <1.0.0-alpha.beta <1.0.0-beta <1.0.0-beta.2 <1.0.0-beta <1.0.0-alfa <1.0.0- rc.1 <1.0.0.

HERRAMIENTA DE GESTIÓN DE VERSIONES

La herramienta seleccionada para la gestión de versiones del proyecto Cuenta Única es Git, y la metodología de flujo seleccionada es el Gitflow. Esta metodología consiste en un paso-a-paso de *branches*, desde cuando se salva un archivo hasta que lo disponga en el *branch* "Master", o sea, del desarrollo inicial hasta la entrega final.

HERRAMIENTA DE GESTIÓN DE DESPLIEGUE

La gestión de despliegue es el proceso que utiliza una herramienta para realizar el *build*, *deploy* y ejecución de pruebas, todos automatizadas. Para ella usaremos la herramienta Jenkins. Como citado en su *website*, es un servidor autónomo de código abierto que se puede usar para automatizar todo tipo de tareas relacionadas con la construcción, prueba y entrega o implementación de *software*.

HERRAMIENTA DE REVISIÓN DE CÓDIGO

Como herramienta para revisión de código utilizaremos el GitLab, que es una aplicación única con recursos para todo el ciclo de vida de desarrollo y las operaciones de *software*. Es un proyecto de código abierto, utilizado por más de 100.000 organizaciones en todo el mundo y tiene una gran comunidad de más de 1.900 personas que contribuyeron con código.

HERRAMIENTA DE AUTOMATIZACIÓN DE BUILDS

Sobre automatización de *builds*, se utilizará el Gradle, que es un sistema avanzado de automatización de *builds* de código abierto que tiene la combinación de la flexibilidad del Ant con la administración de dependencias, la vida y la estructura de directorios de Maven. Utiliza DAG (*Directed Acyclic Graph*) para determinar el orden en que se pueden ejecutar las tareas.

ENTREGA CONTINUA

HERRAMIENTA DE *CONTAINERS*

Teniendo en cuenta que el proceso de entrega continua ocurre después de la integración continua, o sea, después de que el código ya ha pasado por el *build* y las pruebas unitarias, el siguiente paso que Jenkins realiza es la creación de imagen Docker y almacenarla en el *Docker Registry* para ser implantada.

Docker es la empresa que impulsa el movimiento de contenedores y el único proveedor de plataformas de contenedores que aborda cada aplicación a través de nube híbrida. Las empresas de hoy en día están bajo presión para transformarse digitalmente, pero están limitadas por aplicaciones e infraestructura existentes mintras racionalizan una cartera cada vez más diversa de nubes, centros de datos y arquitecturas de aplicaciones. Docker permite una verdadera independencia entre las aplicaciones y la infraestructura y los desarrolladores y las operaciones de TI para desbloquear su potencial y crea un modelo para una mejor colaboración e innovación.

HERRAMIENTA DE PRUEBAS AUTOMATIZADAS

Además de las pruebas unitarias, para garantizar la excelencia del entregable, es necesaria la verificación por pruebas funcionales, y para la Cuenta Única, siguiendo los conceptos de entrega continua, se eligió realizar de forma automatizada.

La creación de las secuencias de comandos de pruebas se producirá paralelamente al desarrollo de código. Después de que se construya el *build*, Jenkins ejecutará las pruebas configuradas y sólo implementará cuando todos los escenarios tengan el estatus igual al aprobado.

Para el desarrollo de los *scripts* de pruebas se utilizarán la siguiente tecnología:

- Framework Rest-Assured: Es un *framework* de *software* libre escrito en Java, que utiliza Hamcrest y JUnit para hacer las validaciones en respuestas. El lenguaje utilizado es simplificado y de fácil aprendizaje, además de permitir estructurar los scripts en el patrón de escritura ágil, utilizando Given, When y Then.

HERRAMIENTA DE GESTIÓN DE CONFIGURACIÓN

Para facilitar los cambios de los parámetros en cada ambiente en el momento del *deploy*, se utilizará la herramienta Consul.

HERRAMIENTA DE APLICACIÓN DE *SCRIPTS* EN BASE DE DATOS

Pensando en facilitar la ejecución de secuencias de comandos en el despliegue de cada versión, utilizaremos la aplicación Flyway. Esta es una herramienta de migración de bases de datos de código abierto, que favorece la simplicidad y la convención sobre la configuración.

DESPLIEGUE CONTINUO

Esta etapa del proceso aborda la cuestión de la automatización del proceso de publicación en el ambiente de producción, tan pronto como esté seguro de que el código ha pasado por todas las pruebas y está listo para ser publicado. La idea es entregar valor al negocio lo más rápido posible y no acumular código nuevo en *stage*. Es decir, si el código pasó por el proceso de integración que es responsable por pruebas de integración o pruebas unitarias, y también fue avanzando en el proceso de entrega con pruebas manuales, visuales y de comportamiento, entonces entra la fase de despliegue que es responsable de publicar el código en producción de forma automatizada.

En esta fase del proceso se asume que todas las pruebas se han realizado, y sólo se tratará la cuestión de publicación automatizada en el ambiente de producción.

Se supone que el ambiente de producción es estable, de que todo ha sido probado y que este proceso debe ser simple como apretar un botón y que debe ser posible por cualquier persona.

Y, para el proyecto Cuenta Única, quien realizará este despliegue en producción es el Jenkins. Después de que todas las pruebas unitarias, automatizadas y funcionales sean aprobadas, el Gestor / P.O. del proyecto, de acuerdo con su planificación, agenda una fecha para la nueva versión entrar en producción.

MODELO DE INTEGRACIÓN, ENTREGA Y DESPLIEGUE CONTINUO

Con base en el ítem anterior, a la continuación es presentado el modelo de integración, entrega y despliegue para el proyecto Cuenta Única:

- La estructura del proyecto está en la nube.
- Para gestión de configuración utilizaremos la herramienta Consul.
- Para aplicar los *scripts* de base de datos en cada implantación utilizaremos el Flyway.
- Para gestión de *build*, utilizaremos el Gradle.
- Los papeles para cada equipo son:
- EQUIPO DE GESTÓN:
    - GERENTE DEL PROYECTO: Responsable por las actividades de gerencia del equipo, haciendo el control de estatus y actuando como facilitador para los demás equipos.
    - P.O.(*PRODUCT OWNER*): Persona que posee el conocimiento del negocio y es responsable de diseminarlo entre el equipo.
- EQUIPO DE INFRAESTRUCTURA:
    - DBA: Responsable de mantenimiento de la base de datos.
    - PROFESIONALES DE INFRAESTRUCTURA: Equipo responsable de desarrollar y ejecutar los scripts para crear y mantener la infraestructura del proyecto y supervisar los recursos.
- EQUIPO DE DESARROLLO:

- LÍDER DE EQUIPO: Profesional con más experiencia y conocimiento de la etapa de desarrollo de software. Es responsable de hacer la revisión de código y ayudar a los desarrolladores en sus dificultades técnicas.
- DESARROLLADORES: Responsables del desarrollo del *software*.
- ANALISTAS DE PRUEBAS: Responsables de la calidad del *software*. Además de crear las pruebas automatizadas, deben asegurarse de que el *software* esté de acuerdo con el documento de especificación.

Con esto podemos explicar el *pipeline*:

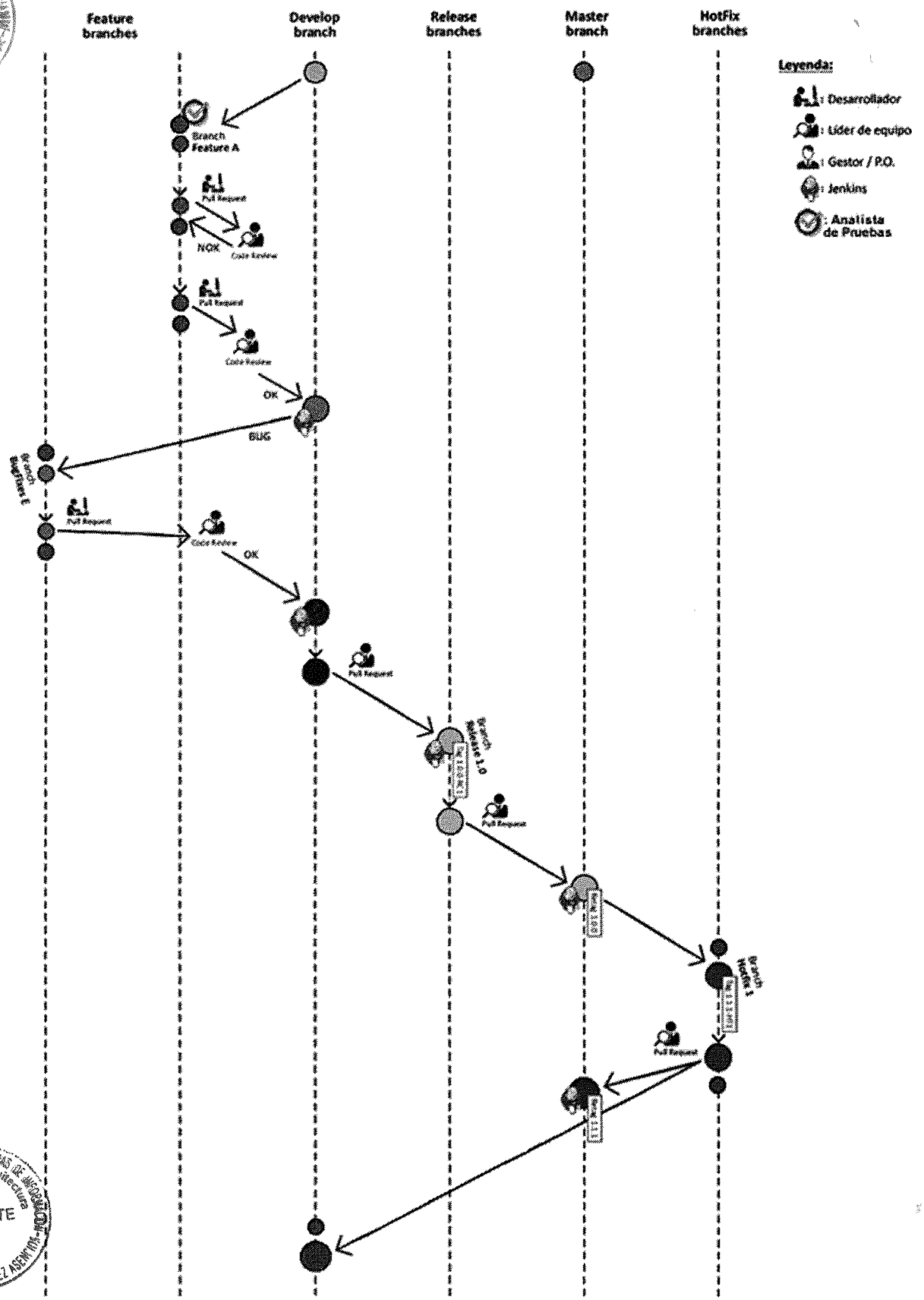

Ilustración 2 - Modelo de integración, entrega y despliegue continuo

1) Para inicio del desarrollo de una funcionalidad, es creada un *branch*, del tipo *Feature*, con base del *branch* Develop.

2) Después que el desarrollo es finalizado, el desarrollador ejecuta un *Pull Request* para enviar el código para revisión.

3) El líder del equipo revisa el código, en GitLab.

4) Si reprobado, el código es enviado, a través de GitLab, para que el desarrollador lo ajuste.

5) Cuando el código es ajustado, el desarrollador ejecuta un nuevo *Pull Request* para enviar el código para revisión.

6) El líder del equipo hace la revisión del código.

7) Si aprobado, es enviado, a través de la interface gráfica del GitLab, para merge en el branch Develop.

8) Despliegue del código del *branch Develop* en ambiente de pruebas, conforme imagen en la continuación. Todas las acciones ejecutadas en Jenkins, están configuradas en JenkinsFile.

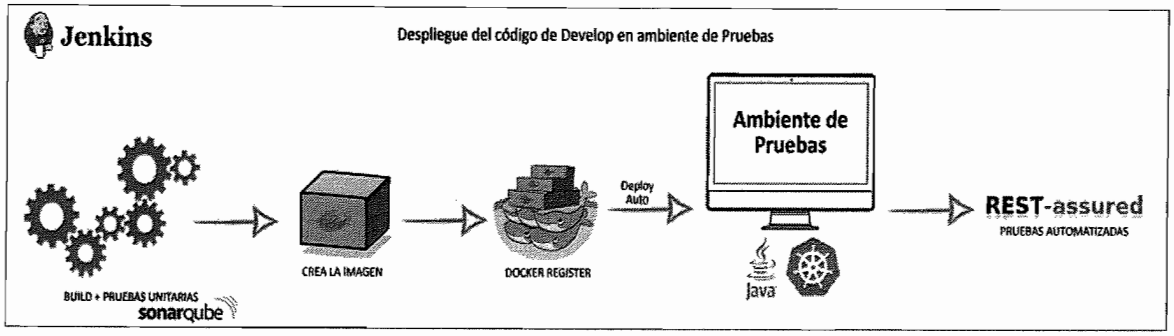

Ilustración 3- Despliegue del código de Develop en ambiente de Pruebas

a) Jenkins reconoce que hay un nuevo código en el *branch Develop*, hace *merge* con la versión actual y ejecuta el *build* y las pruebas unitarias, el resultado y la cobertura de las pruebas unitarias pueden ser visualizadas por el SonarQube.

b) Jenkins crea la imagen Docker (Esta imagen ya está pre-configurada en *Docker Repository*) con el archivo .JAR generado por el *build*, también vía configuración en el JenkinsFile.

c) Almacena el Docker creado en el *Docker Registry.*

d) Implanta automáticamente en el ambiente de pruebas.

e) Con la aplicación implementada, se realizan las pruebas automatizadas. Si es encontrado un *bug* en la ejecución de las pruebas automatizadas, es enviado un mensaje en el log de implementación y el desarrollador debe corregirlo. La versión entra en ambiente de pruebas así mismo, pero el líder de equipo solo debe liberar para la próxima etapa, después que el *bug* sea corregido.

   (1) El desarrollador crea un nuevo *branch feature* para la corrección.

   (2) Cuando la corrección es finalizada, el desarrollador ejecuta un *Pull Request* para enviar el código para revisión.

   (3) El líder del equipo revisa el código, en GitLab.

   (4) Si aprobado, es enviado, a través de la *interface* gráfica del GitLab, para *merge* en el *Develop* (que ya está con la última versión).

   (5) De acuerdo con el GitFlow, después que un *branch Feature* es enviada para el *branch Develop,* es excluido.

   (6) Igual al paso 9.a., Jenkins reconoce que hay un nuevo código en el *branch Develop* y ejecuta el despliegue de la versión en ambiente de pruebas, haciendo *merge* con la versión actual. Después del despliegue, son ejecutadas las pruebas automatizadas nuevamente.

9) Cuando tenemos un número de funciones listas para generar una versión, de acuerdo con la planificación del Gestor/P.O., el líder de equipo ejecuta un *Pull Request* para crear un nuevo *branch Release* con el código que está en el *branch Develop*.

10) Despliegue del código del *branch Develop* en ambiente de pruebas, conforme imagen en la continuación.

Todas las acciones ejecutadas en Jenkins, están configuradas en JenkinsFile.

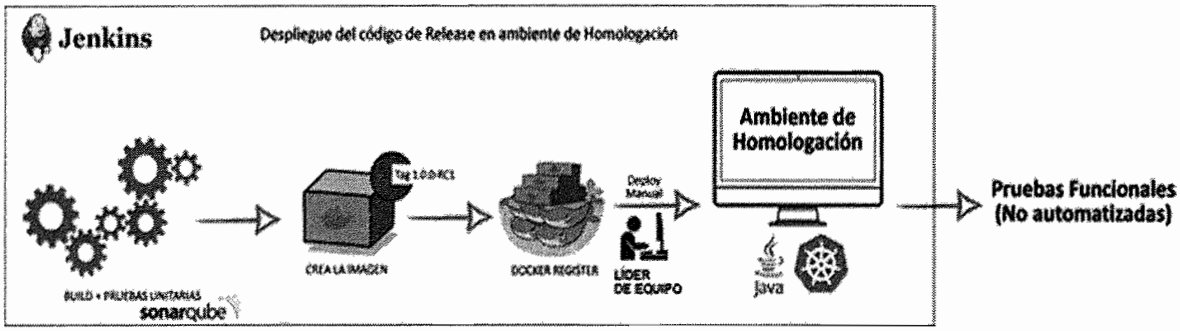

Ilustración 4- Despliegue del código de Release en ambiente de Homologación

a) Jenkins reconoce que hay un nuevo código en el *branch Release* e inicia la ejecución del *build* y las pruebas unitarias.

b) Jenkins crea la imagen Docker etiquetada con la misma versión que está en la Release, para acompañamiento, con el archivo .JAR generado por el *build*. Ese control es necesario para verificar el "nombre" de la versión que está con determinada funcionalidad y fase.

c) Almacena el Docker creado en el *Docker Registry*.

d) La implantación en ambiente de Homologación no es automática, es ejecutada por el Líder de Equipo de acuerdo con la planificación del Gestor/P.O.

e) Después del deploy del Docker en el container, se realizan pruebas funcionales (no automatizadas) y pruebas no funcionales. Si se encuentra alguna falla, el desarrollador debe ser informado y realizar la corrección.

11) Cuando todas las pruebas fueron realizadas y de acuerdo con la planificación del Gestor/P.O., el líder de equipo ejecuta un *Pull Request* para enviar el código al *branch Master*.

12) Jenkins reconoce que hay un nuevo código en el *branch Master*, realiza el *merge* con la versión actual, retag la versión existente en el *branch Release* y ejecuta el despliegue de la versión en ambiente de producción, conforme imagen en la continuación.

Todas las acciones ejecutadas en Jenkins, están configuradas en JenkinsFile.

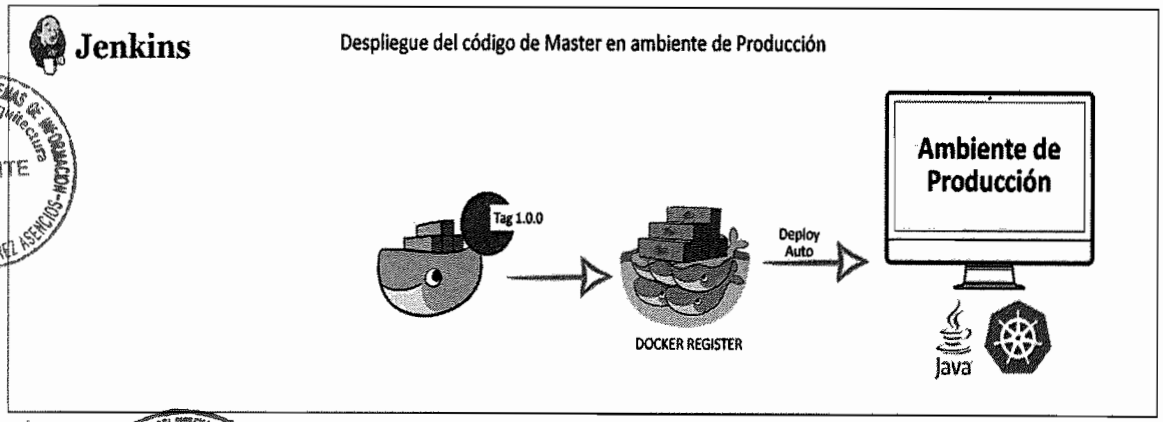

Ilustración 5- Despliegue del código de Master en ambiente de Producción

13) De acuerdo con el GitFlow, después que un *branch Release* es enviada para el *branch Master*, es excluida.

14) Aplicación antes de la nueva versión en ambiente de producción (branch Master).

Flujo de Hotfix:

1) El código es enviado del *branch Master* para la un nuevo *branch Hotfix*.
2) Recebe una nueva *Tag* para ser controlada.
3) El *bug* es corregido y el líder de equipo ejecuta un *Pull Request* para enviar el código al *branch Master* y actualizar el *branch Develop*.
4) Despliegue del código del *branch Hotfix* en ambiente de producción, conforme imagen en la continuación.

Todas las acciones ejecutadas en Jenkins, están configuradas en JenkinsFile.

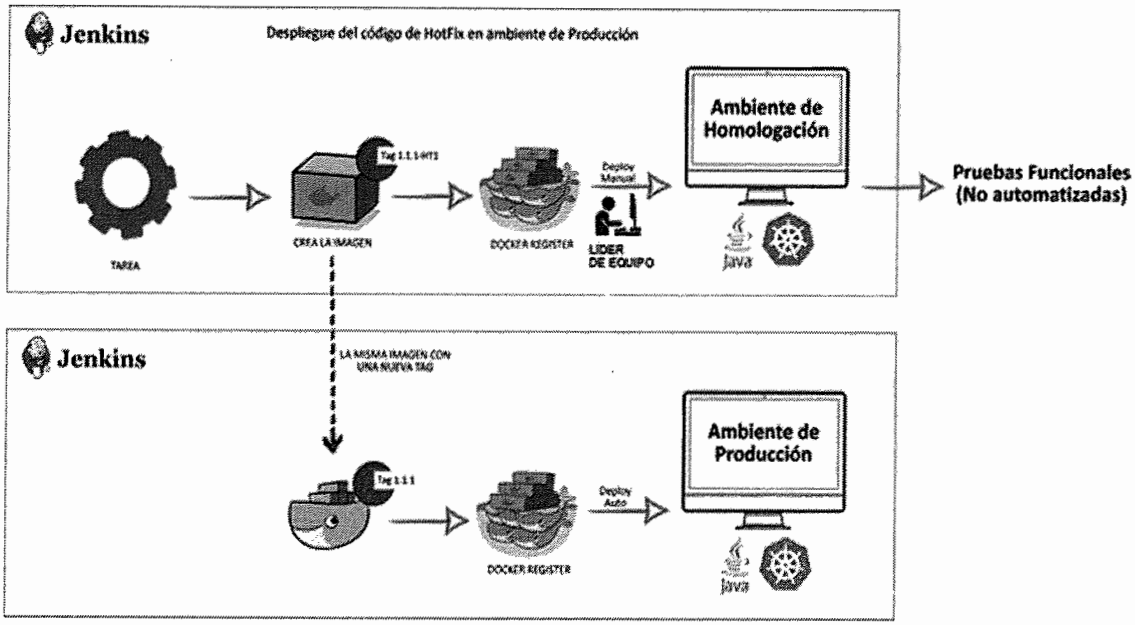

Ilustración 6- Despliegue del código de HotFix en ambiente de Producción

a) Jenkins reconoce que hay un nuevo código en el *branch Master* y ejecuta la tarea para crear un nuevo Docker.
b) Jenkins crea el Docker con el archivo .JAR generado y agrega el *tag* para acompañamiento.
c) Almacena el Docker creado en el *Docker Registry*.
d) La implantación en ambiente de Homologación no es automática, es ejecutada por el Líder de Equipo de acuerdo la planificación de Gestor.
e) Con el servicio implementado, se realizan las pruebas funcionales, no automatizadas. Si se encuentra alguna falla, el desarrollador debe ser informado y realizar la corrección.
f) Después de las pruebas funcionales, el Gestor/P.O. autoriza la implantación en producción, donde el Jenkins localiza el Docker generado en el último *branch Hotfix*, genera una nueva tag para acompañamiento y es implantada en producción automáticamente. Además de esta implantación, el código también es enviado para el *branch Develop* para actualizar las próximas versiones.

5) El *branch Develop* es actualizada para tener la última versión para el próximo *merge*.

Anexo 16: Modelo de Seguridad de Datos

Perfiles de Seguridad

Los perfiles de seguridad permiten representar las restricciones de acceso de los usuarios de una solución de manera agrupada y organizada. Esa definición es realizada durante la analice de negocio y ofrece suporte a la arquitectura de datos. Además de los perfiles de negocio descritos en las historias de usuario, hay algunas acciones que son realizadas por los componentes de software de la solución y adicionalmente, se definirán perfiles de acceso al personal técnico de la Intendencia Nacionalde Sistemas de información. En este documento serán referenciados los siguientes perfiles de acceso de componentes de software:

Perfil	Descripción
Gestor de cuenta (Software)	✓ Representa los componentes de software de la solución propuesta, o sea, los procesos automáticos que no son, necesariamente iniciados por una persona;
Sistema Legacy	✓ Representa los mecanismos de integración con los sistemas legacy de la SUNAT a partir de los cuales serán obtenidas informaciones para la cuenta única. ✓ La información obtenida por los componentes de software de integración debe poder ser grabada en las entidades del agente extractor de datos.

Contextualización

El documento de seguridad de datos está organizado para mostrar la relación entre los perfiles de usuarios y las respectivas entidades de negocio de la solución de Cuenta Única. Cada sección presenta un diagrama de seguridad, en notación ArchiMate, y una matriz de seguridad que indica cuales son las permisiones de cada perfil de manera consistente con el informe de clasificación de datos.

AED – Agente De Extracción De Datos

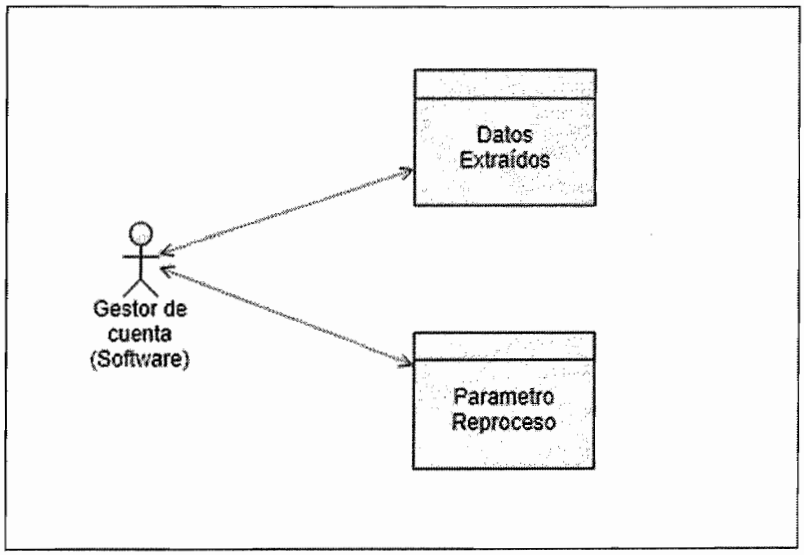

Figura 47 – Diagrama de Seguridad AED (Agente Extractor de Datos)

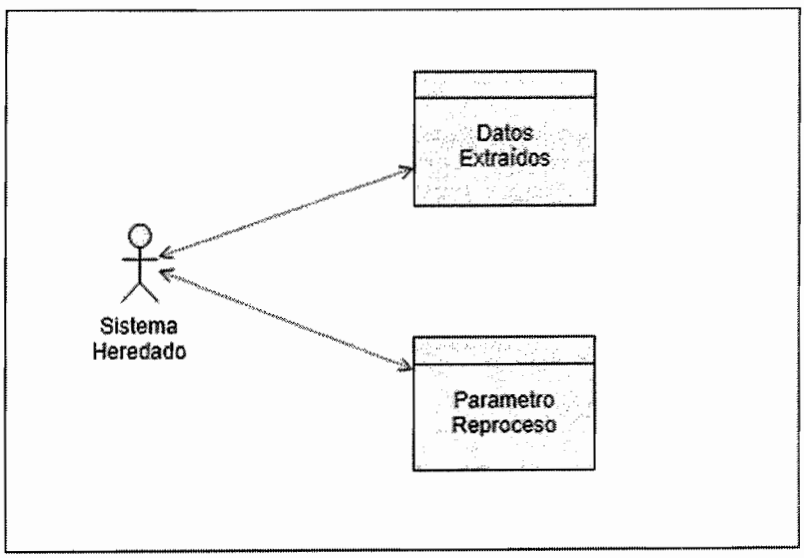

Figura 48 – Diagrama de Seguridad AED (Sistema Legacy)

	Gestor de cuenta (Software)				Sistema Legacy			
Entidad	C	R	U	D	C	R	U	D
DatosExtraidos		X	X	X	X	X	X	
ParametroReproceso		X	X	X	X	X	X	

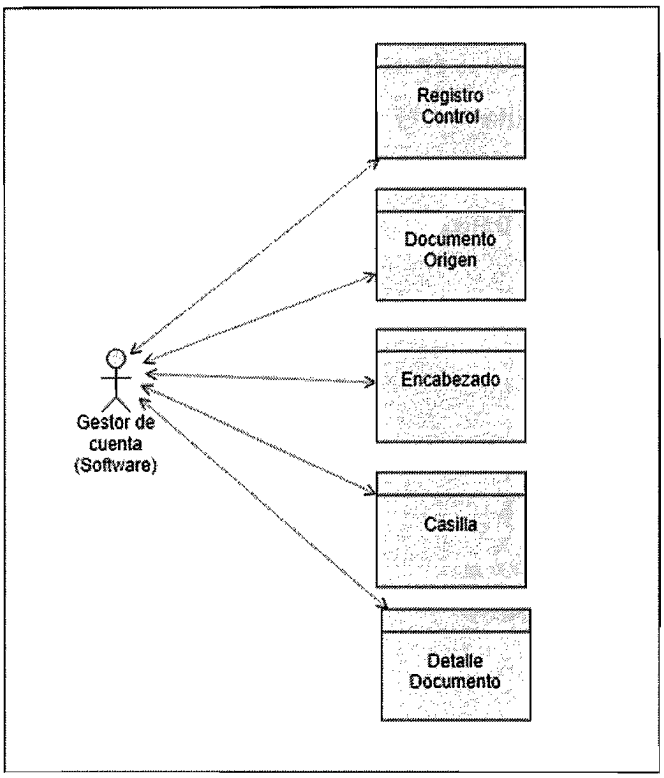

Figura 49 – Diagrama de Seguridad Solicitud

	Gestor de cuenta (Software)			
Entidad	C	R	U	D
RegistroControl	X	X	X	
DocumentoOrigen	X	X	X	
Encabezado	X	X	X	
Casilla	X	X	X	
DetalleDocumento	X	X	X	

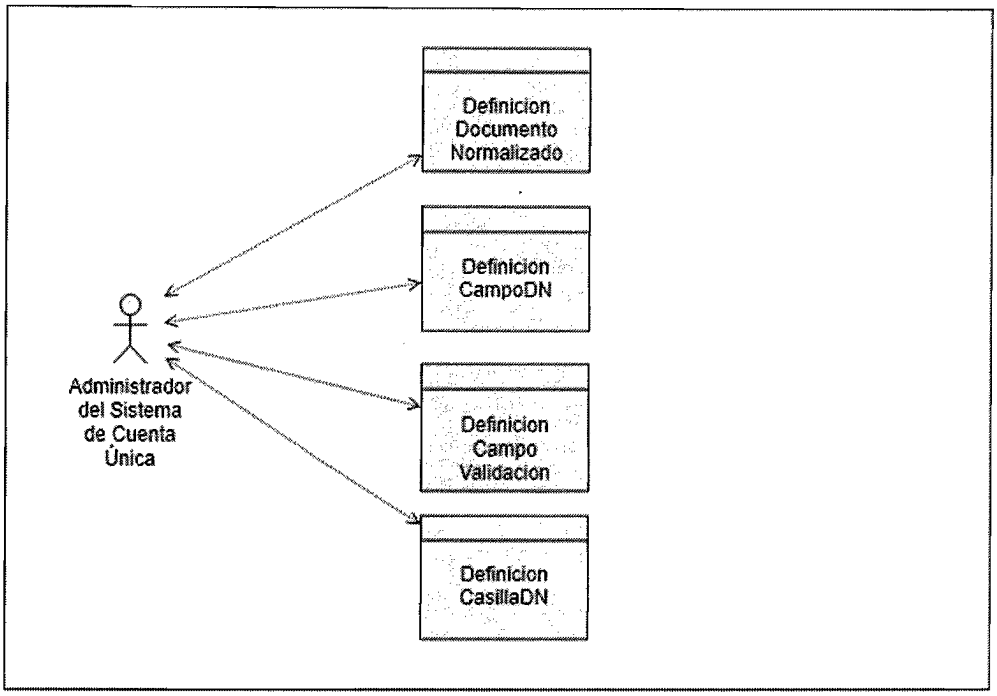

Figura 50 - Diagrama de Seguridad Reglas de Normalización (Administrador del Sistema de Cuenta Única)

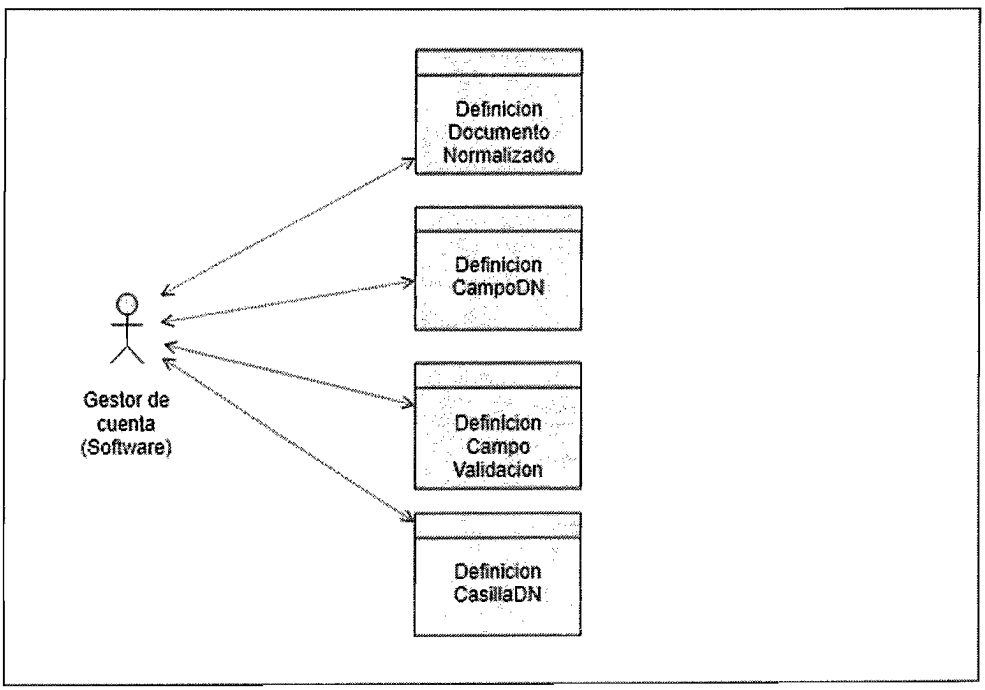

Figura 51 - Diagrama de Seguridad Reglas de Normalización (Gestor de Cuenta - Software)

Gestor de cuenta (Software)	Administrador del Sistema de Cuenta Única

Entidad	C	R	U	D	C	R	U	D
DefinicionDocumentoNormalizado		X			X	X	X	X
DefinicionCampoDN		X			X	X	X	X
DefinicionCampoValidacion		X			X	X	X	X
DefinicionCasillaDN		X			X	X	X	X

Reglas de Registro de Movimientos en Cuenta

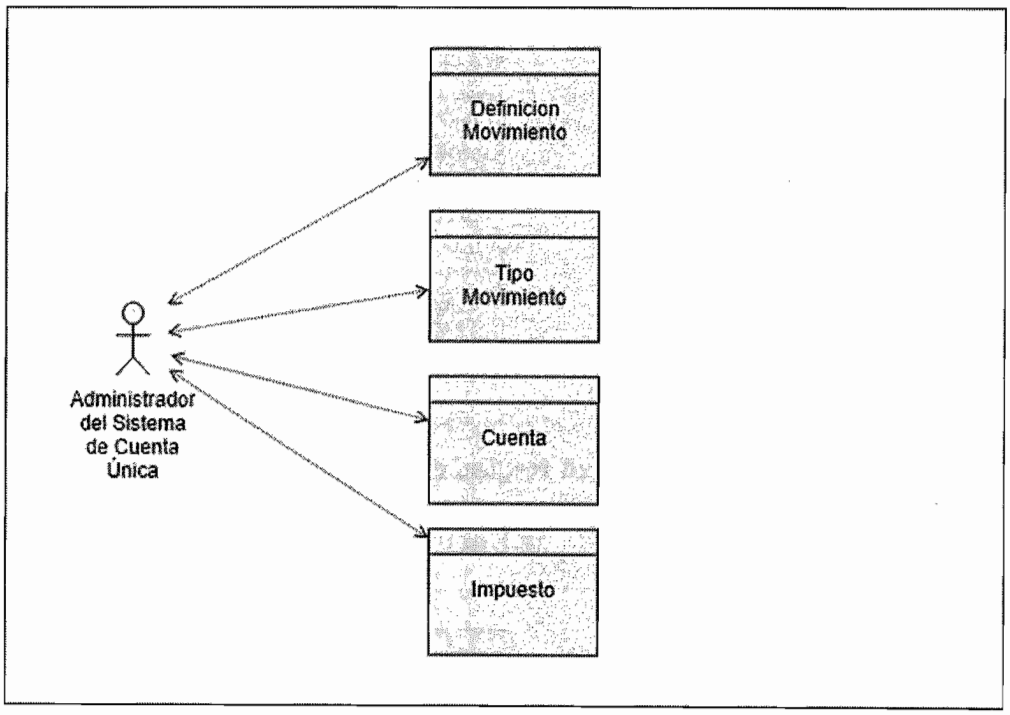

Figura 52 - Diagrama de Seguridad Reglas de Registro de Movimientos en Cuenta (Administrador del Sistema de Cuenta Única)

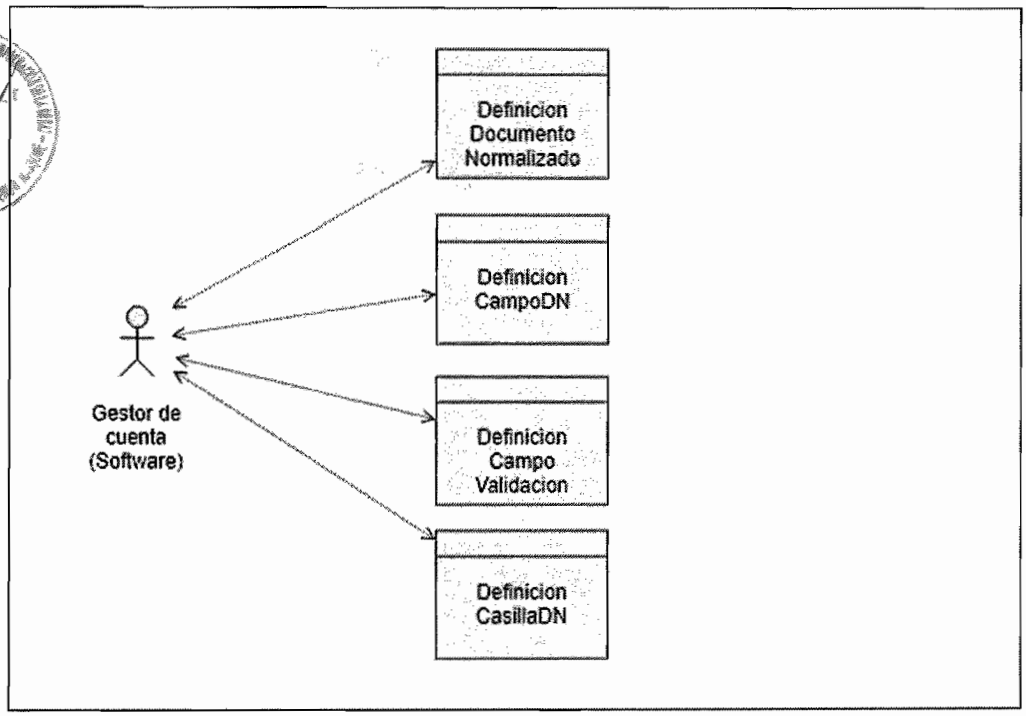

Figura 53 - Diagrama de Seguridad Reglas de Registro de Movimientos en Cuenta (Gestor de Cuenta - Software)

	Gestor de cuenta (Software)				Administrador del Sistema de Cuenta Única			
Entidad	C	R	U	D	C	R	U	D
DefinicionMovimiento		X			X	X	X	X
TipoMovimiento		X			X	X	X	X
Cuenta		X			X	X	X	X
Impuesto		X			X	X	X	X

Validación

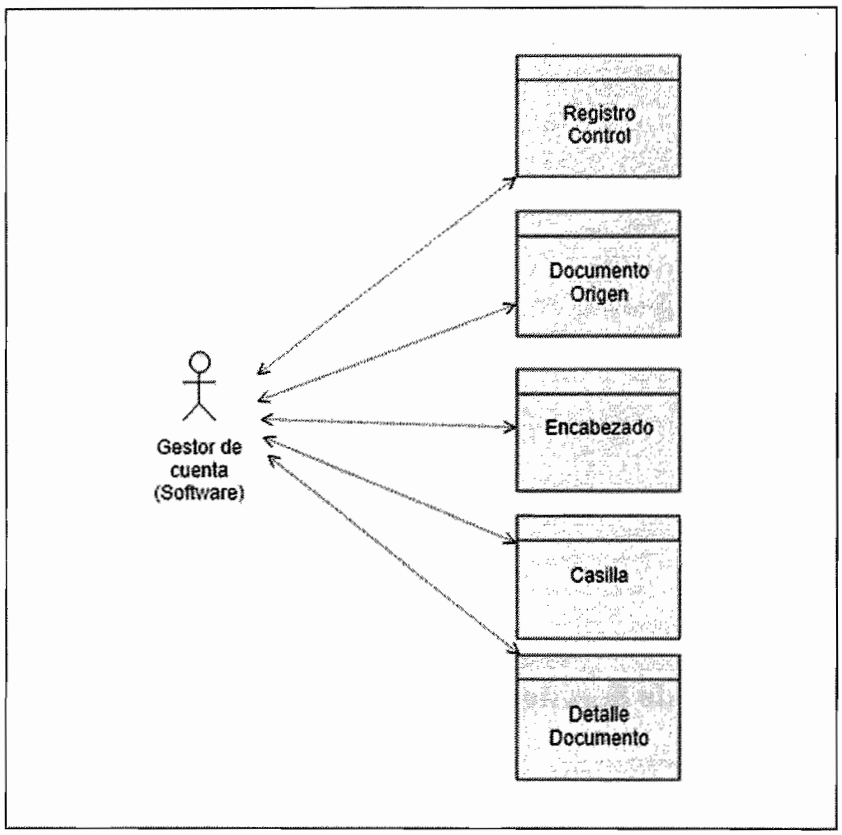

Figura 54 - Diagrama de Seguridad Validación

	Gestor de cuenta (Software)			
Entidad	C	R	U	D
RegistroControl		X		
DocumentoOrigen		X		
Encabezado		X		
Casilla		X		
DetalleDocumento		X		

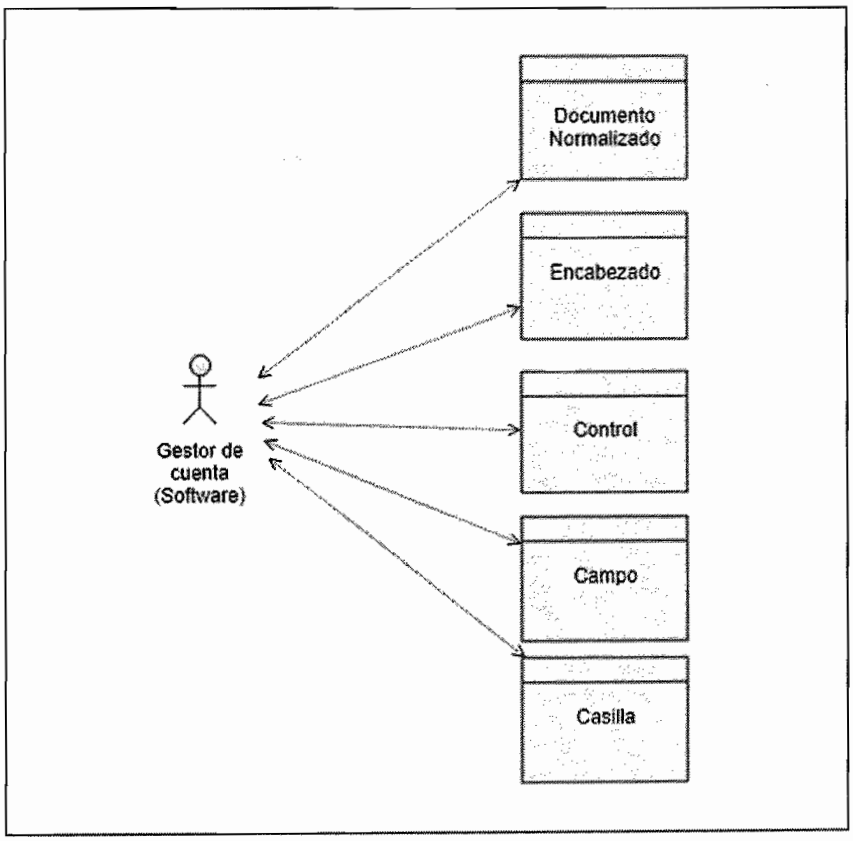

Figura 55 - Diagrama de Seguridad Motor DN

	Gestor de cuenta (Software)			
Entidad	C	R	U	D
DocumentoNormalizado	X	X	X	
Encabezado	X	X	X	
Control	X	X	X	
Campo	X	X	X	
Casilla	X	X	X	

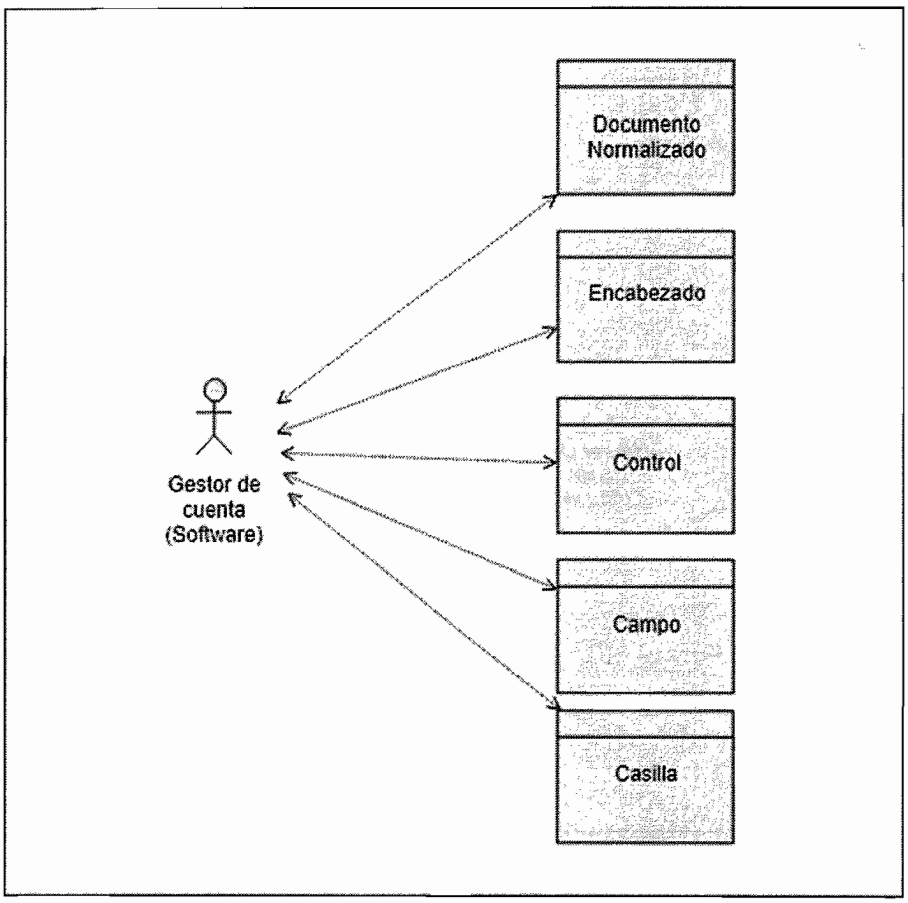

Figura 56 - Diagrama de Seguridad Cálculo DN

	Gestor de cuenta (Software)			
Entidad	C	R	U	D
DocumentoNormalizado		X	X	
Encabezado		X	X	
Control		X	X	
Campo		X	X	
Casilla		X	X	

Motor Lanzamiento

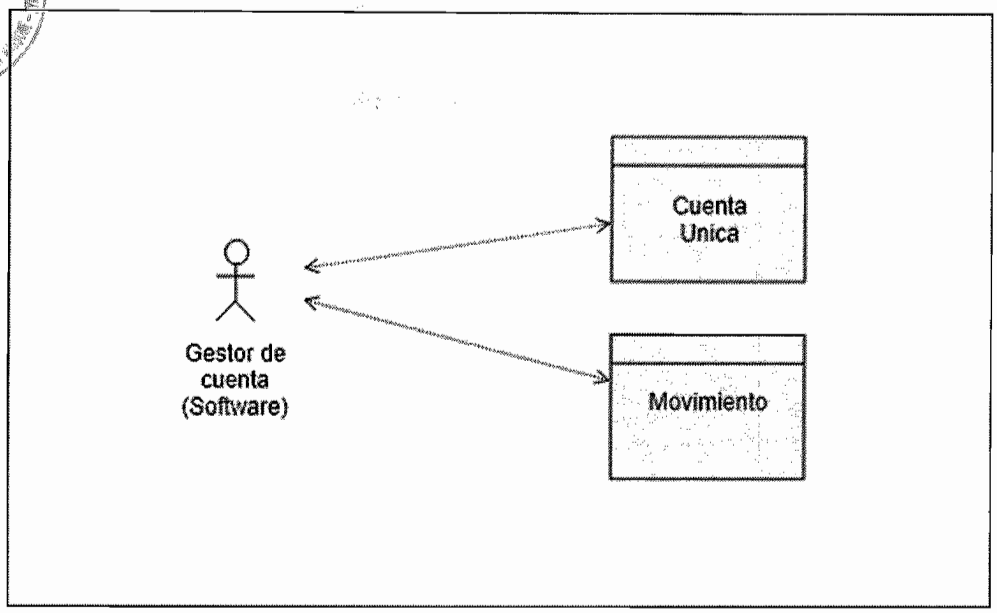

Figura 57 - Diagrama de Seguridad Motor Lanzamiento

	Gestor de cuenta (Software)			
Entidad	C	R	U	D
CuentaUnica	X	X	X	X
MovimientoCuenta	X	X	X	X

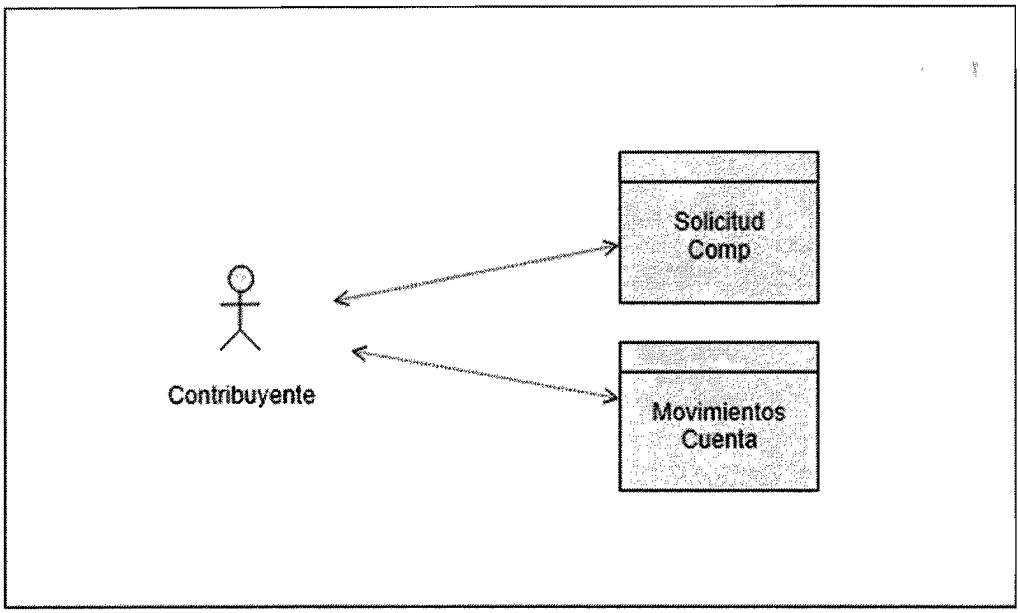

Figura 58 - Diagrama de Seguridad Compensaciones (Contribuyente)

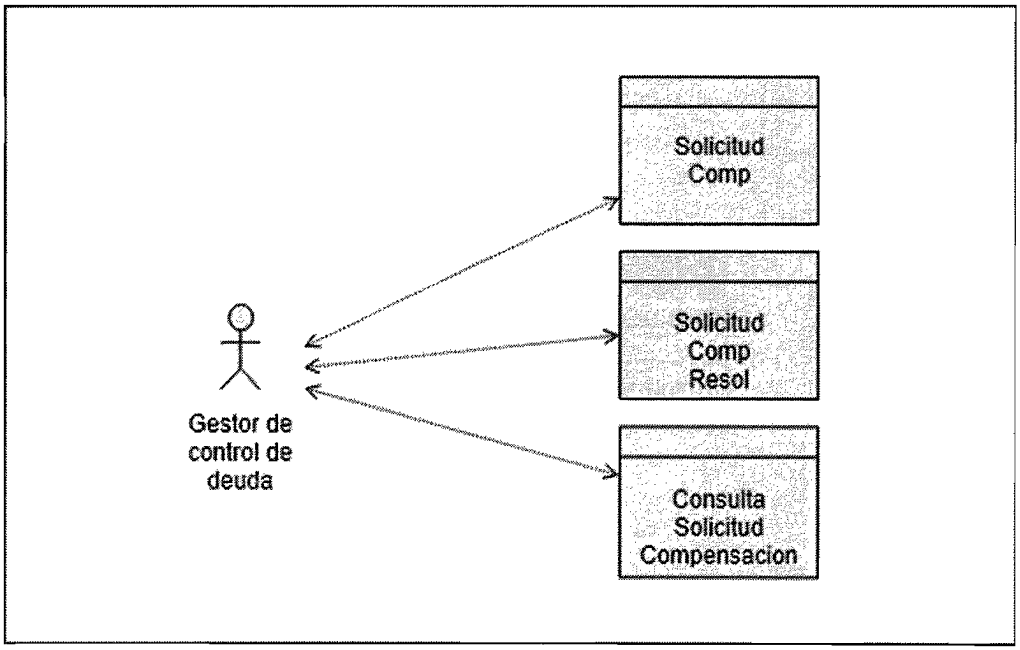

Figura 59 - Diagrama de Seguridad Compensaciones (Gestor de Control de Deuda)

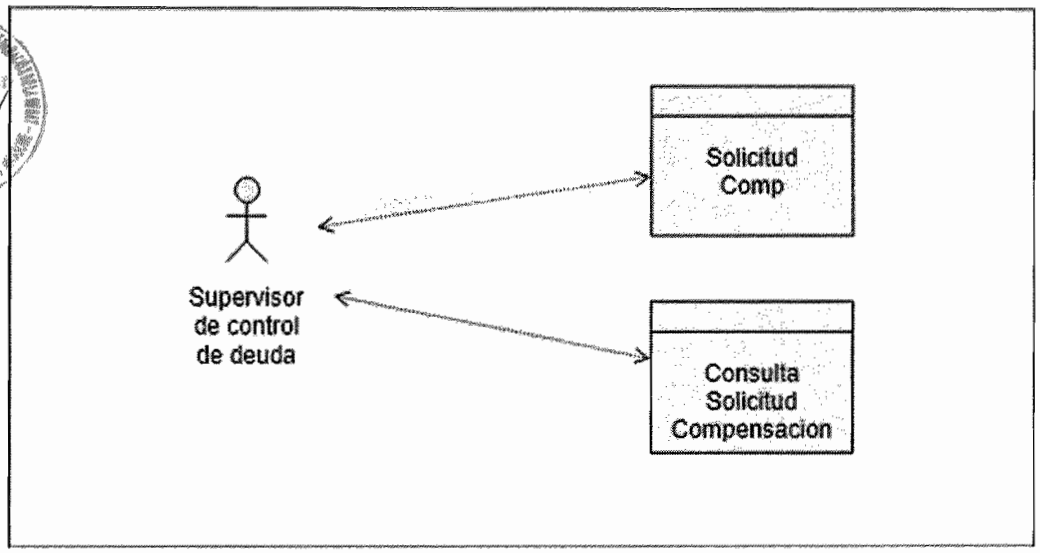

Figura 60 - Diagrama de Seguridad Compensaciones (Supervisor de Control de Deuda)

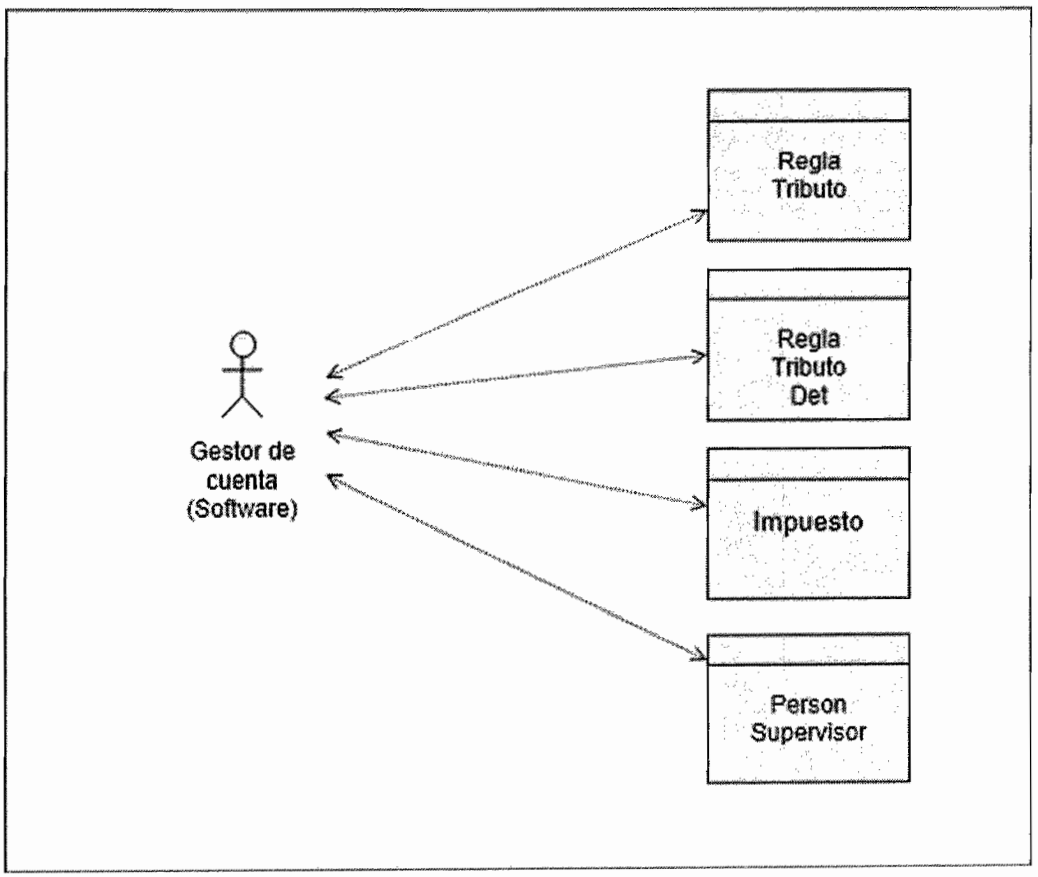

Figura 61 - Diagrama de Seguridad Compensaciones (Gestor de cuenta - Software)

	Contribuyente				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D

SolicitudComp	X								X	X	X				X	
SolicitudCompResol									X							
ReglaTributo						X										
ReglaTributoDet						X										
Impuesto						X										
Consulta Solicitud Compensacion										X				X		
PersonSupervisor						X										

EP002 – Suspensión

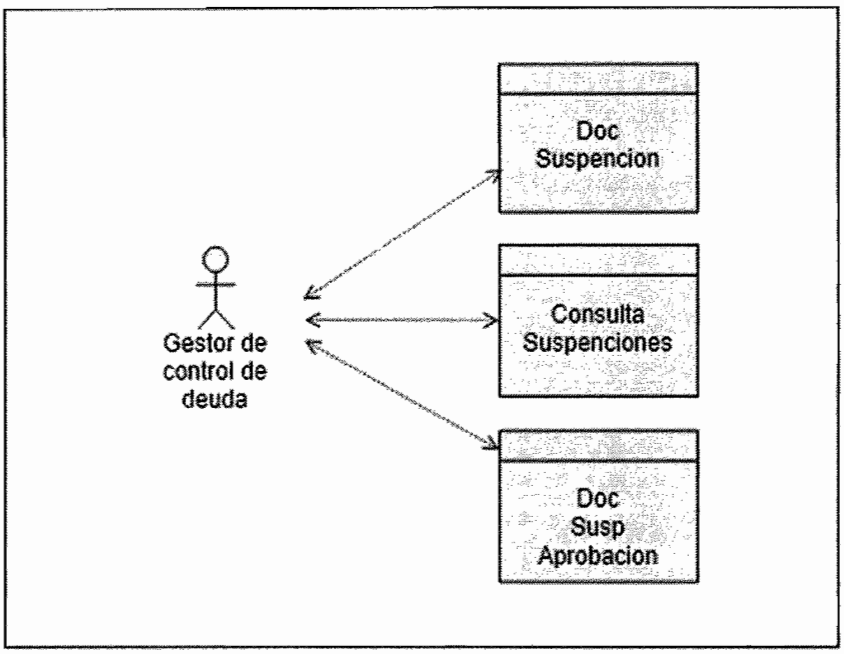

Figura 62 - Diagrama de Seguridad Suspensión (Gestor de Control de Deuda)

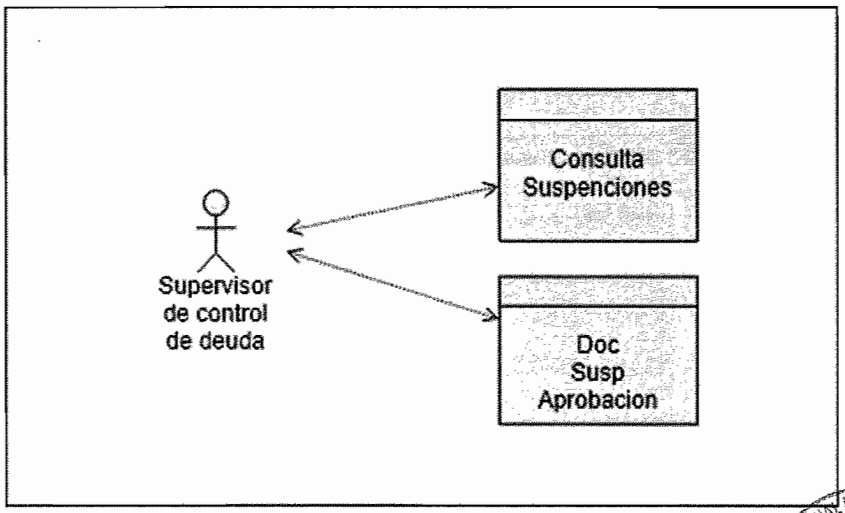

Figura 63 - Diagrama de Seguridad Suspensión (Supervisor de Control de Deuda)

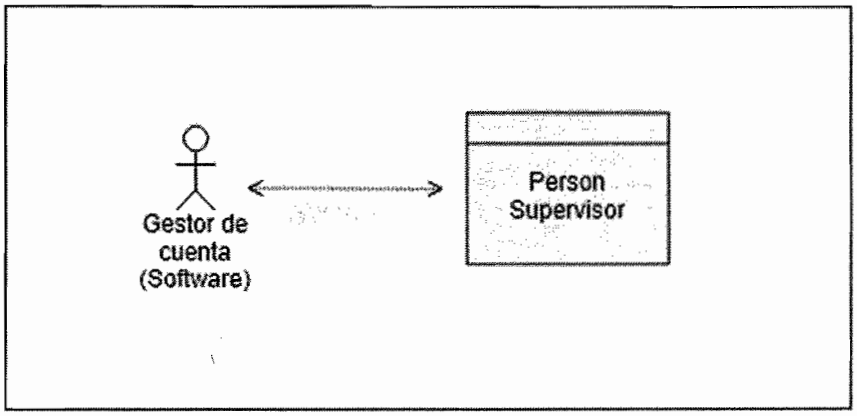

Figura 64 - Diagrama de Seguridad Suspensión (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D
DocSuspencion					X	X	X	X				
Consulta Suspenciones						X				X		
DocSuspAprobacion						X			X			
PersonSupervisor		X										

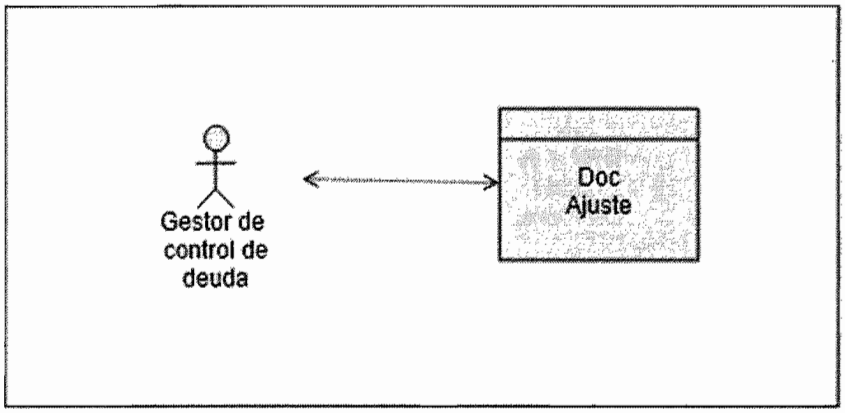

Figura 65 - Diagrama de Seguridad Ajustes (Gestor de Control de Deuda)

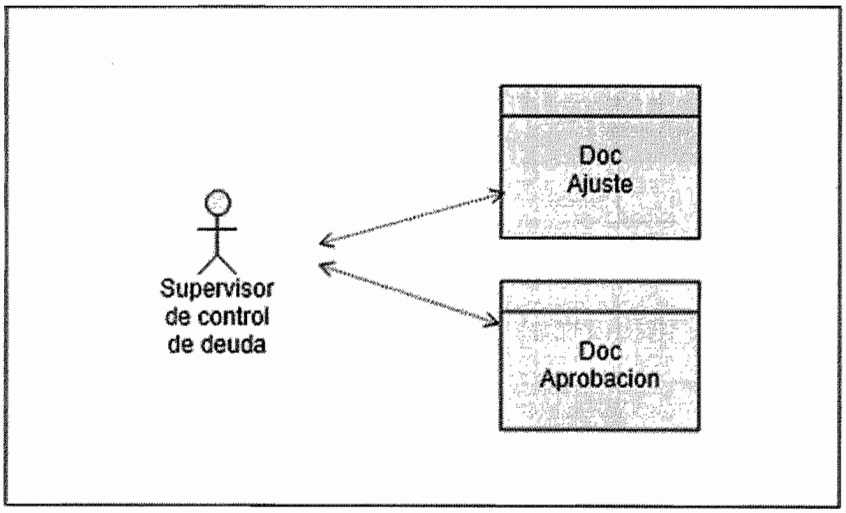

Figura 66 - Diagrama de Seguridad Ajustes (Supervisor de Control de Deuda)

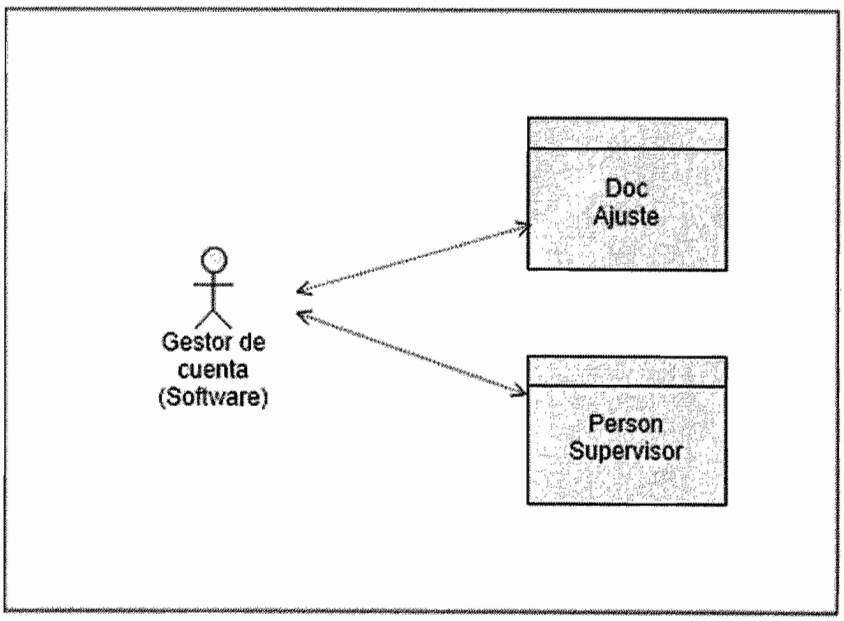

Figura 67 - Diagrama de Seguridad Ajustes (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D
DocAjuste		X			X	X	X	X		X	X	
DocAprobacion									X			
PersonSupervisor		X										

EP004 - Comunicación De Diferencias

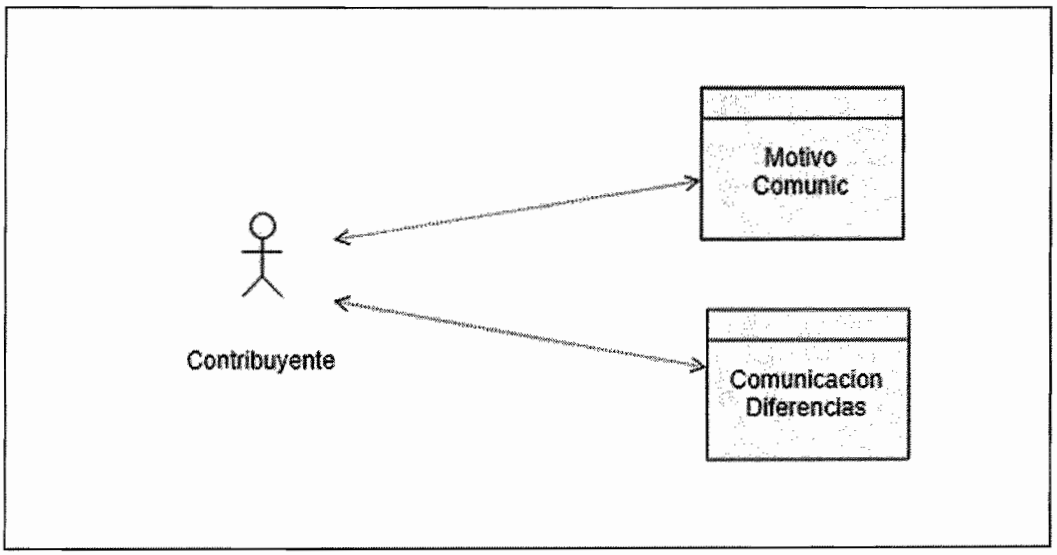

Figura 68 - Diagrama de Seguridad Comunicación de Diferencias (Contribuyente)

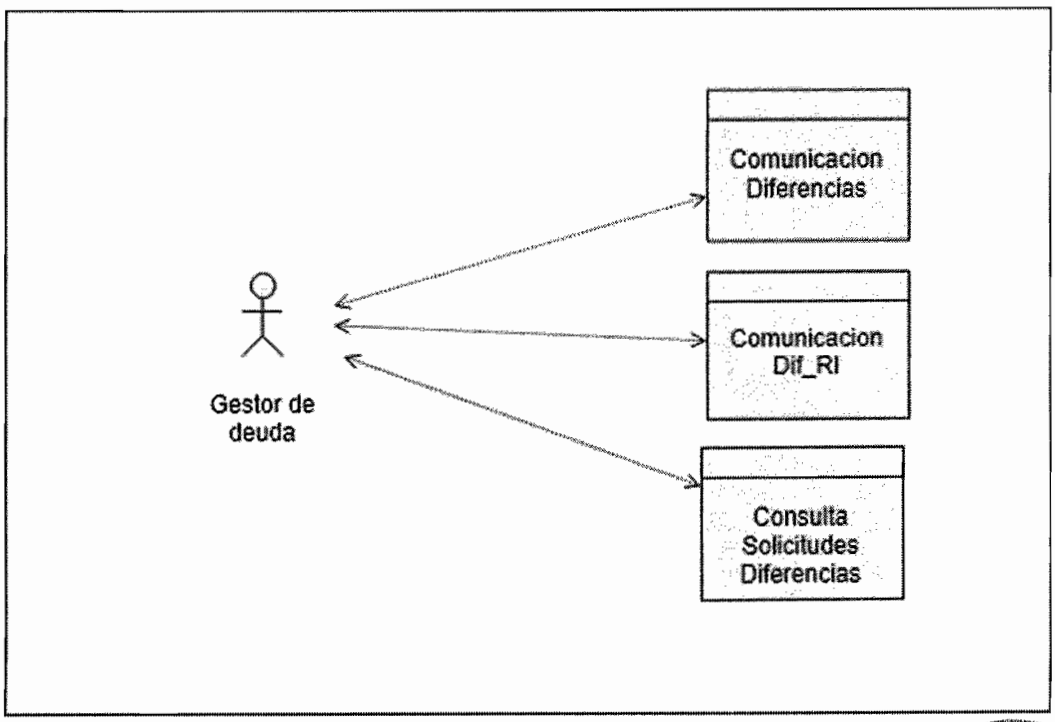

Figura 69 - Diagrama de Seguridad Comunicación de Diferencias (Gestor de Deuda)

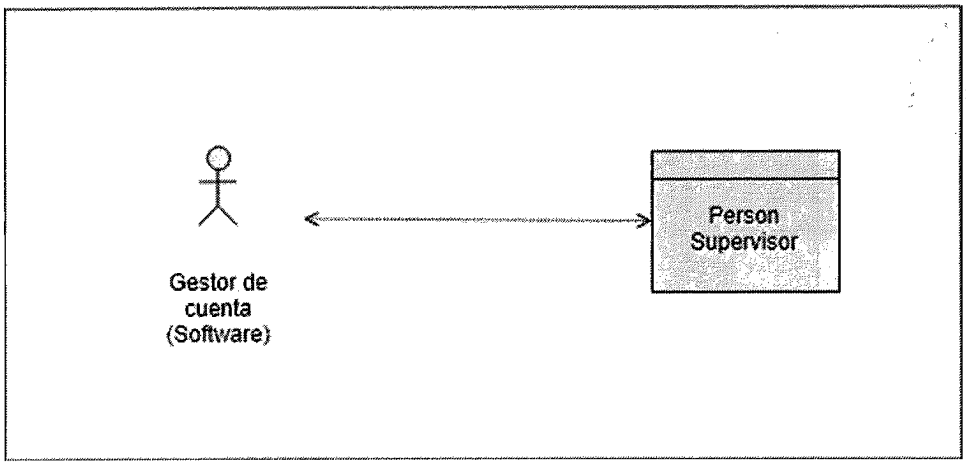

Figura 70 - Diagrama de Seguridad Comunicación de Diferencias (Gestor de Cuenta - Software)

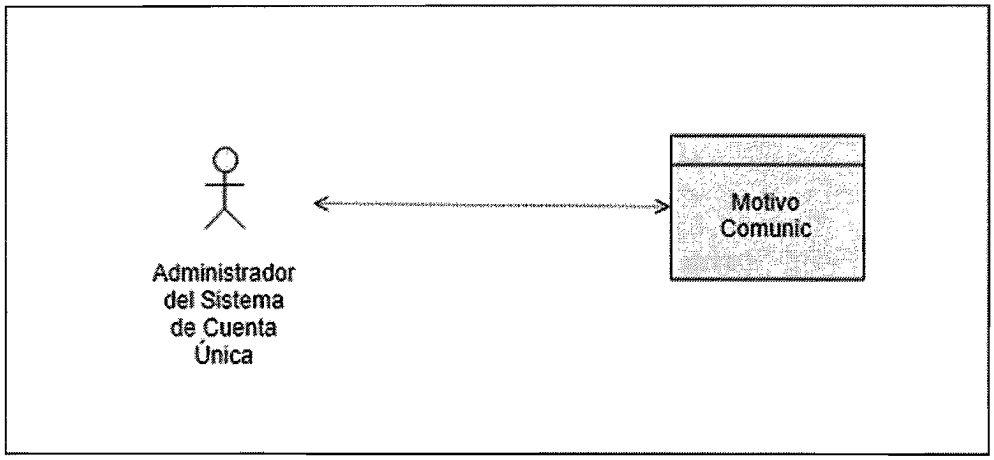

Figura 71 - Diagrama de Seguridad Comunicación de Diferencias (Administrador del Sistema de Cuenta Única)

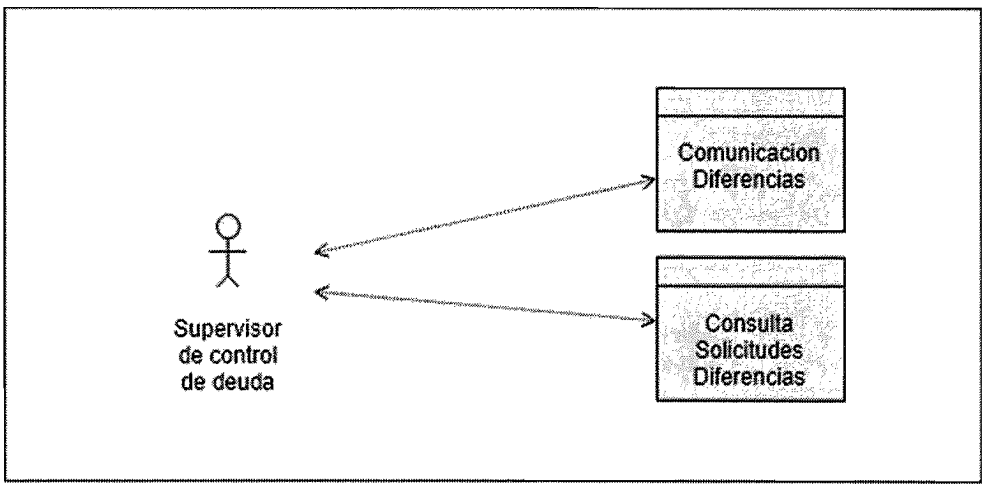

Figura 72 - Diagrama de Seguridad Comunicación de Diferencias (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Deuda				Gestor de Cuenta (Software)				Administrador del Sistema de Cuenta Única				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
MotivoComunic		X											X	X	X	X				
ComunicacionDiferencias	X	X				X	X											X	X	
ComunicacionDif_RI					X															
Consulta Solicitudes Diferencias						X												X		
PersonSupervisor										X										

EP005 - Reconocimiento de pago con error

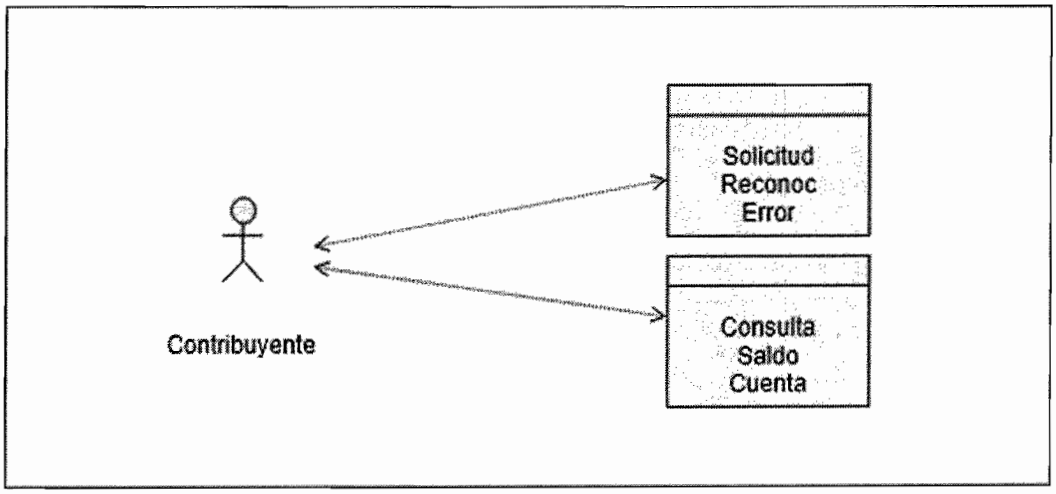

Figura 73 - Diagrama de Seguridad Pago con Error (Contribuyente)

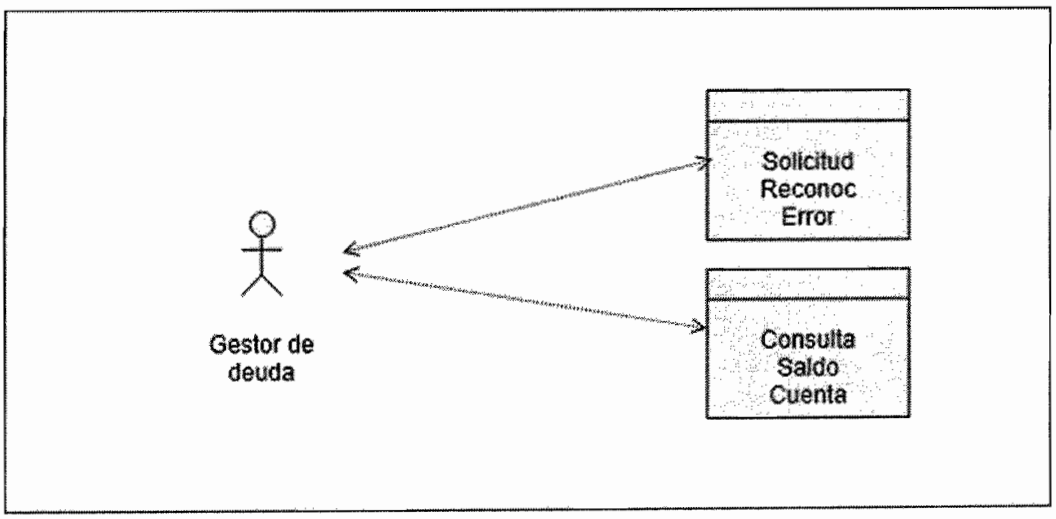

Figura 74 - Diagrama de Seguridad Pago con Error (Gestor de deuda)

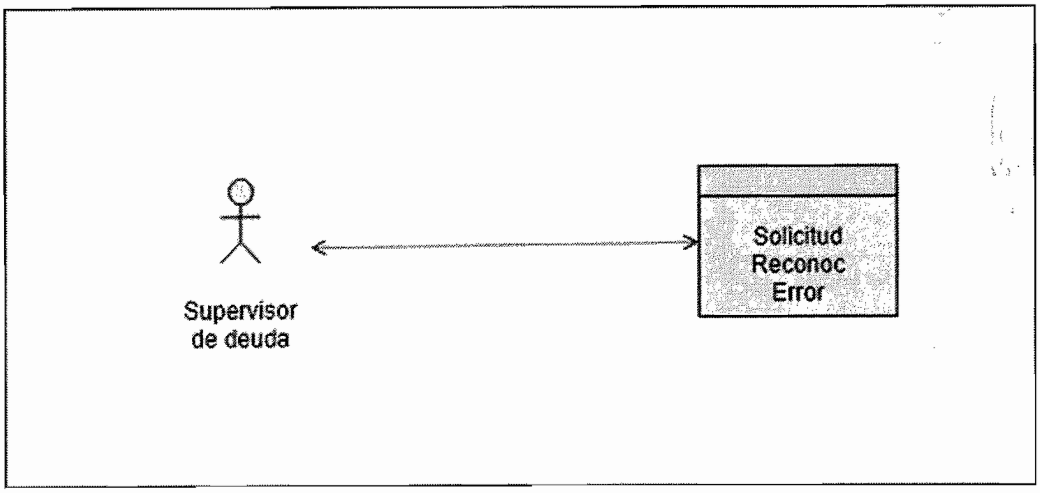

Figura 75 - Diagrama de Seguridad Pago con Error (Supervisor de deuda)

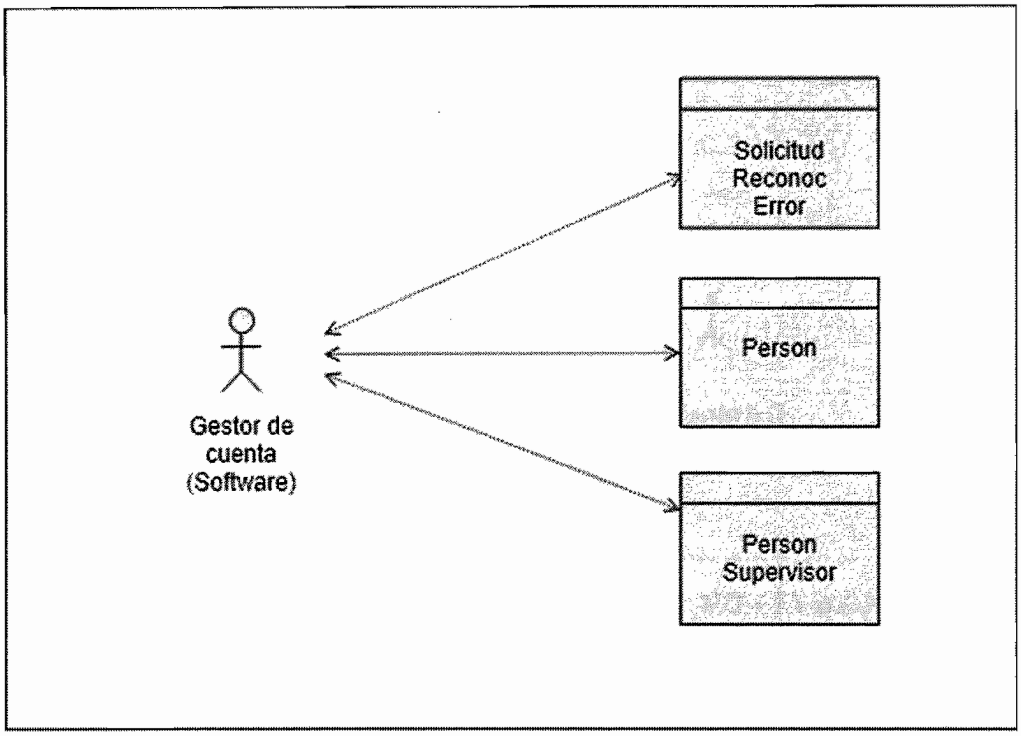

Figura 76 - Diagrama de Seguridad Pago con Error (Gestor de cuenta - Software)

	Contribuyente				Gestor de Deuda				Supervisor de Deuda				Gestor de Cuenta (Software)			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
SolicitudReconocError	X				X					X	X			X		
ConsultaSaldoCuenta		X				X										
Person														X		
PersonSupervisor														X		

EP006 - Modificación de datos

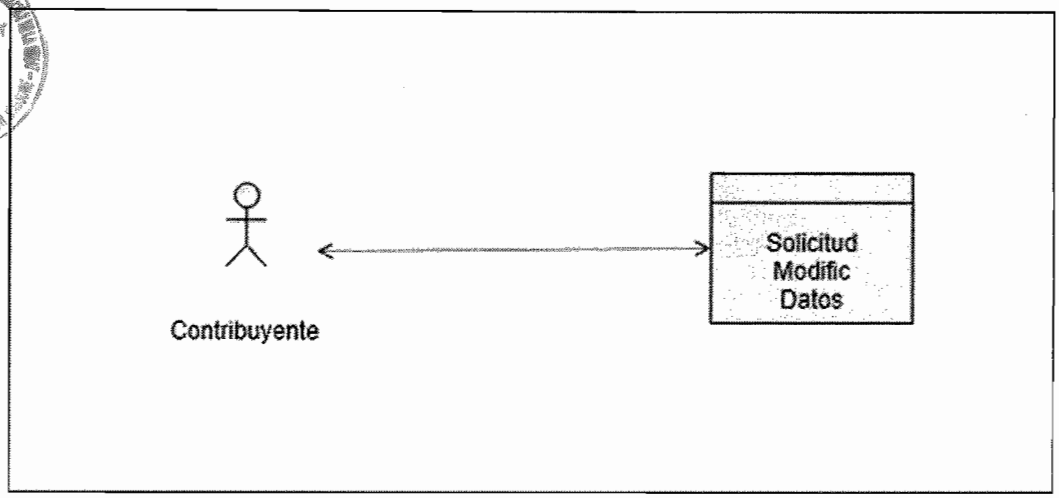

Figura 77 - Diagrama de Seguridad Modificación de Datos (Contribuyente)

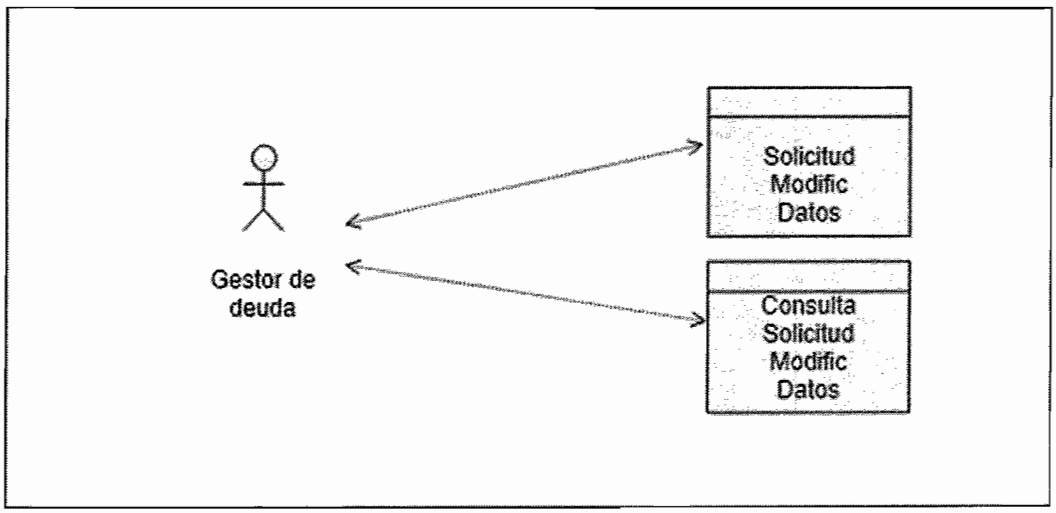

Figura 78 - Diagrama de Seguridad Modificación de Datos (Gestor de Deuda)

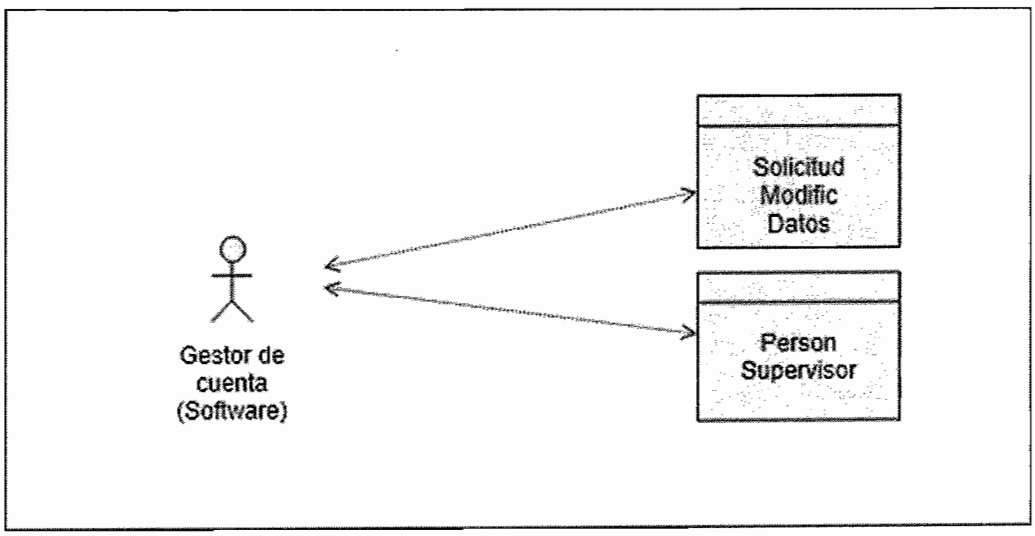

Figura 79 - Diagrama de Seguridad Modificación de Datos (Gestor de Cuenta - Software)

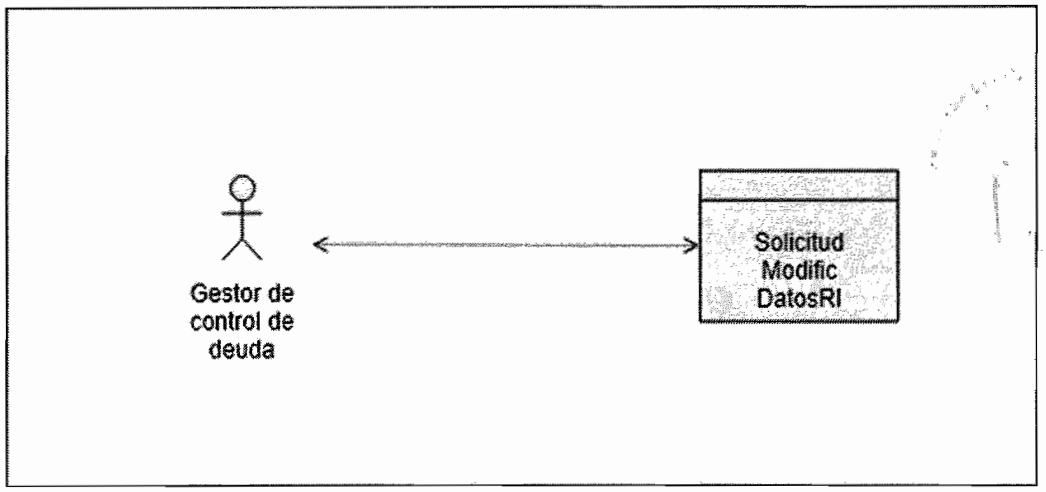

Figura 80 - Diagrama de Seguridad Modificación de Datos (Gestor de Control de Deuda)

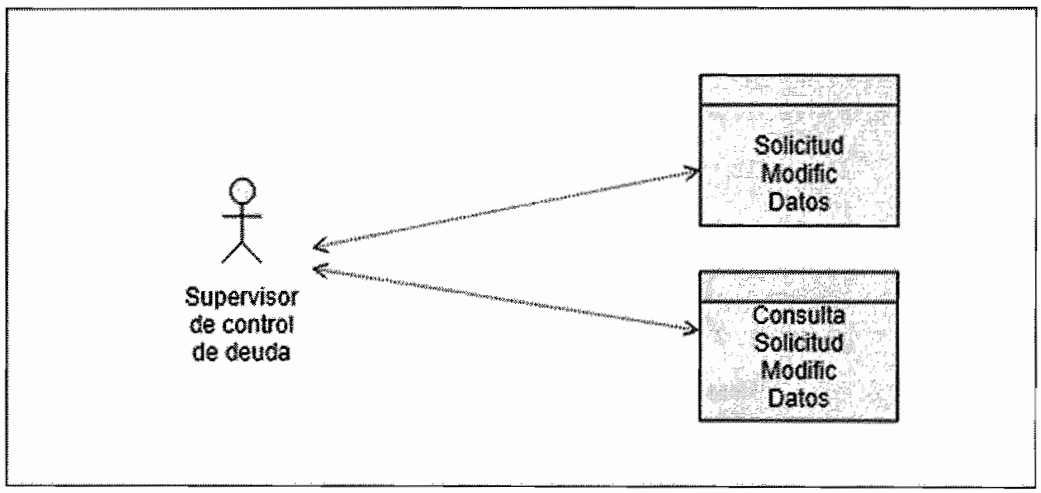

Figura 81 - Diagrama de Seguridad Modificación de Datos (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Deuda				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
SolicitudModificDatos	X				X					X								X	X	
Consulta Solicitud Modific Datos						X												X		
PersonSupervisor										X										
SolicitudModificDatosRI													X							

EP007 - Veredictos de rectificatoria

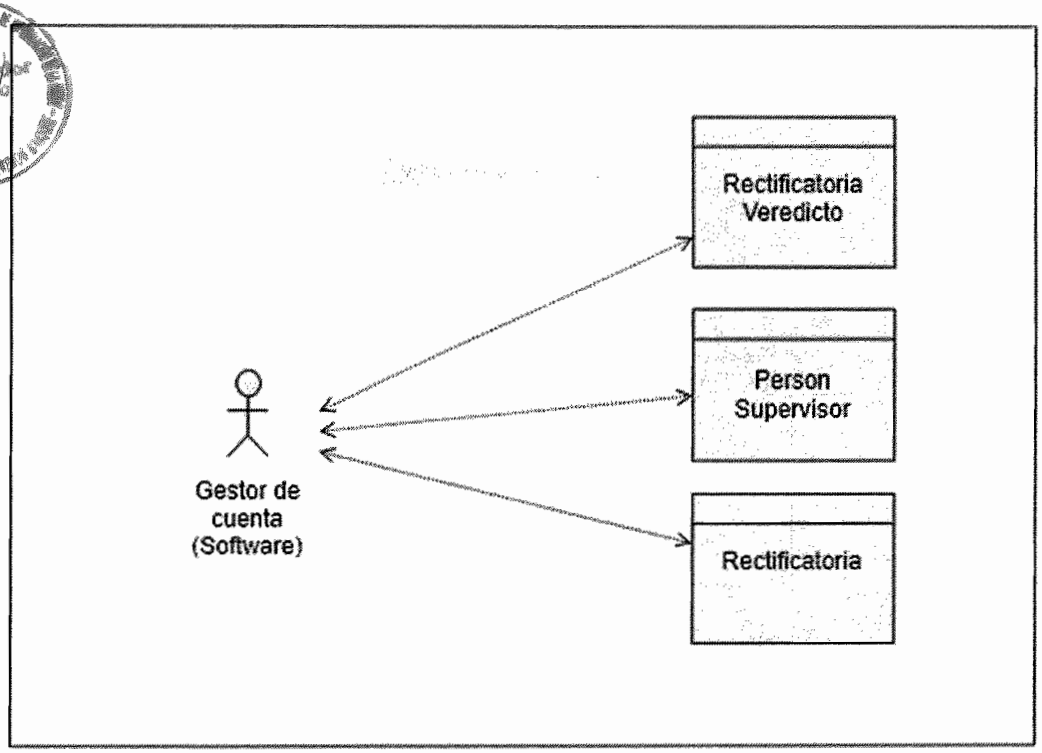

Figura 82 - Diagrama de Seguridad Rectificatoria (Gestor de cuenta - Software)

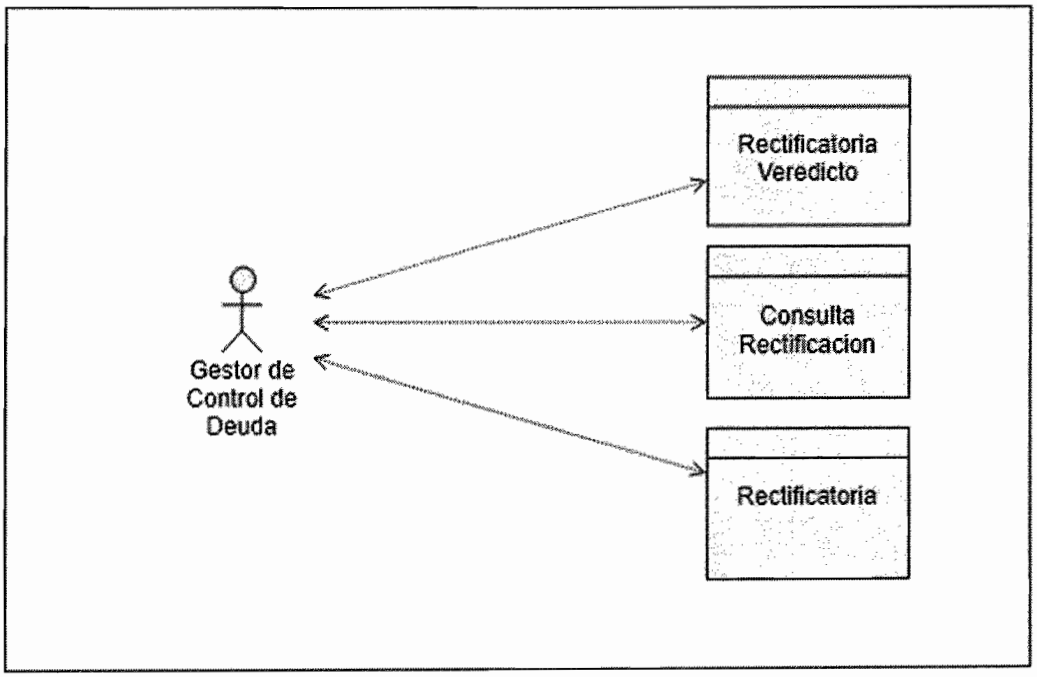

Figura 83 - Diagrama de Seguridad Rectificatoria (Gestor de Control de Deuda)

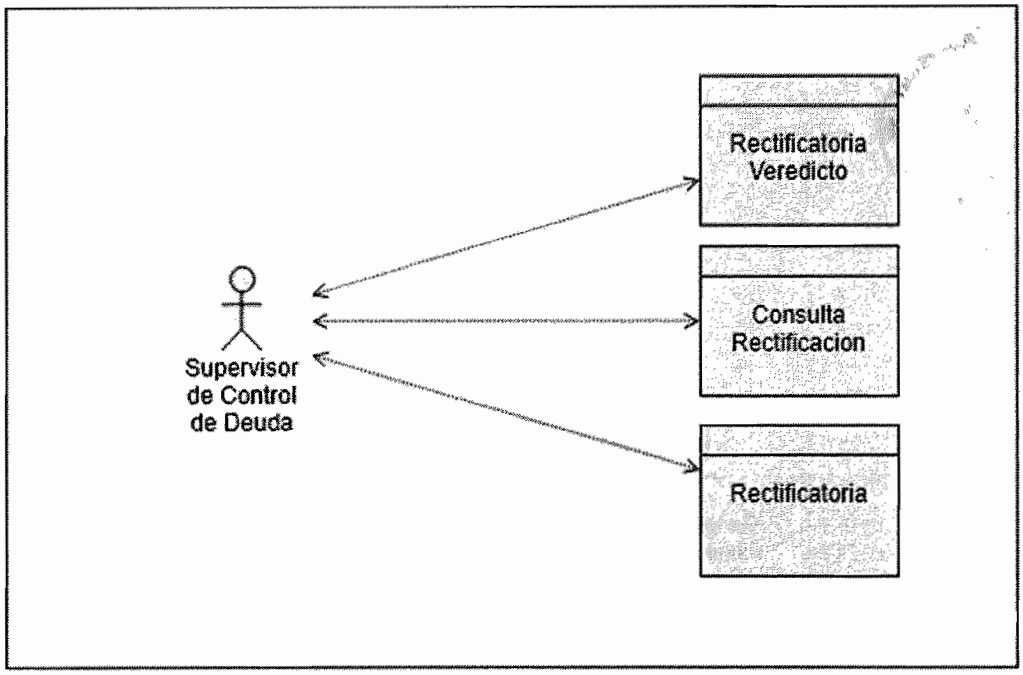

Figura 84 - Diagrama de Seguridad Rectificatoria (Supervisor de Control de Deuda)

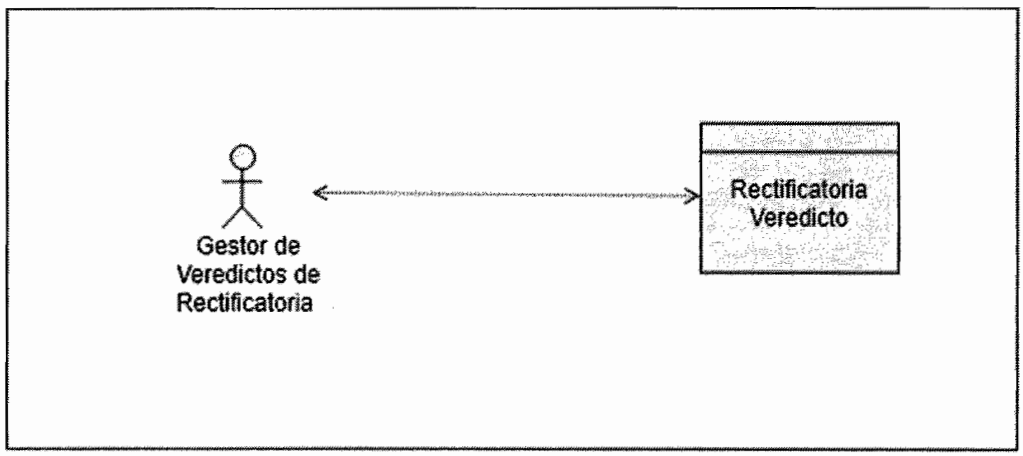

Figura 85 - Diagrama de Seguridad Rectificatoria (Gestor de Veredictos de Rectificatoria)

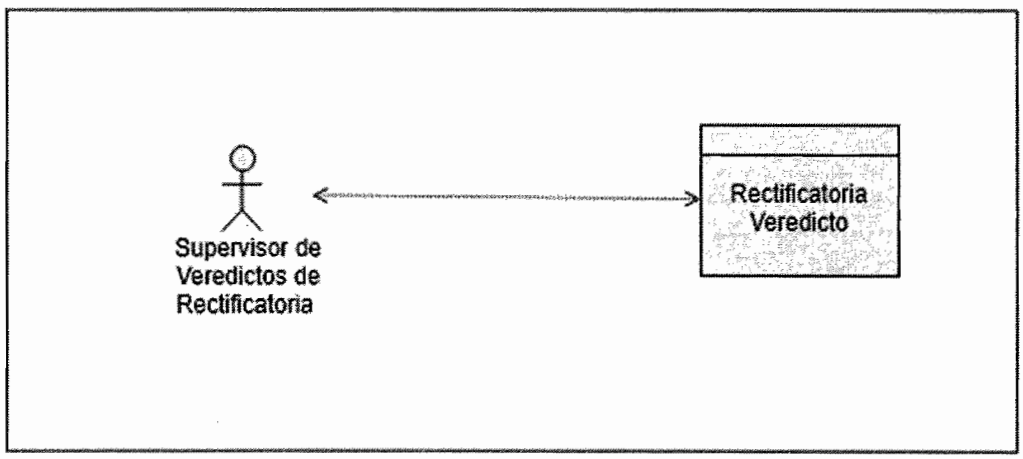

Figura 86 - Diagrama de Seguridad Rectificatoria (Supervisor de Veredictos de Rectificatoria)

	Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda				Gestor de Veredictos de Rectificatoria				Supervisor de Veredictos de Rectificatoria			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
RectificatoriaVeredicto		X	X			X	X			X	X		X	X	X	X		X	X	
Consulta Rectificacion						X				X										
PersonSupervisor		X																		
Rectificatoria	X					X	X			X	X									

EP008 – Devoluciones

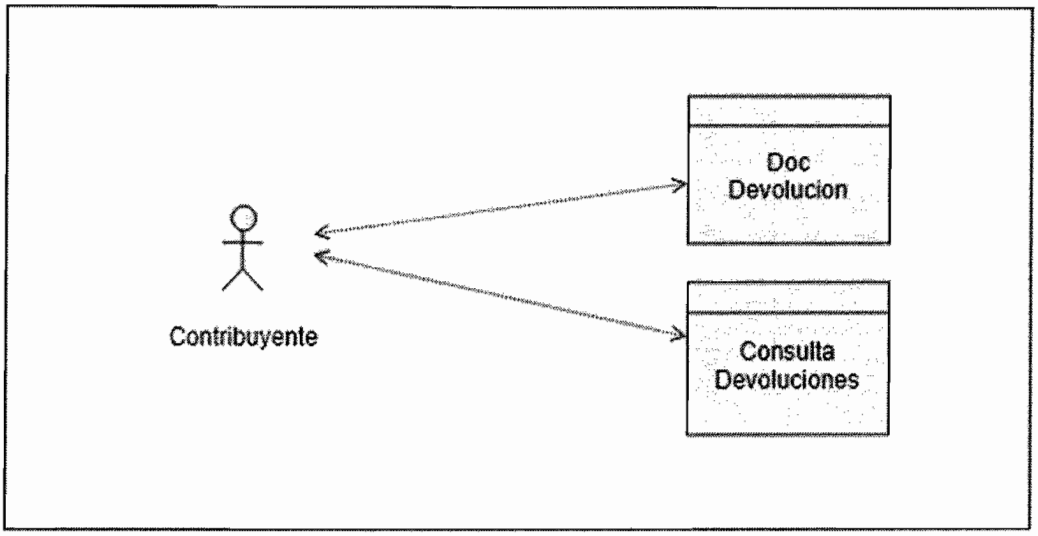

Figura 87 - Diagrama de Seguridad Devoluciones (Contribuyente)

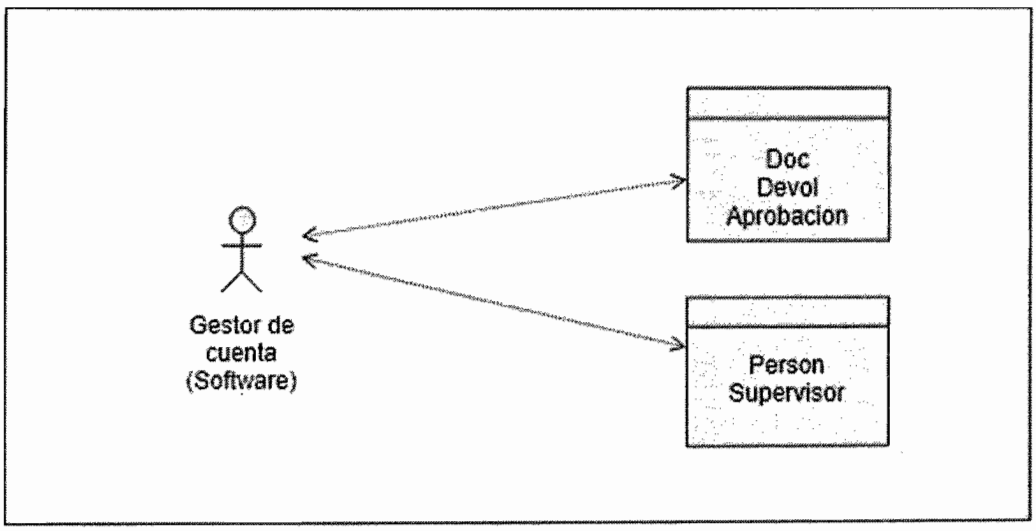

Figura 88 - Diagrama de Seguridad Devoluciones (Gestor de Cuenta - Software)

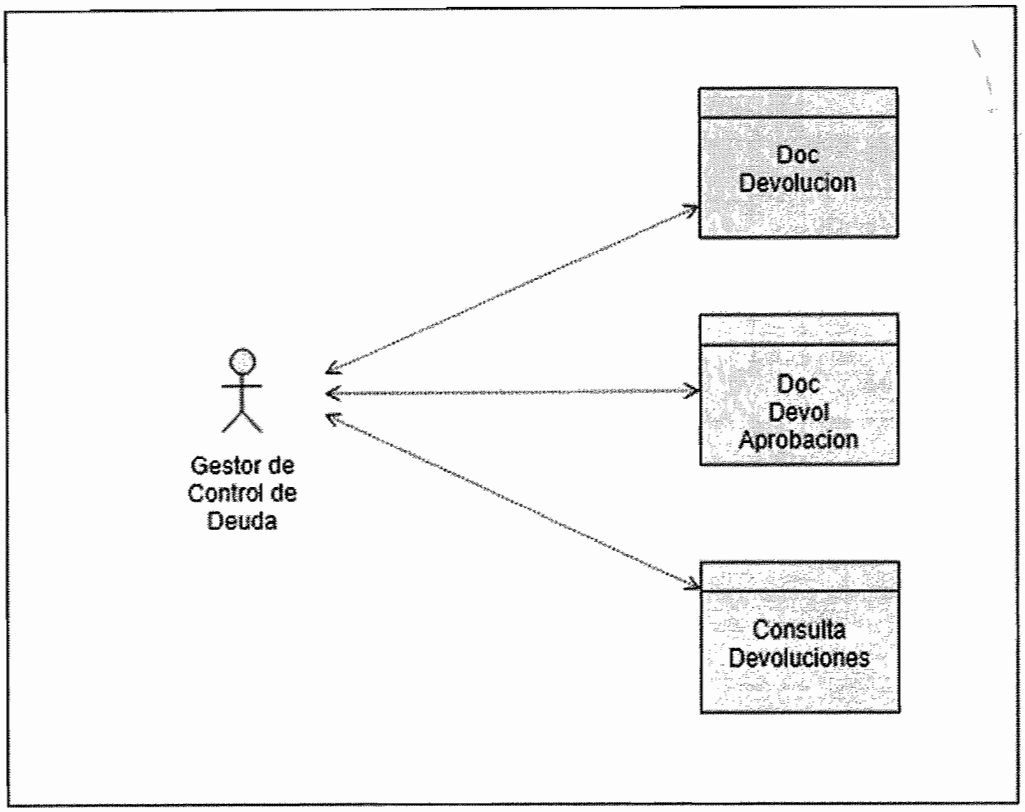

Figura 89 - Diagrama de Seguridad Devoluciones (Gestor de Control de Deuda)

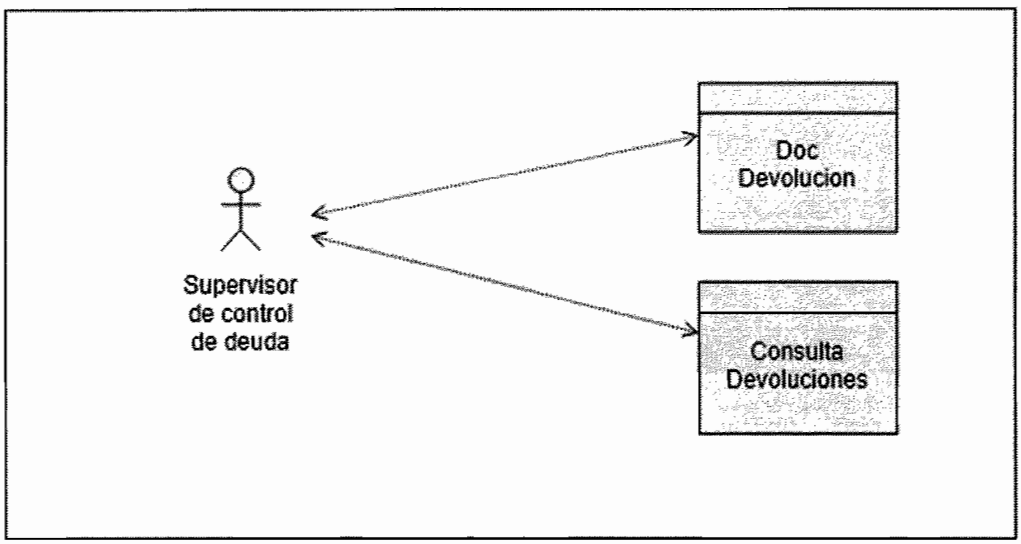

Figura 90 - Diagrama de Seguridad Devoluciones (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
DocDevolucion	X									X	X				X	
DocDevolAprobacion					X						X					
Consulta Devoluciones		X								X				X		
PersonSupervisor						X										

EP009 - Aplicar documento en cuenta única

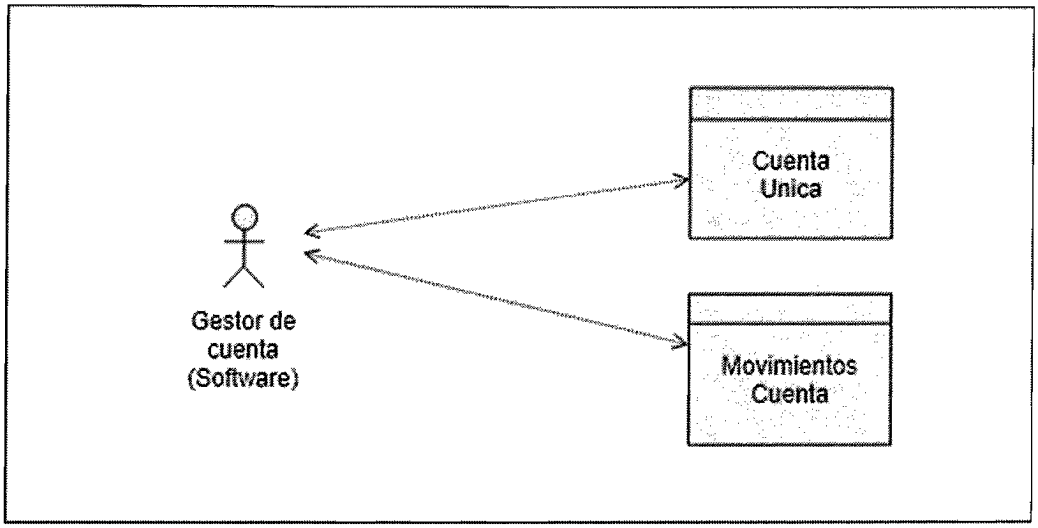

Figura 91 - Diagrama de Seguridad Aplicar Documento en Cuenta Única (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
CuentaUnica	X	X	X	
MovimientosCuenta	X	X	X	

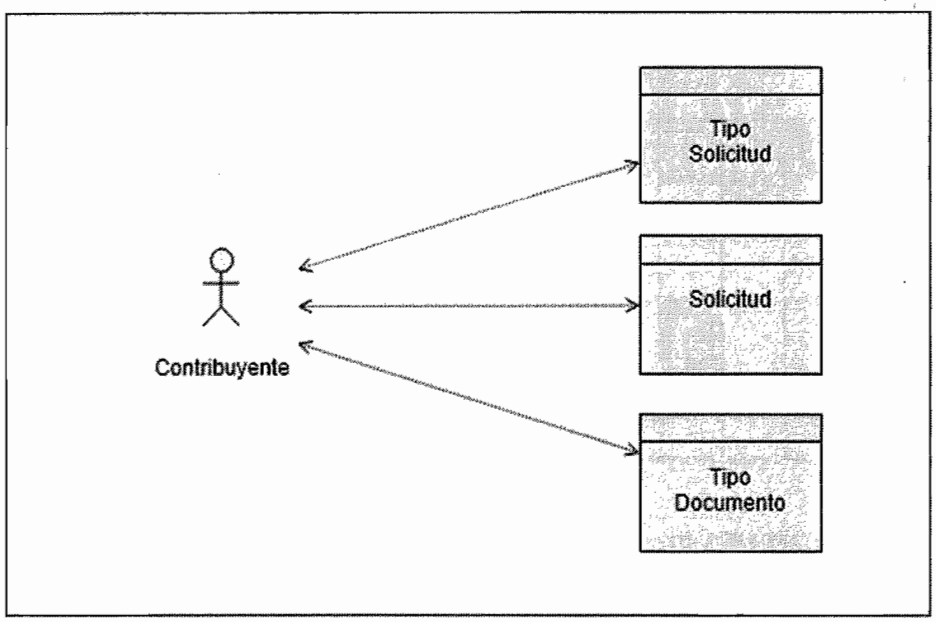

Figura 92 - Diagrama de Seguridad Error Material (Contribuyente)

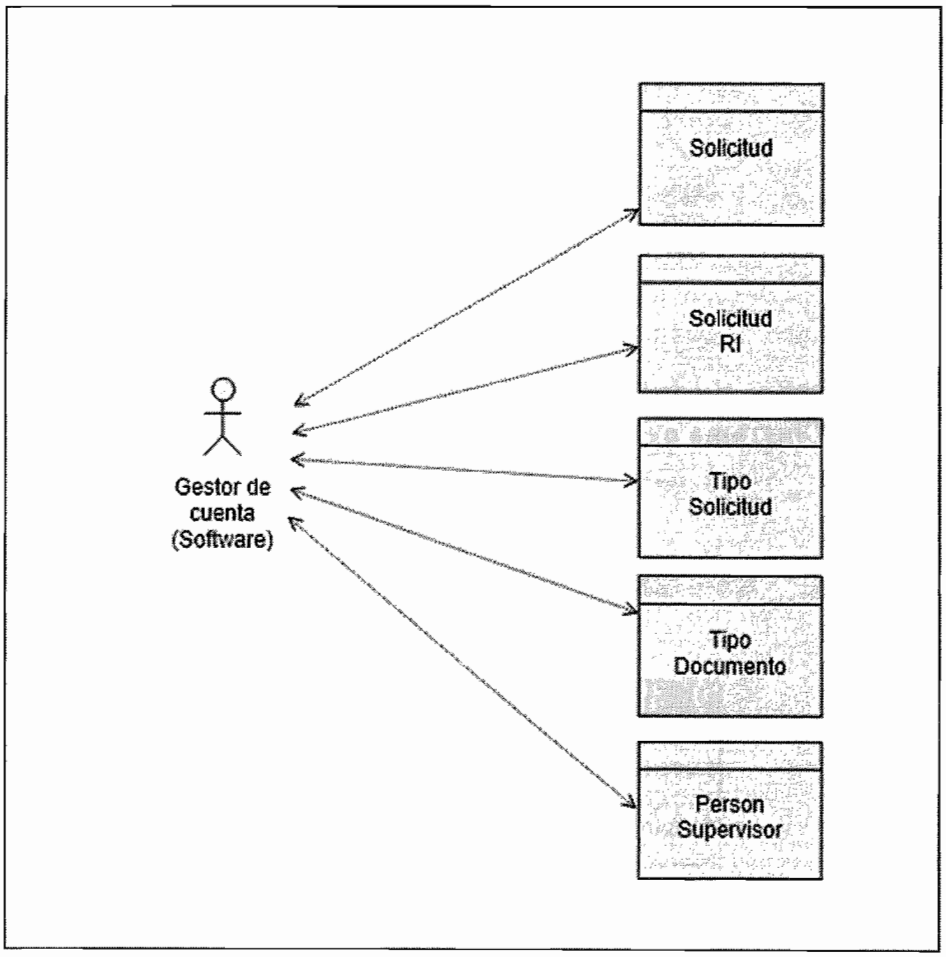

Figura 93 - Diagrama de Seguridad Error Material (Gestor de Cuenta - Software)

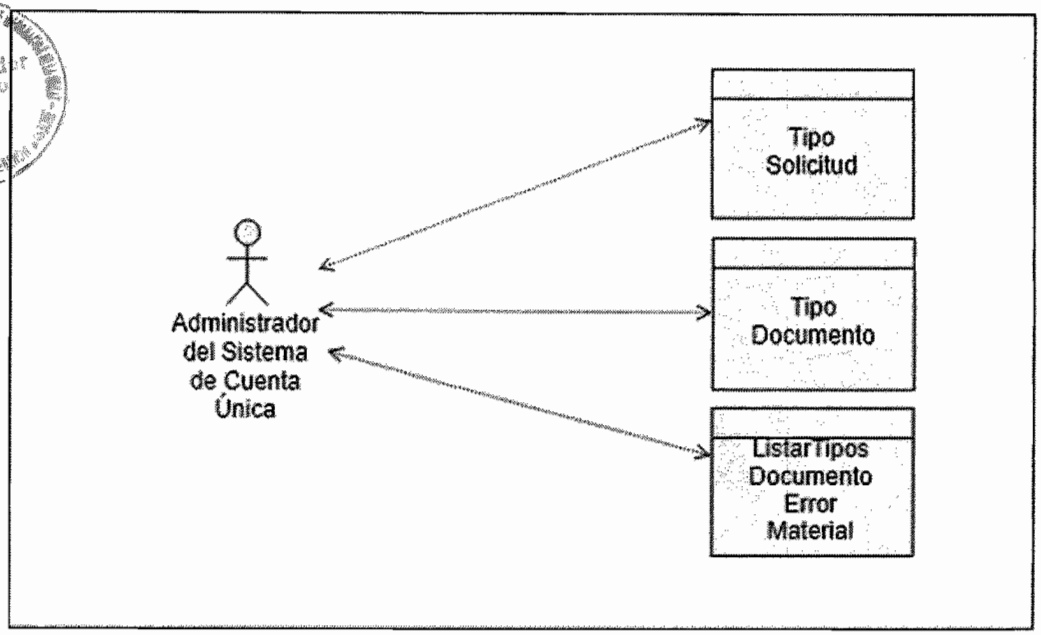

Figura 94 - Diagrama de Seguridad Error Material (Administrador del Sistema de Cuenta Única)

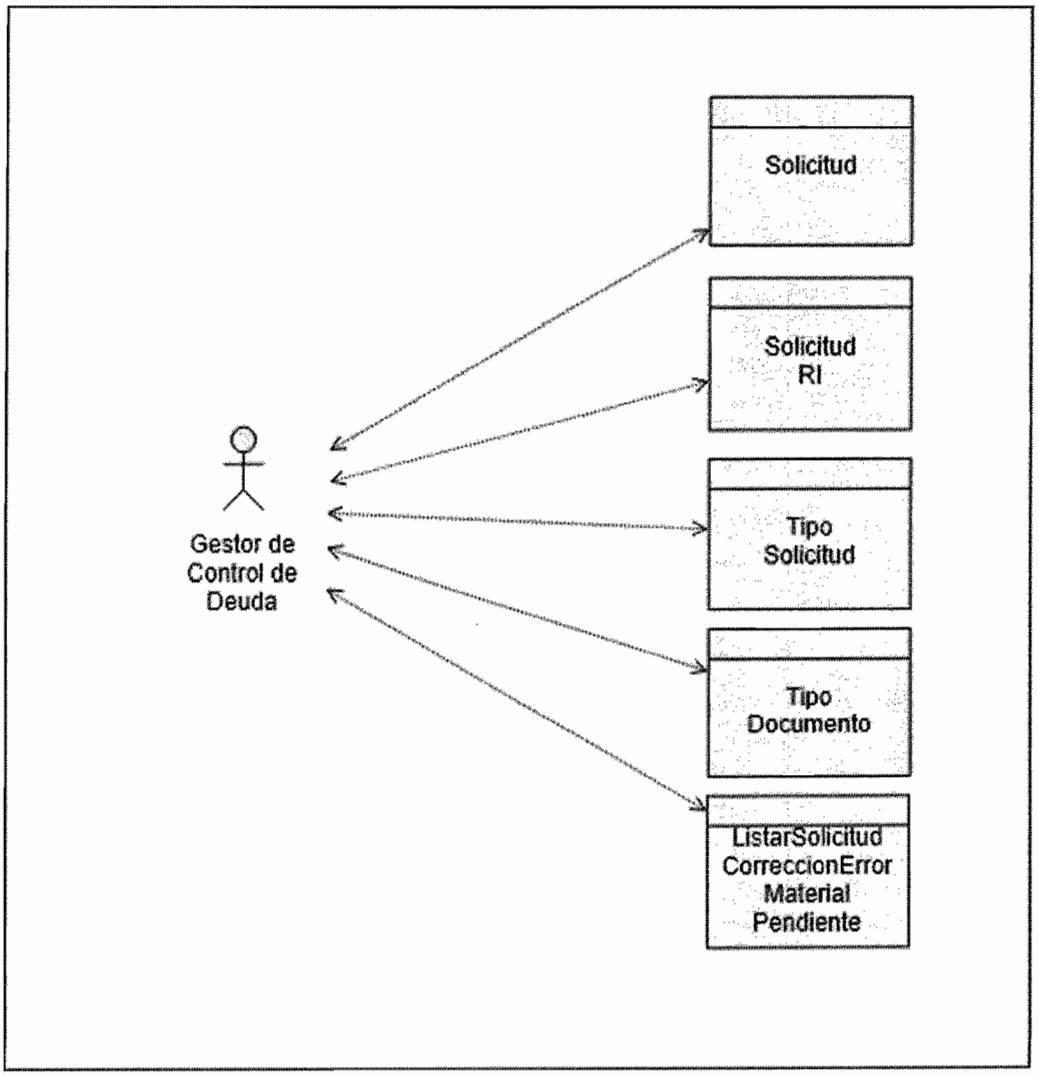

Figura 95 - Diagrama de Seguridad Error Material (Gestor de Control de Deuda)

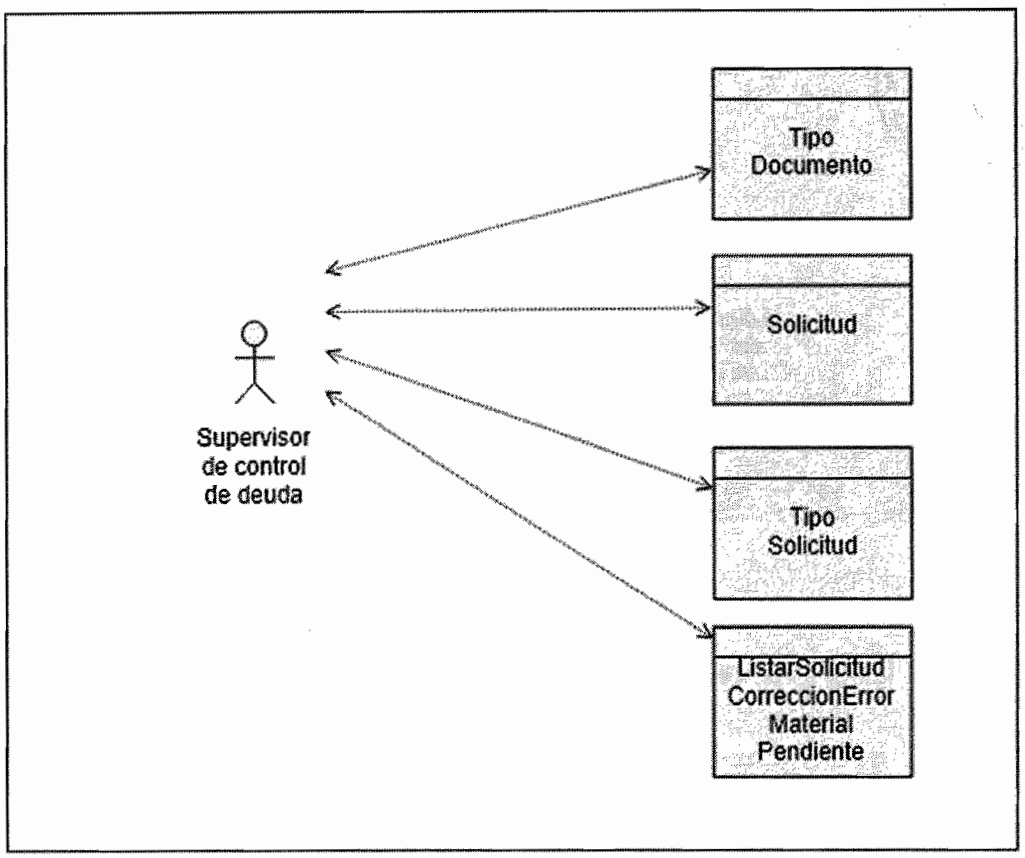

Figura 96 - Diagrama de Seguridad Error Material (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Cuenta (Software)				Administrador del Sistema de Cuenta Única				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
Solicitud	X					X								X	X			X	X	
SolicitudRI						X							X							
TipoSolicitud		X				X			X	X	X	X		X				X		
TipoDocumento		X				X			X	X	X	X		X				X		
PersonSupervisor						X														
Listar Solicitud Correccion Error Material Pendiente														X				X		
Listar Tipos Documento Error Material										X										

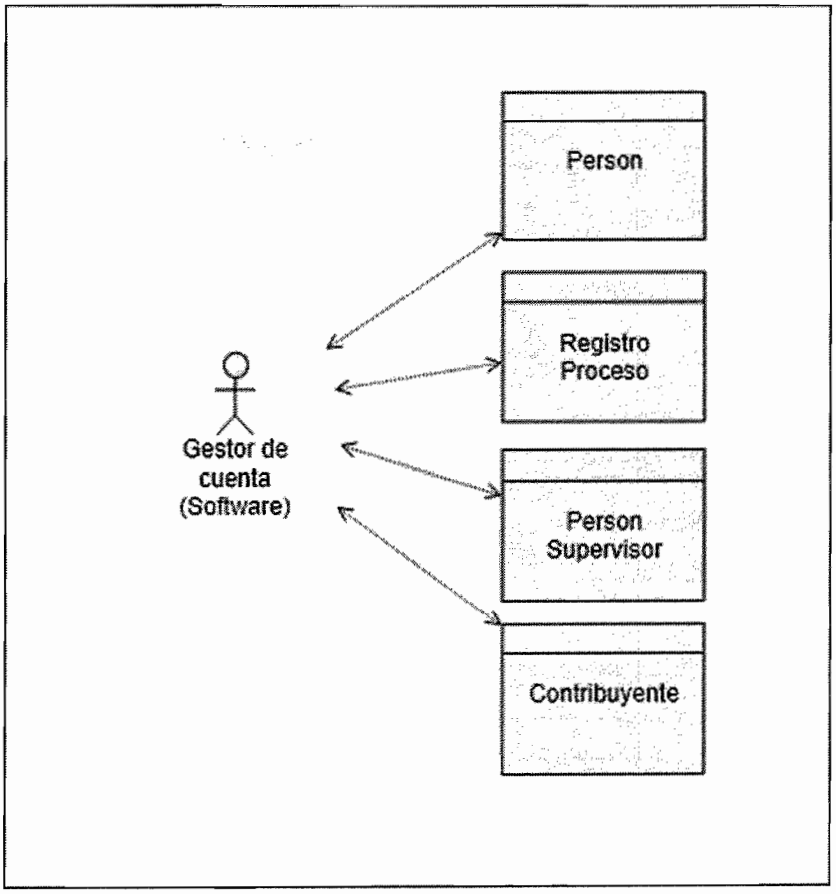

Figura 97 - Diagrama de Seguridad Corrección de Errores (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
RegistroProceso	X			
PersonSupervisor		X		
Contribuyente		X		
Person		X		

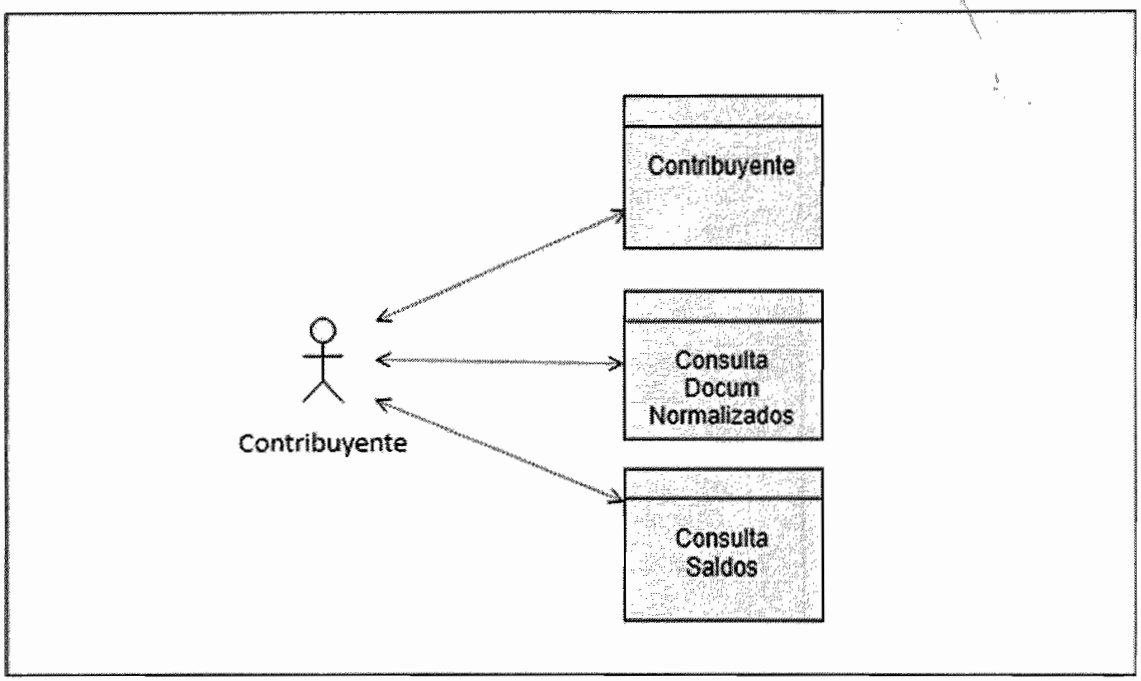

Figura 98 - Diagrama de Seguridad Consultas (Contribuyente)

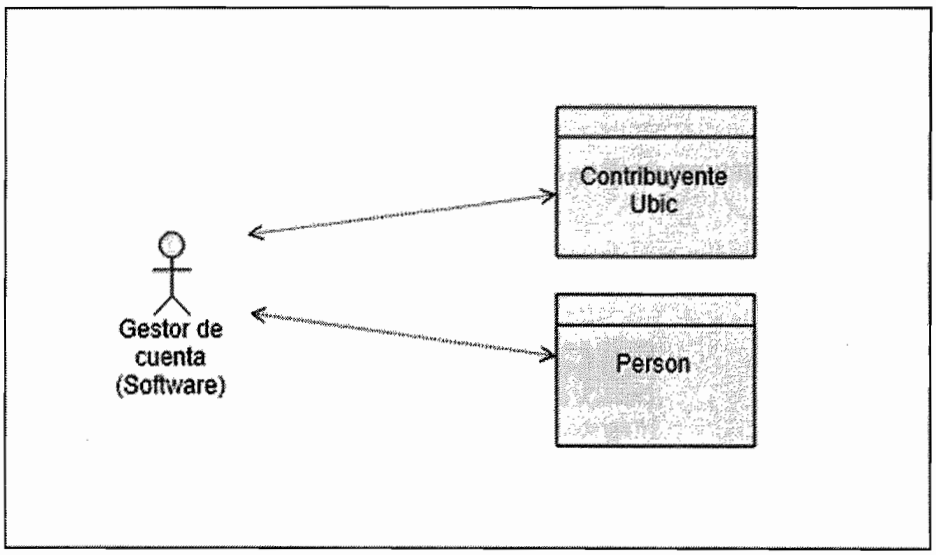

Figura 99 - Diagrama de Seguridad Consultas (Gestor de Cuenta - Software)

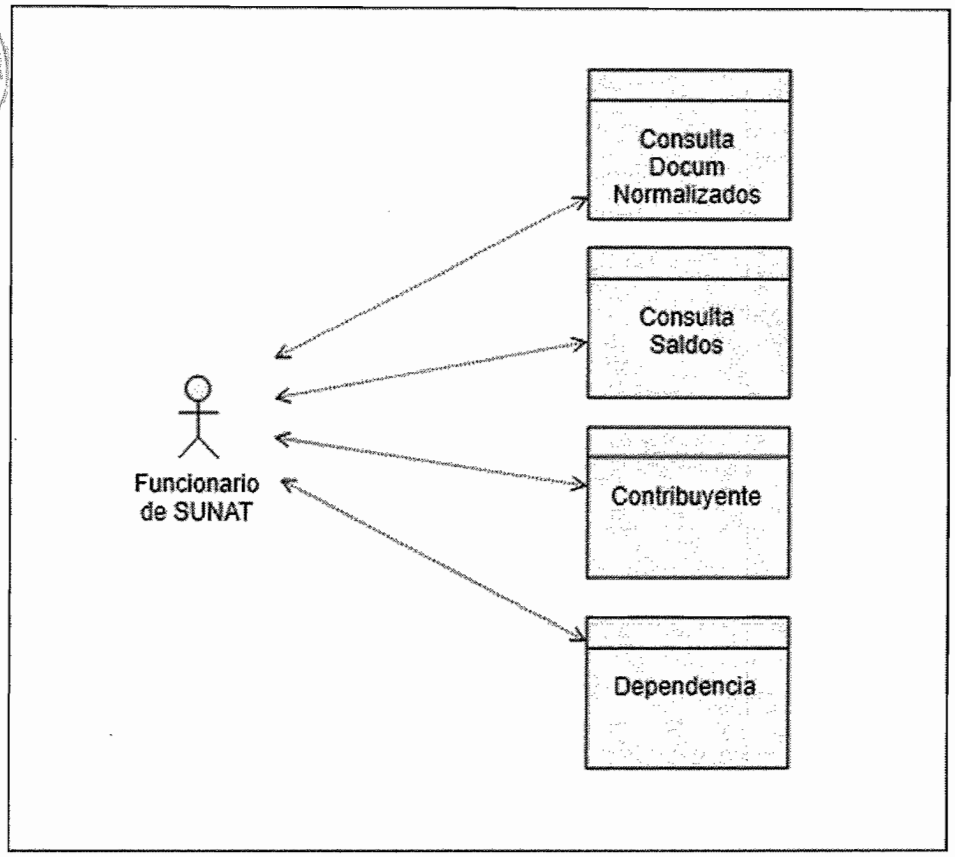

Figura 100 - Diagrama de Seguridad Consultas (Funcionario de SUNAT)

	Contribuyente				Gestor de Cuenta (Software)				Funcionario de SUNAT			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D
Contribuyente		X								X		
Dependencia										X		
ConsultaDocumNormalizados		X								X		
ConsultaSaldos		X								X		
ContribuyenteUbic						X						
Person						X						

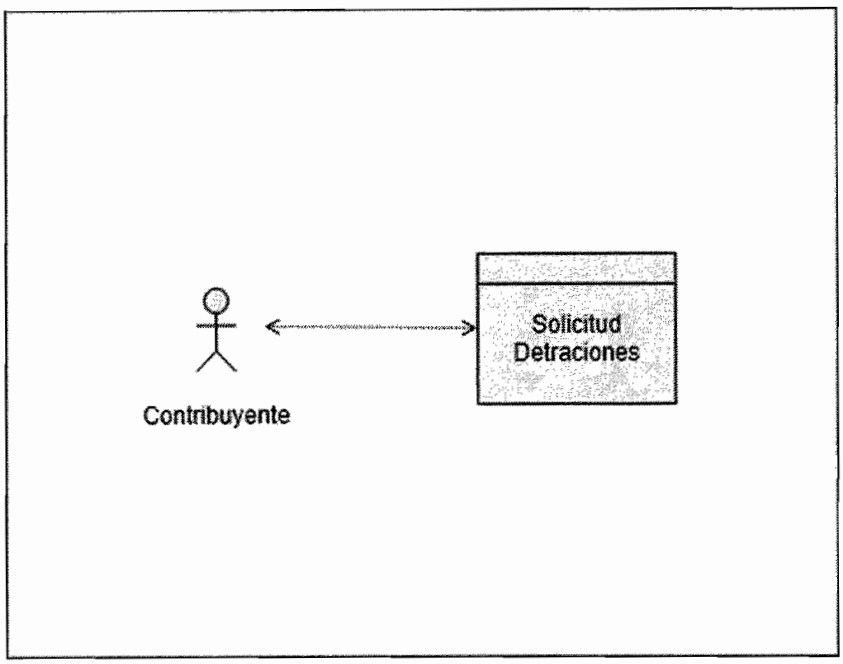

Figura 101 - Diagrama de Seguridad Aplicar Cuenta de Ingreso Detracciones (Contribuyente)

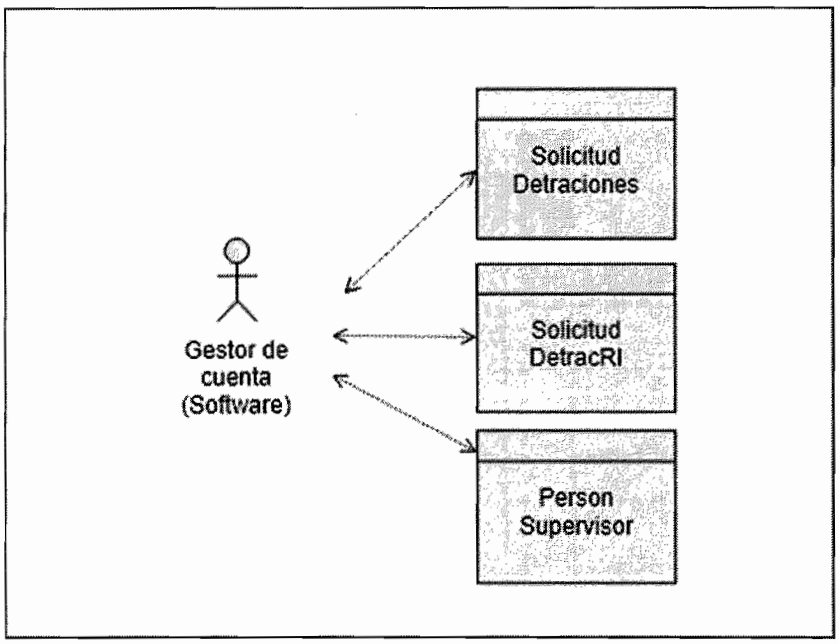

Figura 102 - Diagrama de Seguridad Aplicar Cuenta de Ingreso Detracciones (Gestor de Cuenta - Software)

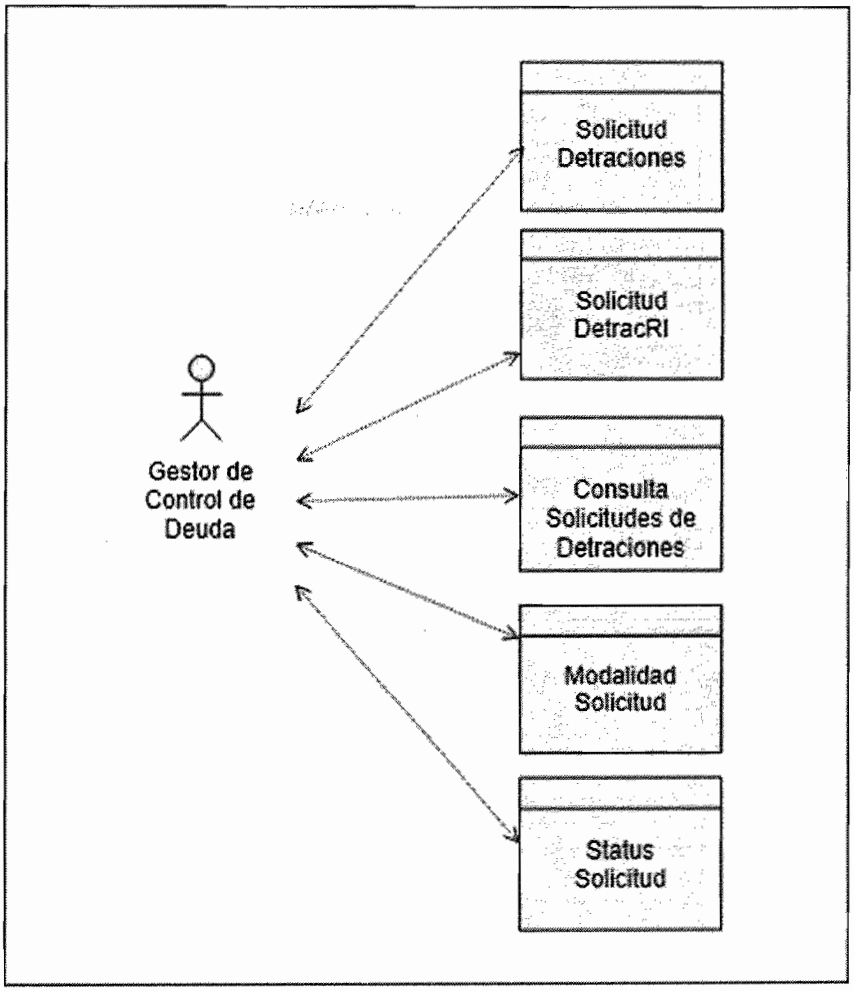

Figura 103 - Diagrama de Seguridad Aplicar Cuenta de Ingreso Detracciones (Gestor de Control de Deuda)

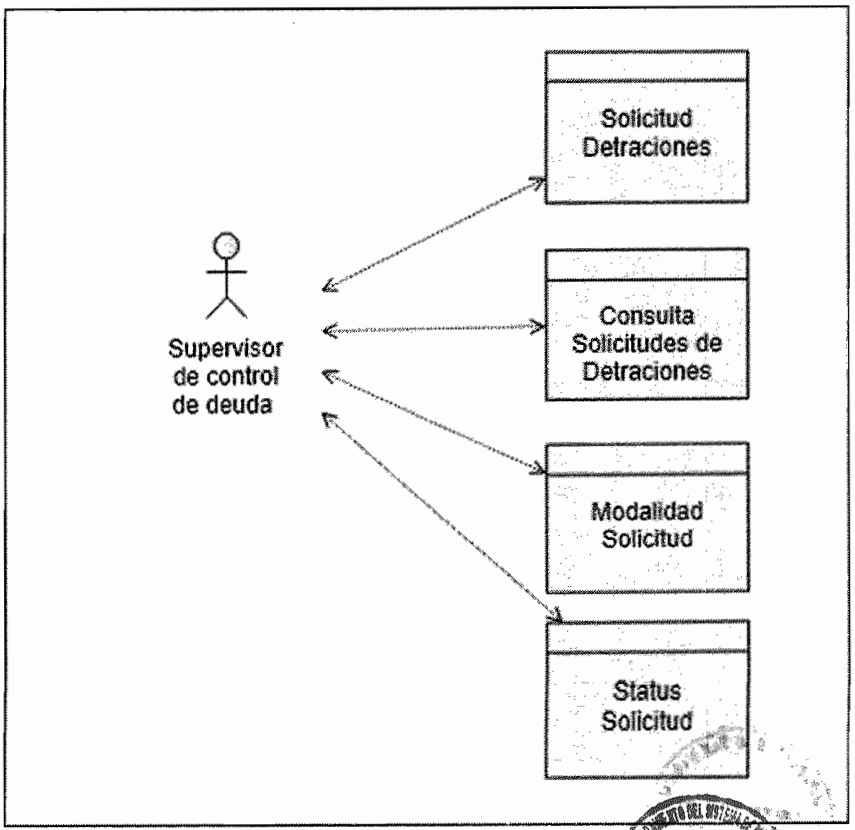

Figura 104 - Diagrama de Seguridad Aplicar Cuenta de Ingreso Detracciones (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
SolicitudDetraciones	X					X			X	X	X				X	
SolicitudDetracRI						X			X							
Consulta Solicitudes de Detraciones										X				X		
ModalidadSolicitud										X				X		
PersonSupervisor						X										
StatusSolicitud										X				X		

EP015 – Formularios y documentos de Cuenta Única

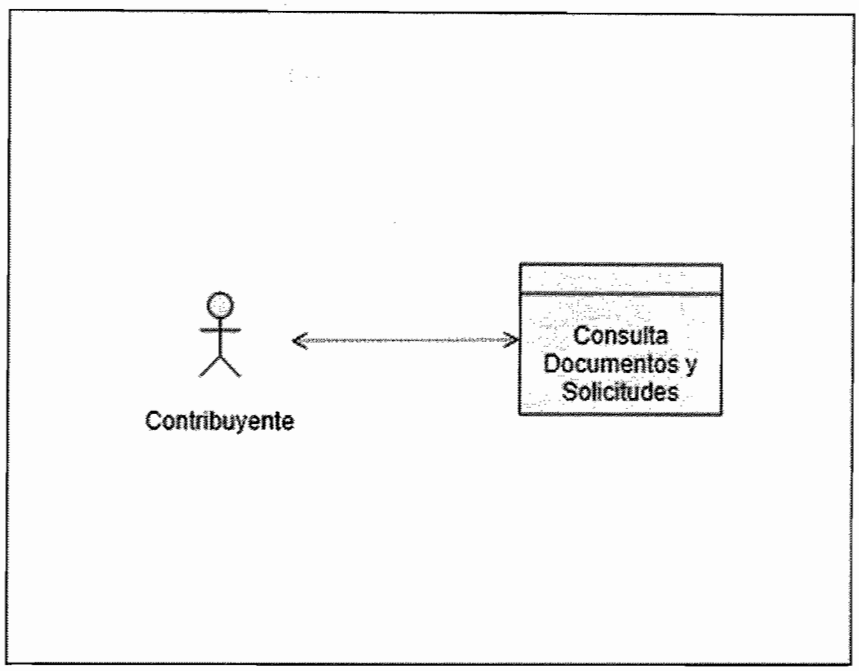

Figura 105 - Diagrama de Seguridad Formularios y Documentos (Contribuyente)

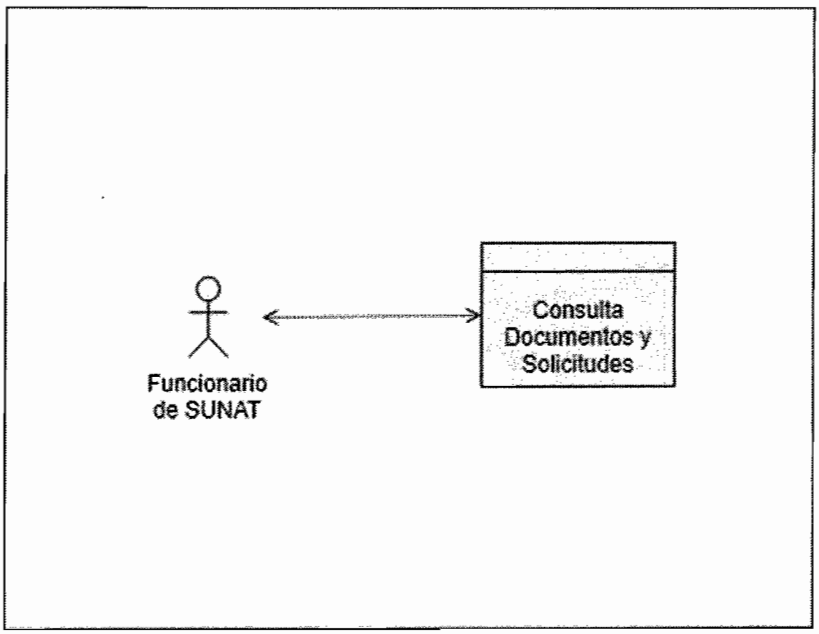

Figura 106 - Diagrama de Seguridad Formularios y Documentos (Funcionario de SUNAT)

	Contribuyente				Funcionario de SUNAT			
Entidad	C	R	U	D	C	R	U	D
Consulta Documentos y Solicitudes		X				X		

EP016 – PRESCRIPCIÓN

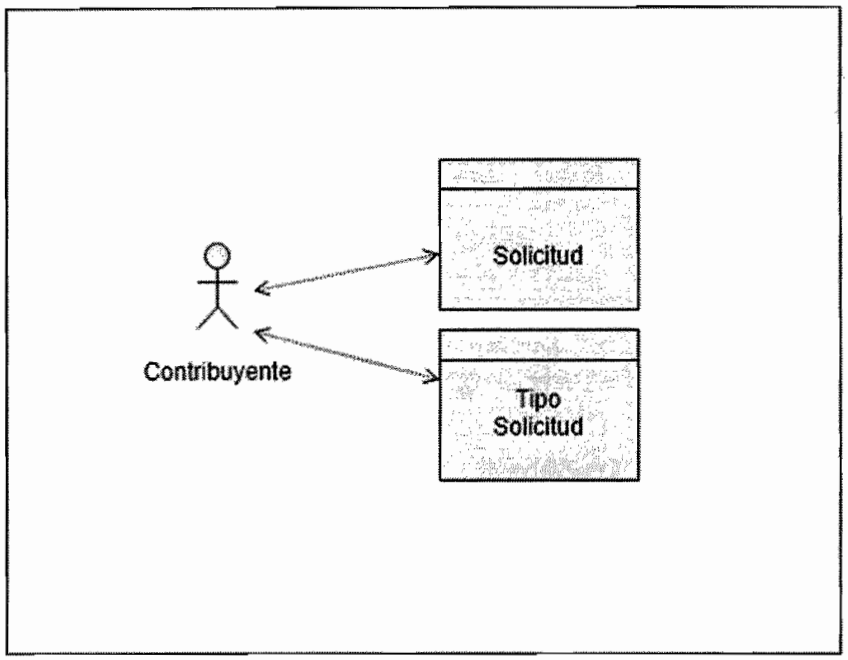

Figura 107 - Diagrama de Seguridad Prescripción (Contribuyente)

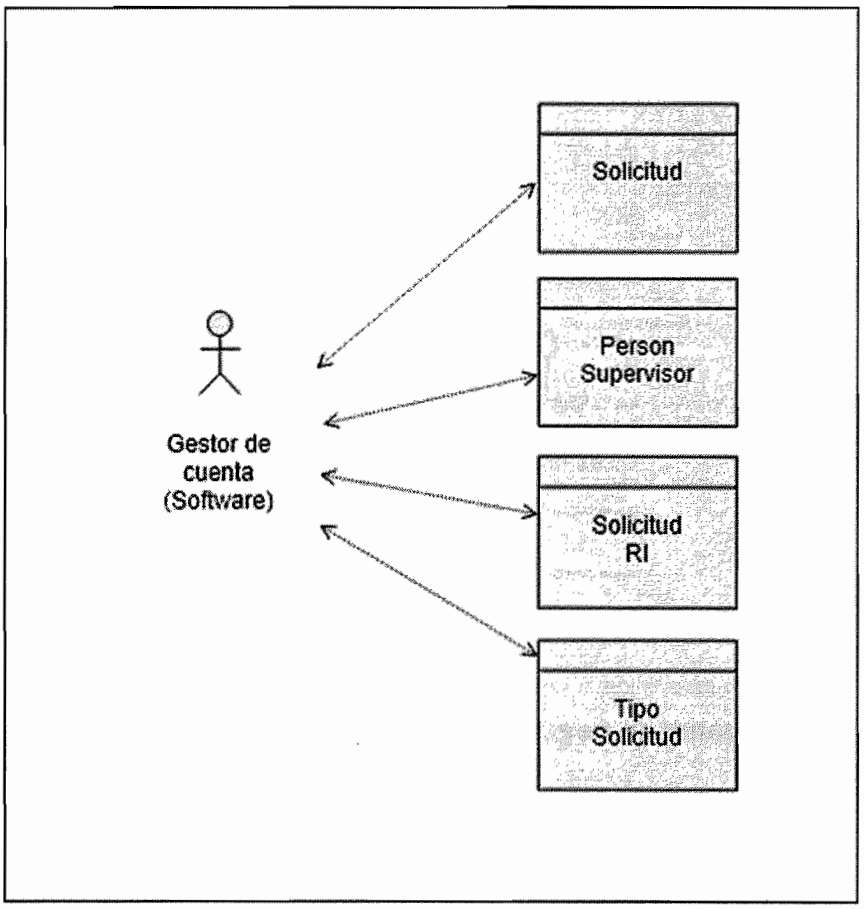

Figura 108 - Diagrama de Seguridad Prescripción (Gestor de Cuenta - Software)

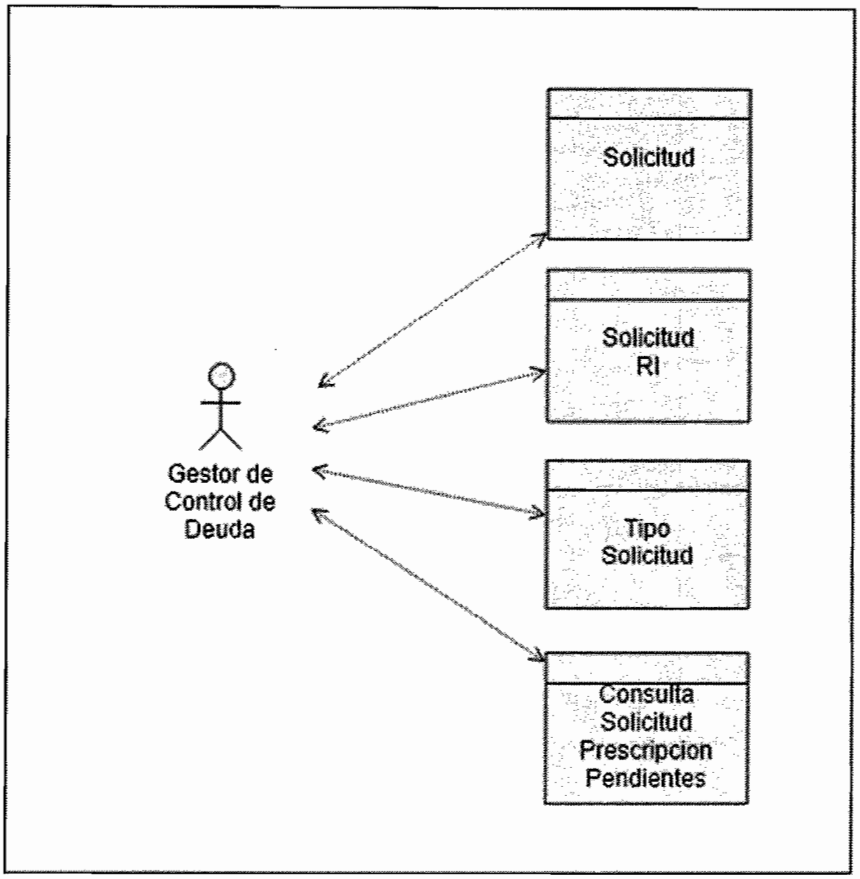

Figura 109 - Diagrama de Seguridad Prescripción (Gestor de Control de Deuda)

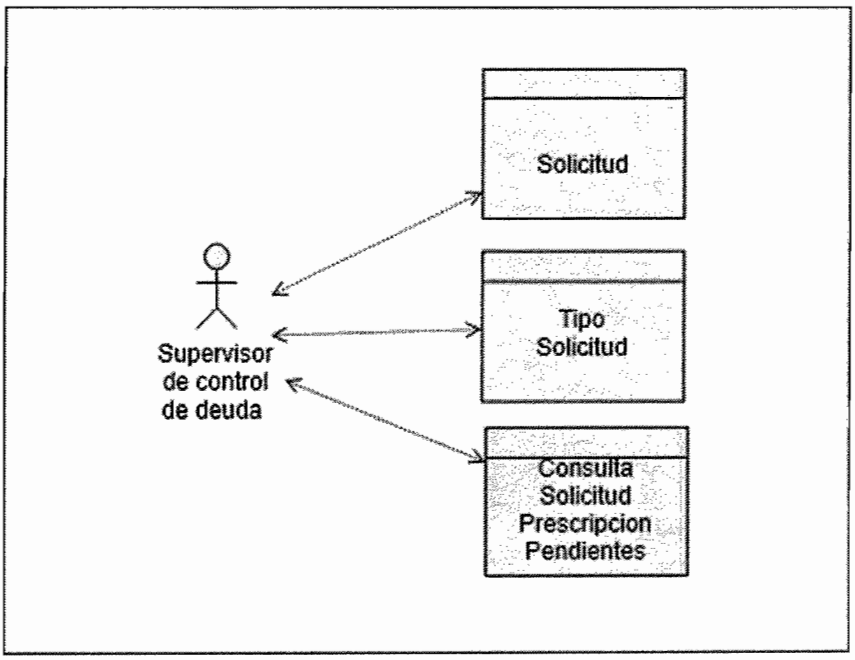

Figura 110 - Diagrama de Seguridad Prescripción (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
Solicitud	X					X	X			X	X			X	X	
PersonSupervisor						X										
SolicitudRI						X	X		X	X	X					
TipoSolicitud		X				X				X				X		
Consulta SolicitudPrescripcion Pendientes										X				X		

EP018 - Incumplimiento Entidades del Estado

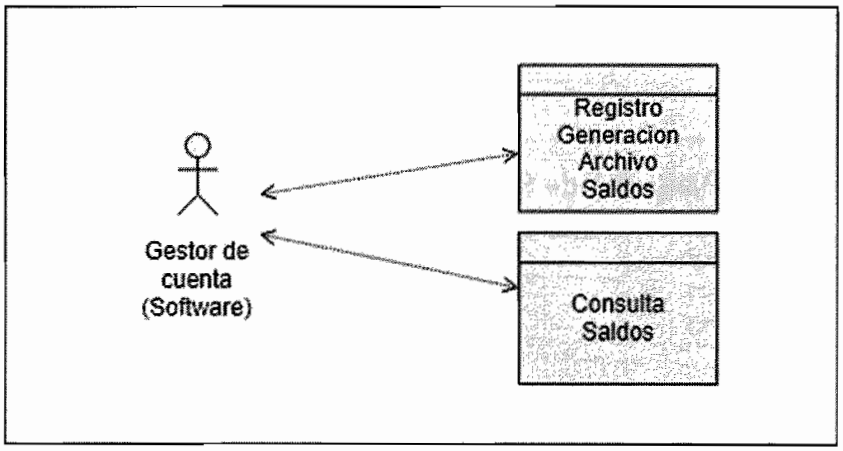

Figura 111 - Diagrama de Seguridad Incumplimiento Entidades del Estado (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
Consulta Saldos		X		
RegistroGeneracionArchivoSaldos	X	X	X	

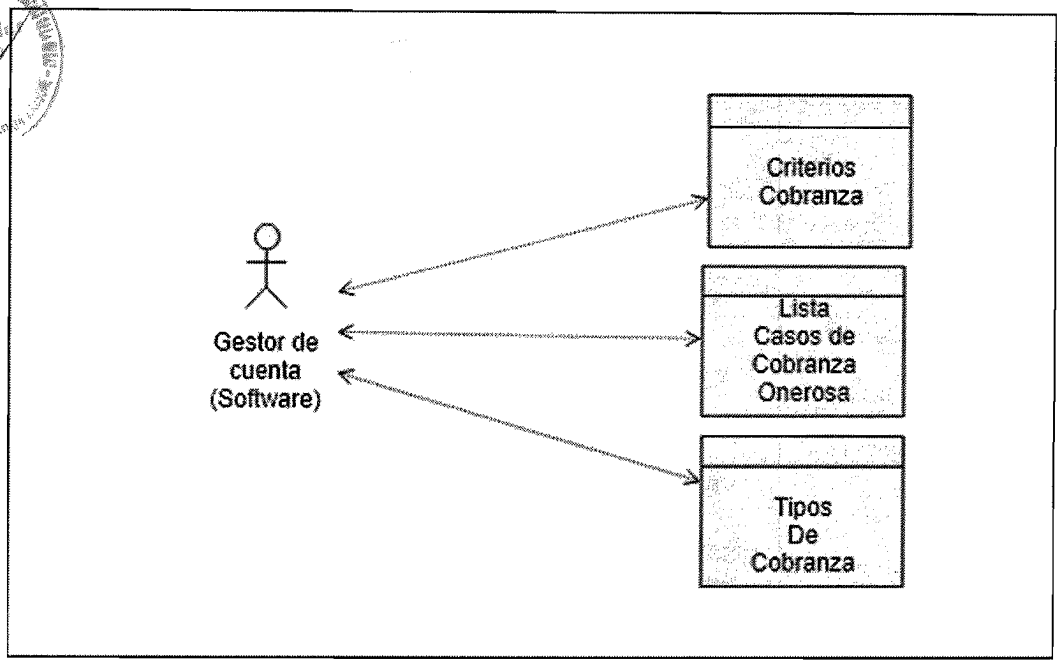

Figura 112 - Diagrama de Seguridad Cobranza Onerosa (Gestor de Cuenta - Software)

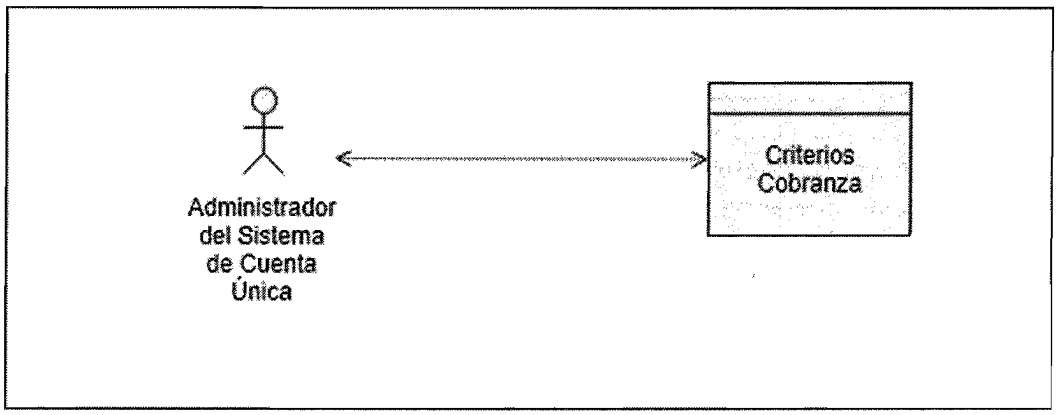

Figura 113 - Diagrama de Seguridad Cobranza Onerosa (Administrador del Sistema de Cuenta Única)

	Gestor de Cuenta (Software)				Administrador del Sistema de Cuenta Única			
Entidad	C	R	U	D	C	R	U	D
CriteriosCobranza		X			X	X	X	X
Lista Casos de Cobranza Onerosa		X						
TiposDeCobranza		X						

EP020 - Devoluciones de otros conceptos

Las entidades de devoluciones de otros conceptos son las mismas que están descritas en la sección 0 - EP008 – Devoluciones.

EP021 - Presentar declaracion jurada

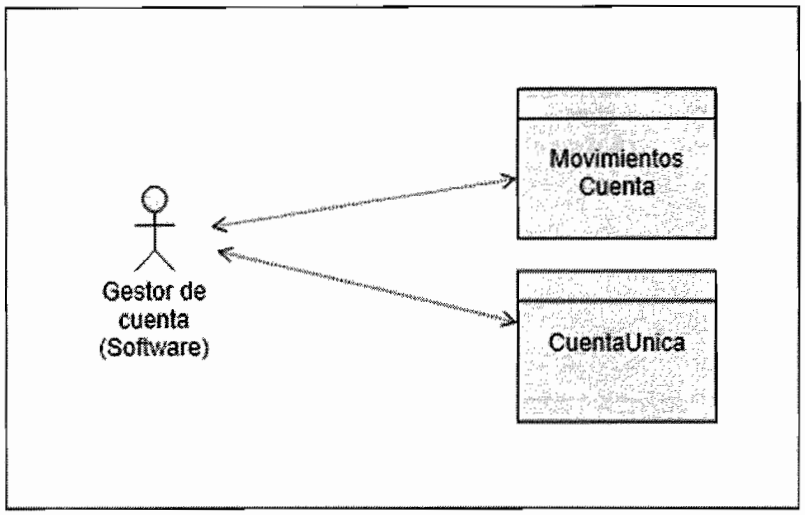

Figura 114 - Diagrama de Seguridad Declaración Jurada (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
MovimientosCuenta	X	X	X	
CuentaUnica	X	X	X	

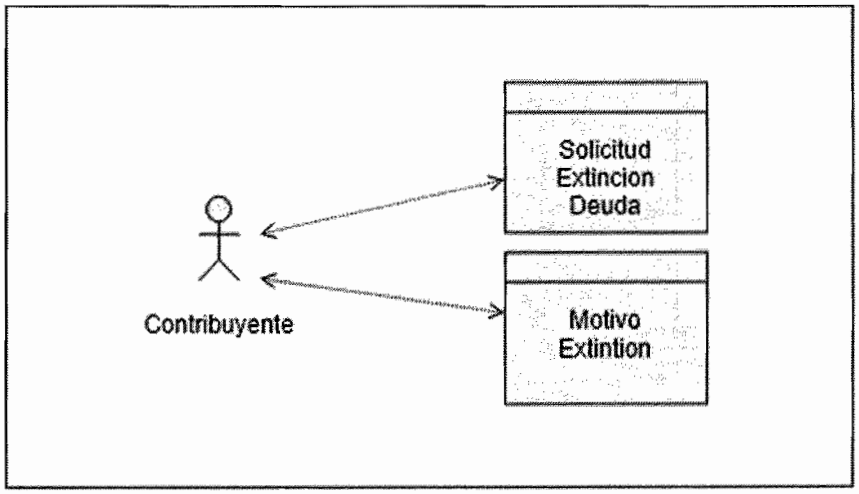

Figura 115 - Diagrama de Seguridad Extinción de Deuda (Contribuyente)

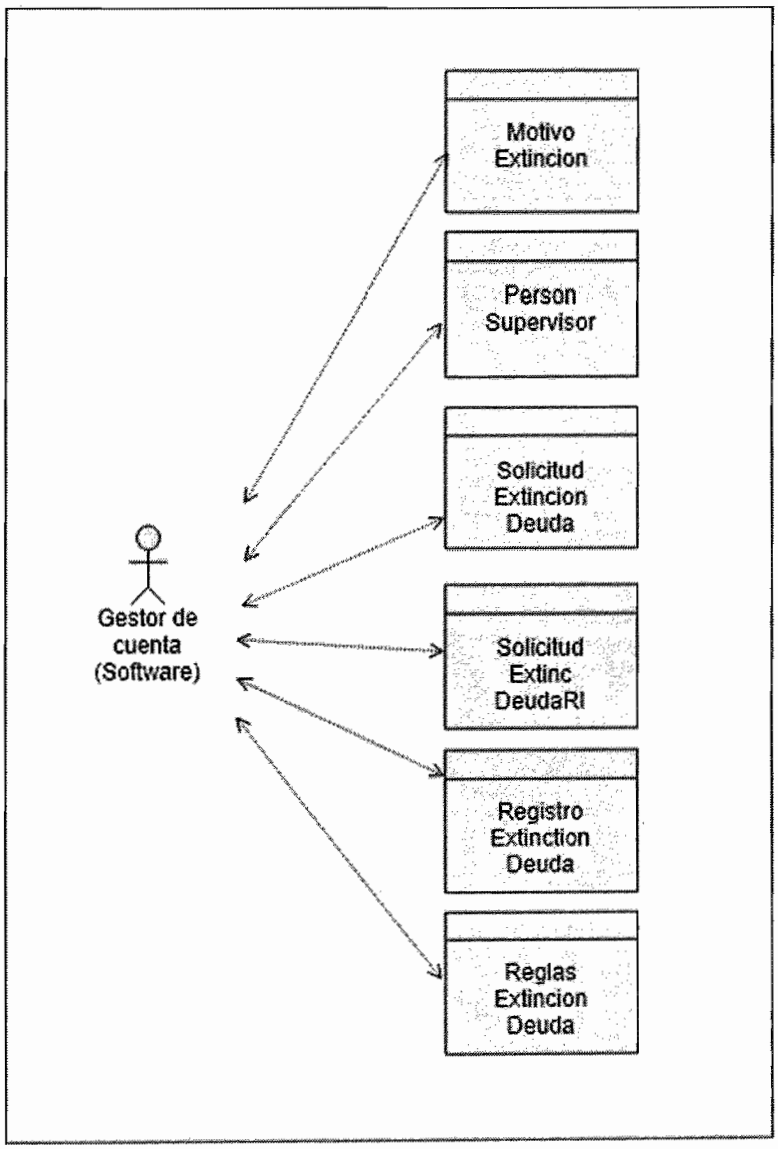

Figura 116 - Diagrama de Seguridad Extinción de Deuda (Gestor de Cuenta - Software)

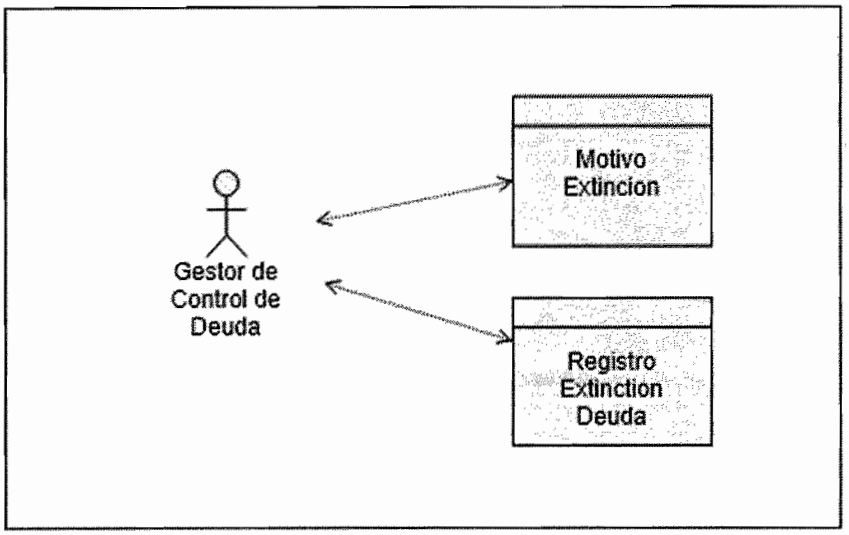

Figura 117 - Diagrama de Seguridad Extinción de Deuda (Gestor de Control de Deuda)

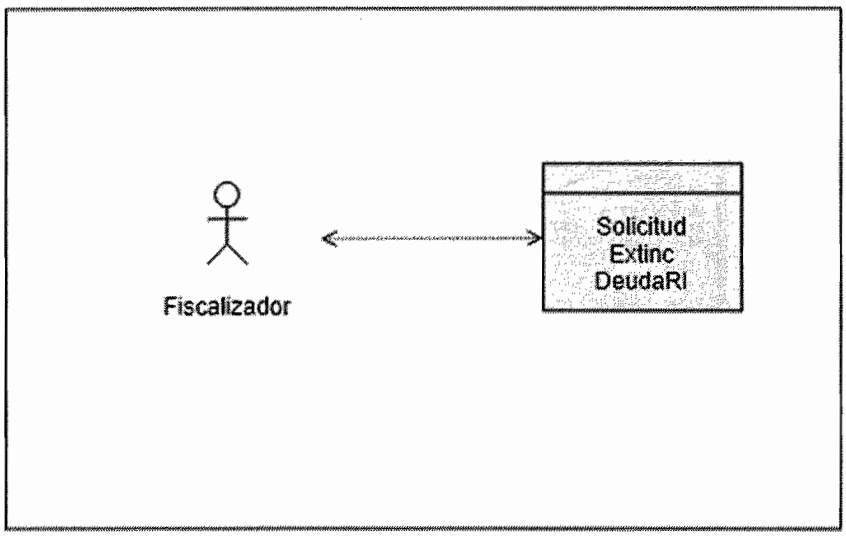

Figura 118 - Diagrama de Seguridad Extinción de Deuda (Fiscalizador)

	Contribuyente				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Fiscalizador			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
SolicitudExtincionDeuda	X					X										
SolicitudExtincDeudaRI						X							X	X		
MotivoExtincion		X				X				X						
PersonSupervisor						X										
RegistroExtinctionDeuda					X				X							
ReglasExtincionDeuda						X										

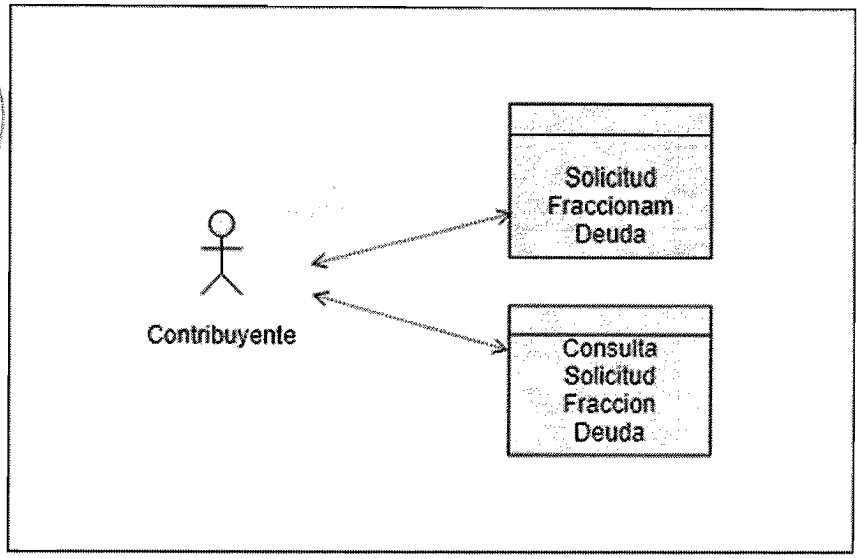

Figura 119 - Diagrama de Seguridad Fraccionamiento (Contribuyente)

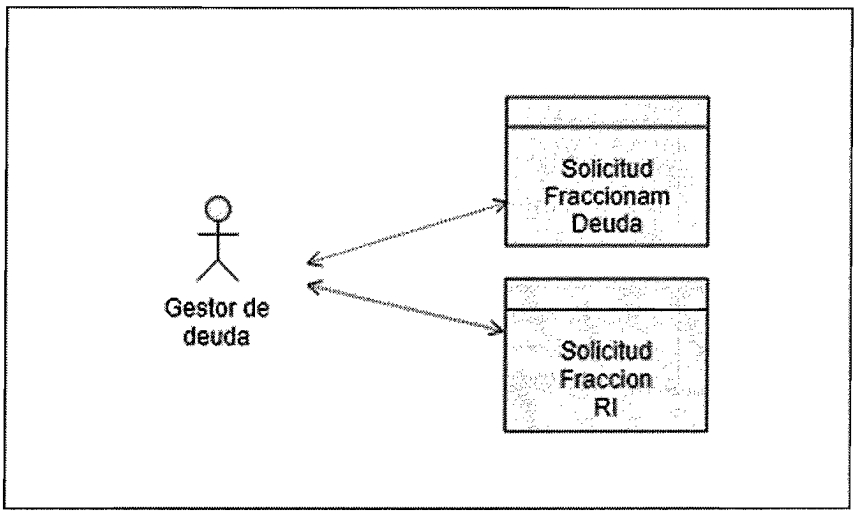

Figura 120 - Diagrama de Seguridad Fraccionamiento (Gestor de Deuda)

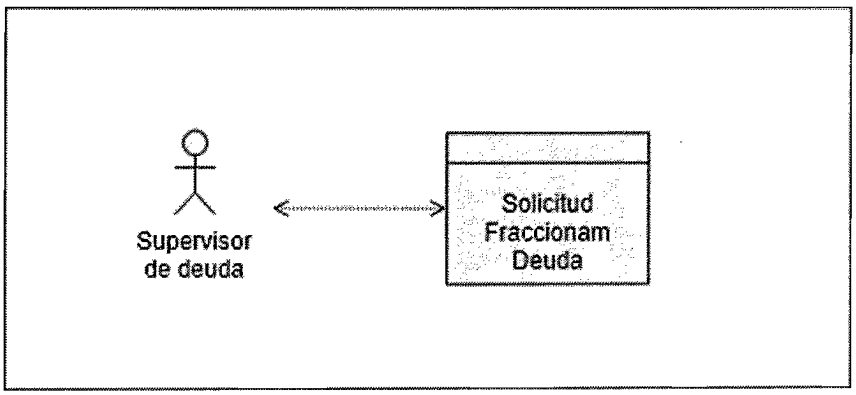

Figura 121 - Diagrama de Seguridad Fraccionamiento (Supervisor de Deuda)

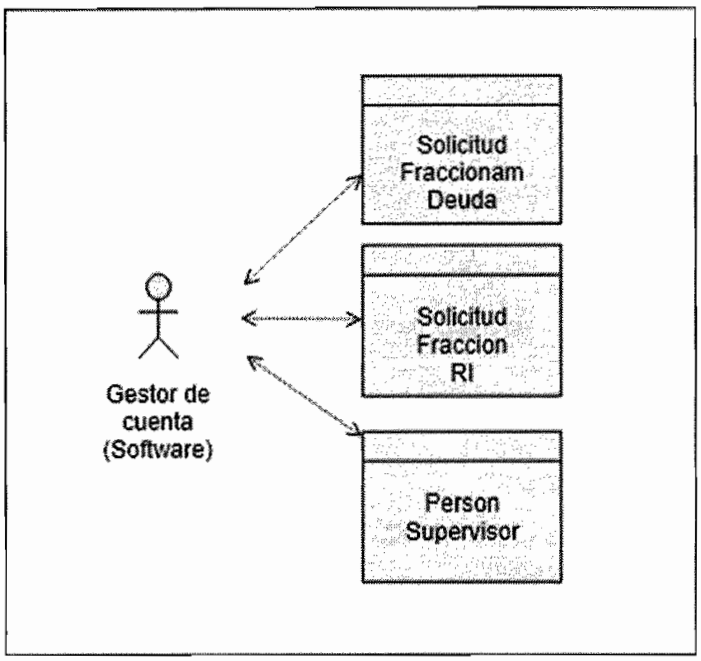

Figura 122 - Diagrama de Seguridad Fraccionamiento (Gestor de Cuenta - Software)

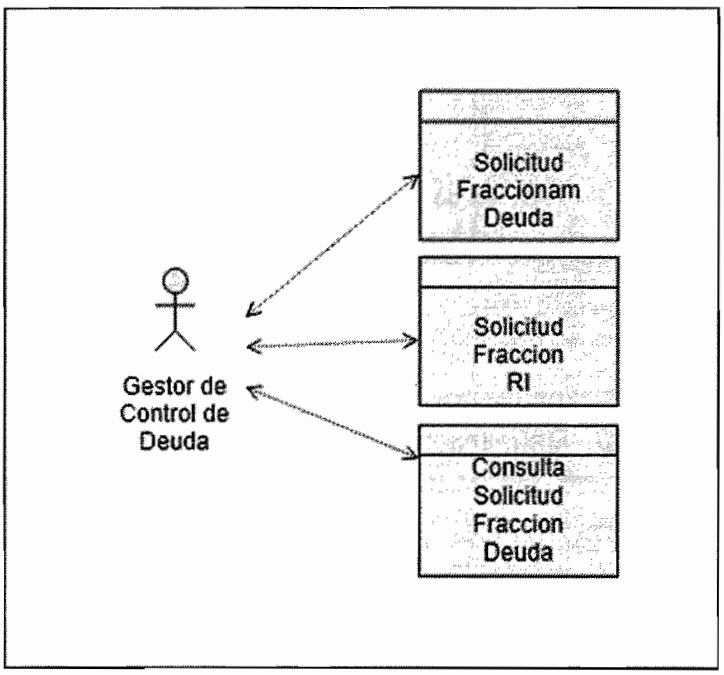

Figura 123 - Diagrama de Seguridad Fraccionamiento (Gestor de Control de Deuda)

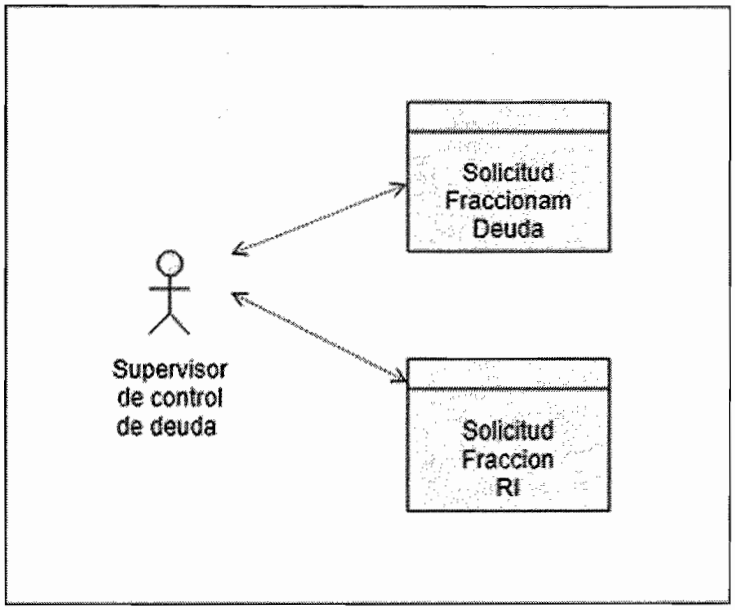

Figura 124 - Diagrama de Seguridad Fraccionamiento (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Deuda				Supervisor de Deuda				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
SolicitudFraccionamDeuda	X		X		X	X	X			X	X			X	X			X				X	X	
SolicitudFraccionRI					X	X	X							X				X				X		
Consulta Solicitud Fraccion Deuda		X																X						
PersonSupervisor														X										

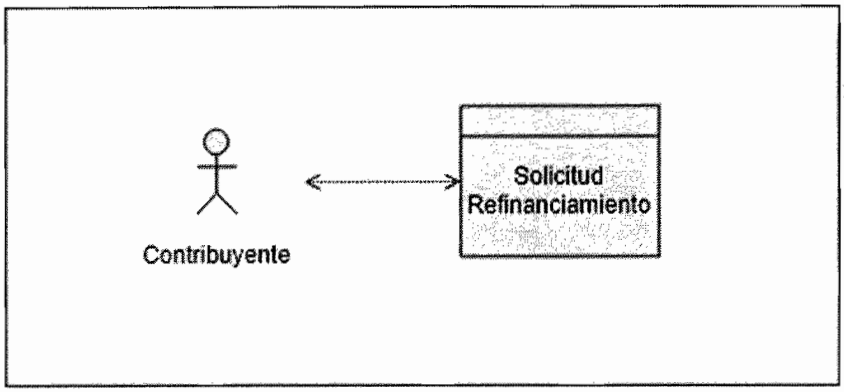

Figura 125 - Diagrama de Seguridad Refinanciamiento (Contribuyente)

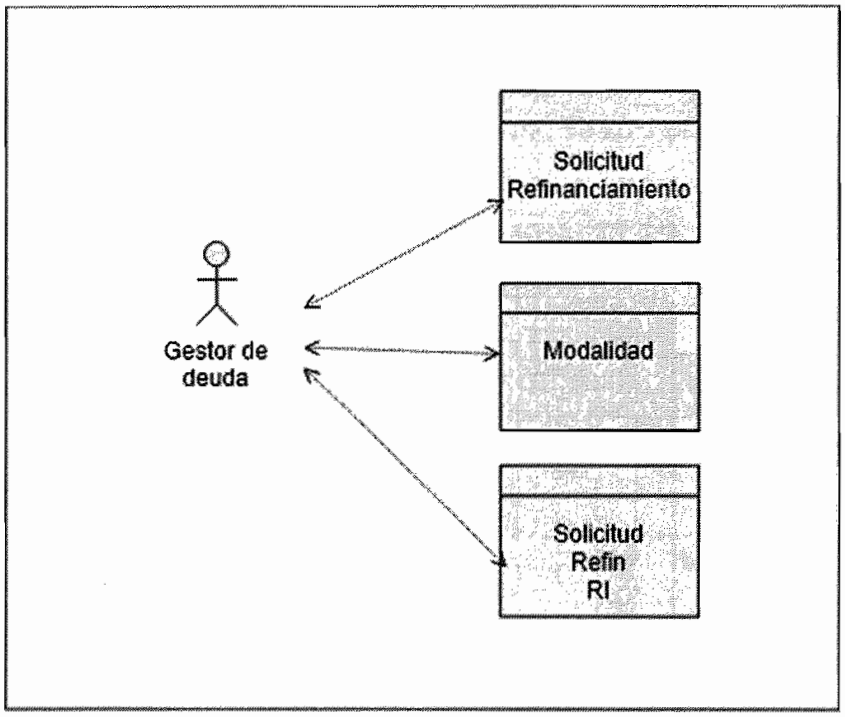

Figura 126 - Diagrama de Seguridad Refinanciamiento (Gestor de Deuda)

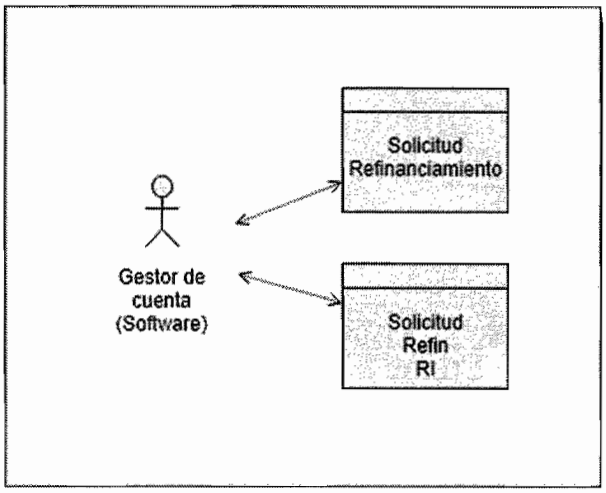

Figura 127 - Diagrama de Seguridad Refinanciamiento (Gestor de Cuenta - Software)

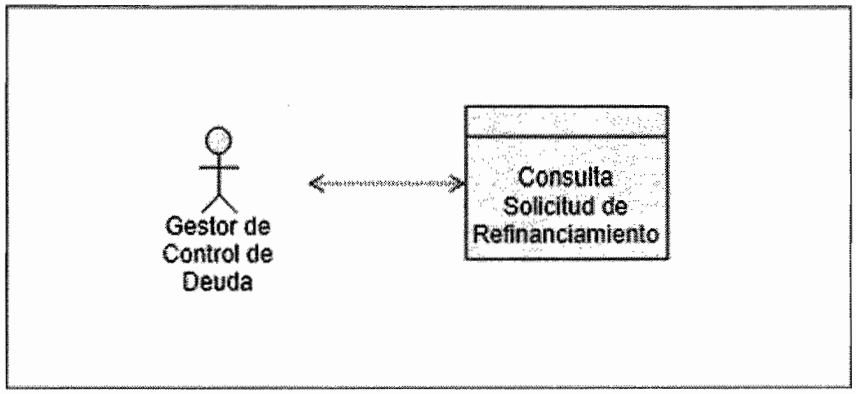

Figura 128 - Diagrama de Seguridad Refinanciamiento (Gestor de Control de Deuda)

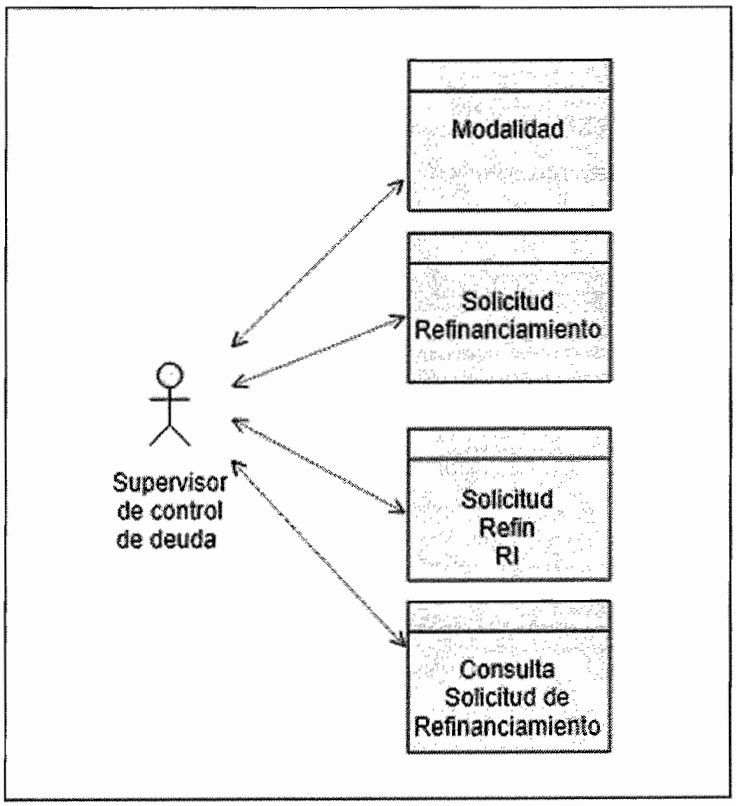

Figura 129 - Diagrama de Seguridad Refinanciamiento (Supervisor de Control de Deuda)

	Contribuyente				Gestor de Deuda				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
SolicitudRefinanciamiento	X				X	X	X			X	X							X	X	
SolicitudRefinRI					X	X			X	X								X		
Consulta Solicitud de Refinanciamiento														X				X		
Modalidad						X												X		

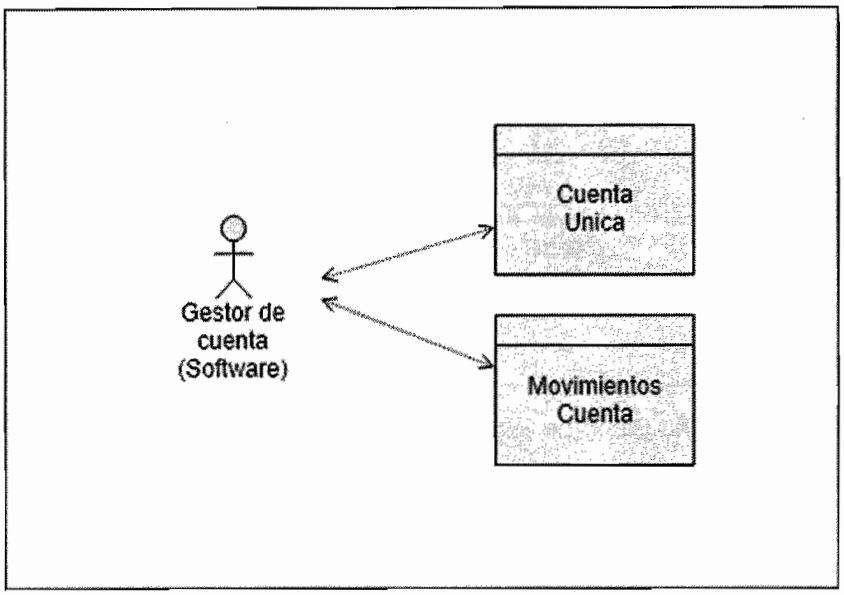

Figura 130 - Diagrama de Seguridad Ordenes de Pago (Gestor de Cuenta - Software)

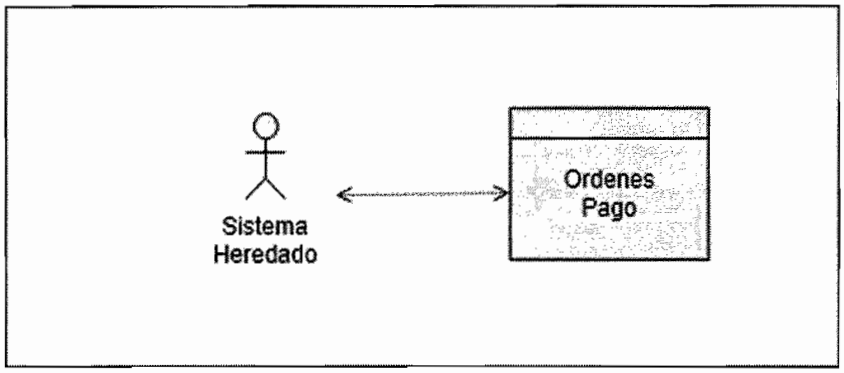

Figura 131 - Diagrama de Seguridad Ordenes de Pago (Sistema Legacy)

	Gestor de Cuenta (Software)				Sistema Legacy			
Entidad	C	R	U	D	C	R	U	D
OrdenesPago					X	X	X	
MovimientosCuenta	X	X	X					
CuentaUnica	X	X	X					

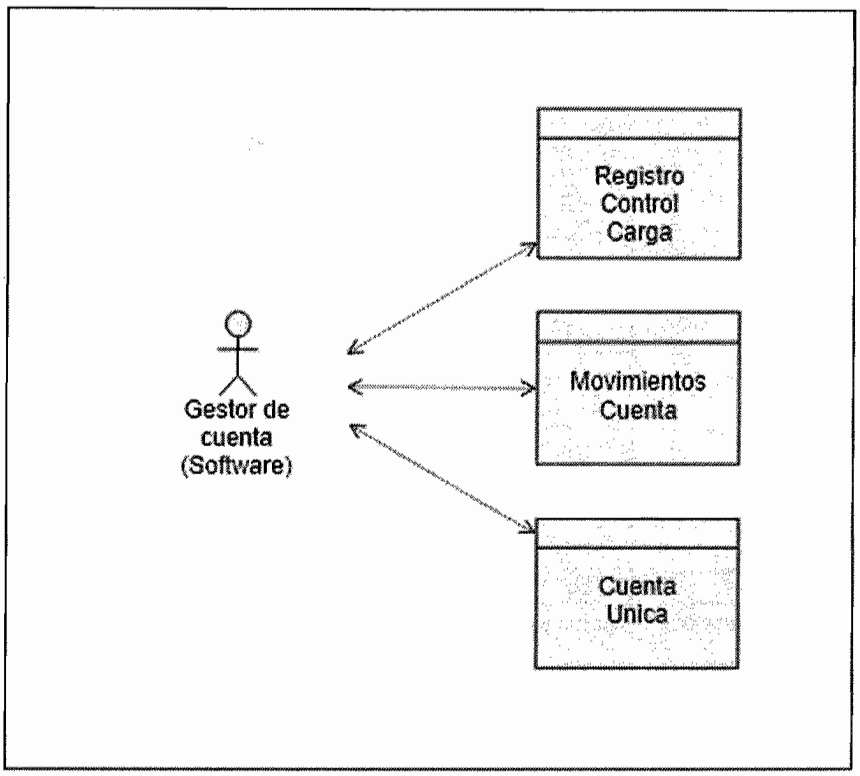

Figura 132 - Diagrama de Seguridad Comprobantes Electrónicos (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
MovimientosCuenta	X	X	X	
CuentaUnica	X	X	X	
RegistroControlCarga	X	X	X	

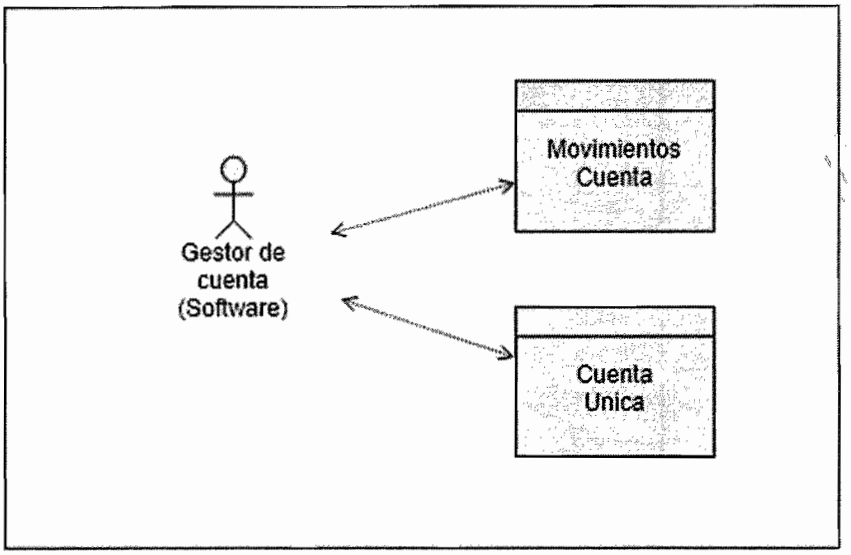

Figura 133 - Diagrama de Seguridad Recepción de Pagos (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
MovimientosCuenta	X	X	X	
CuentaUnica	X	X	X	

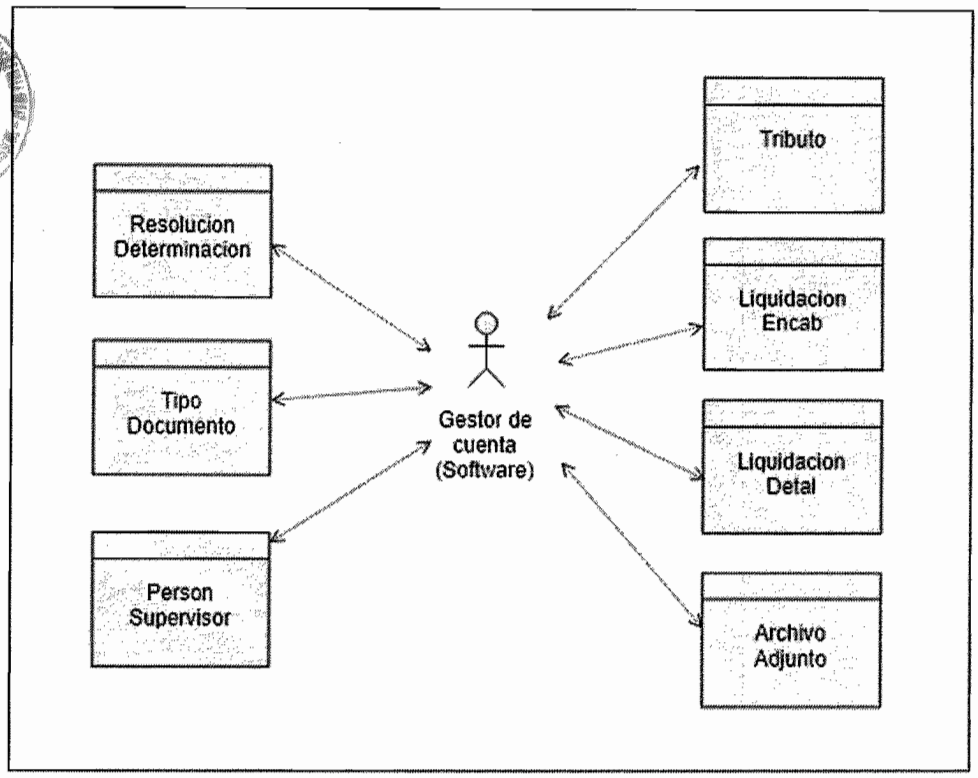

Figura 134 - Diagrama de Seguridad Liquidación de Fiscalización (Gestor de Cuenta - Software)

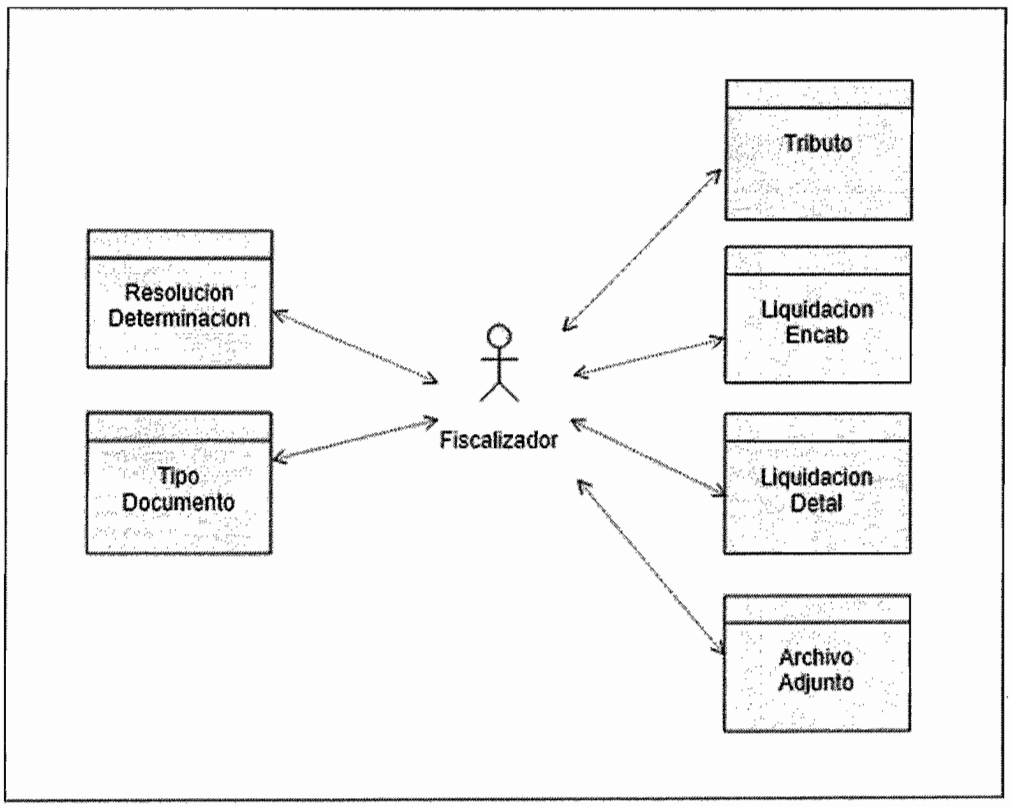

Figura 135 - Diagrama de Seguridad Liquidación de Fiscalización (Fiscalizador)

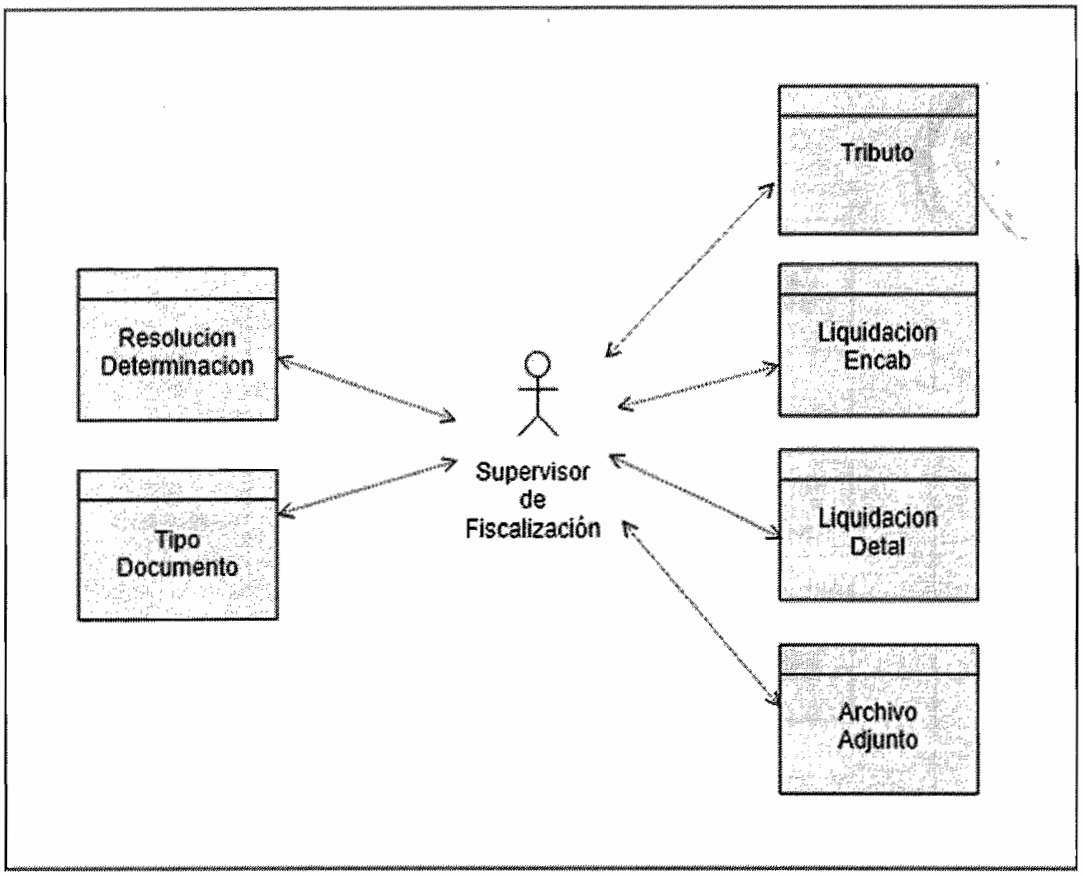

Figura 136 - Diagrama de Seguridad Liquidación de Fiscalización (Supervisor de Fiscalización)

	Gestor de Cuenta (Software)				Fiscalizador				Supervisor de Fiscalización			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D
Tributo		X				X				X		
LiquidacionEncab		X			X	X	X			X	X	
LiquidacionDetal		X			X	X	X	X		X	X	
ArchivoAdjunto		X			X	X	X	X		X	X	
ResolucionDeterminacion		X			X	X	X			X	X	
TipoDocumento		X				X				X		
PersonSupervisor		X										

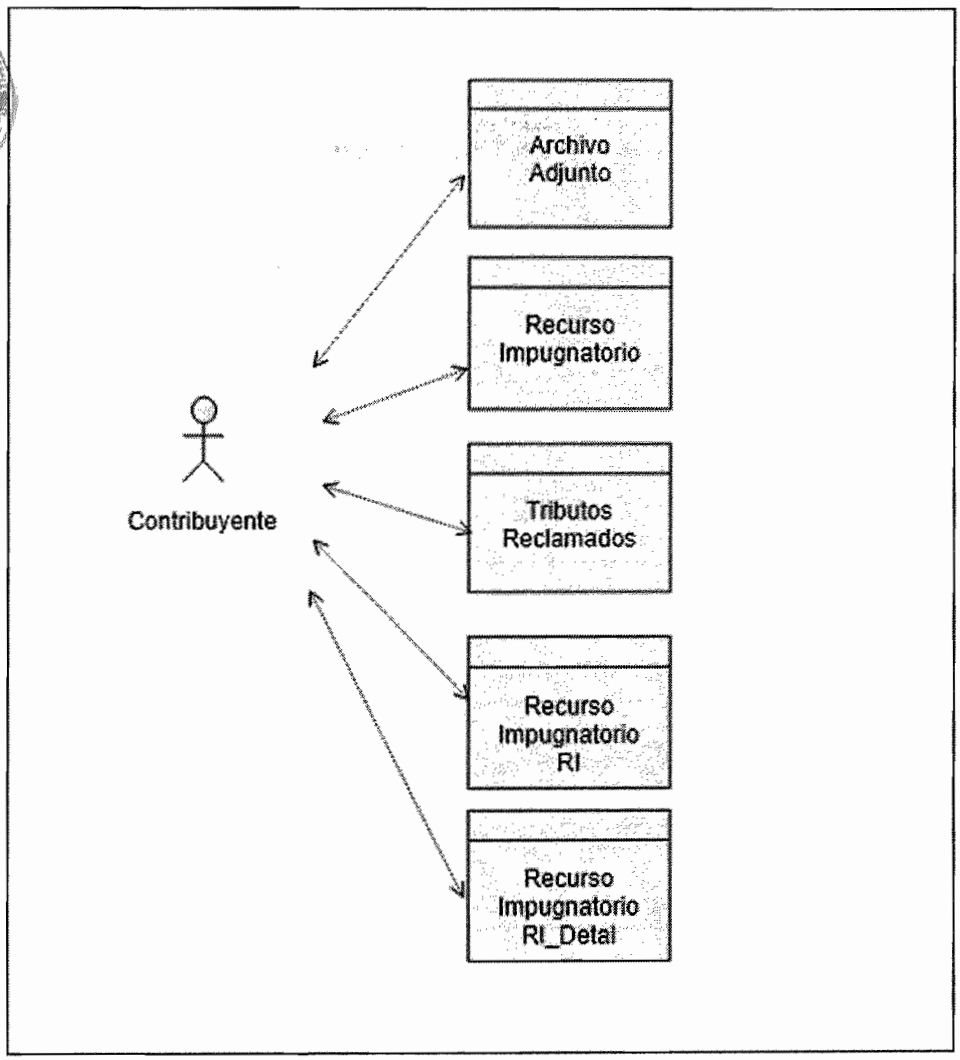

Figura 137 - Diagrama de Seguridad Recursos Impugnatorios (Contribuyente)

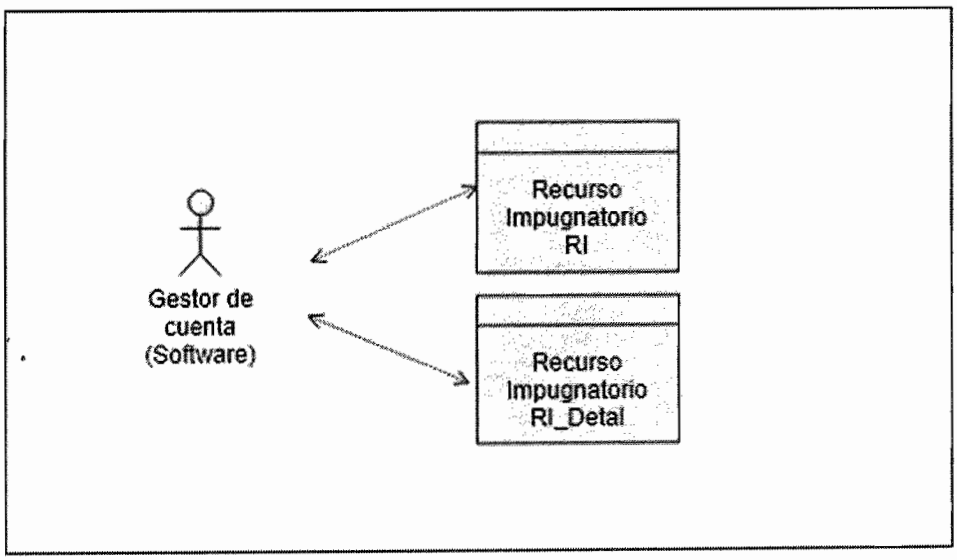

Figura 138 - Diagrama de Seguridad Recursos Impugnatorios (Gestor de Cuenta - Software)

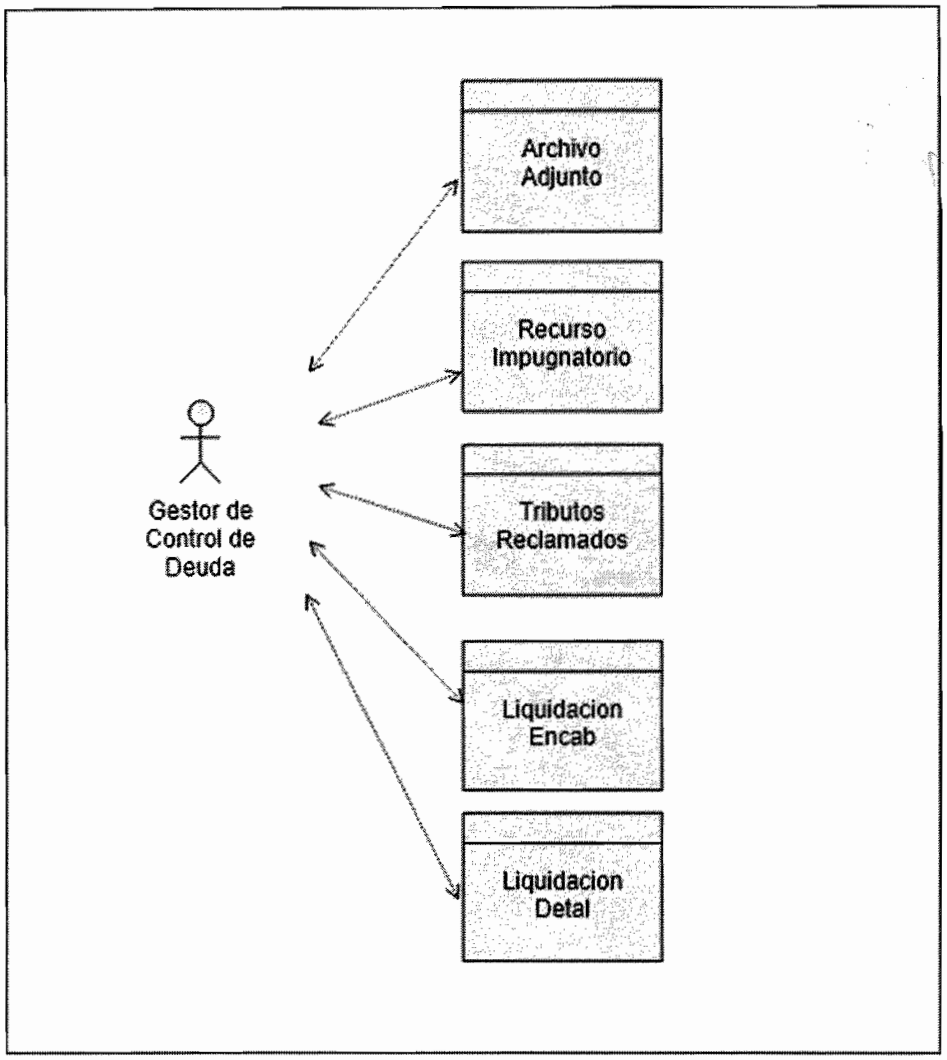

Figura 139 - Diagrama de Seguridad Recursos Impugnatorios (Gestor de Control de Deuda)

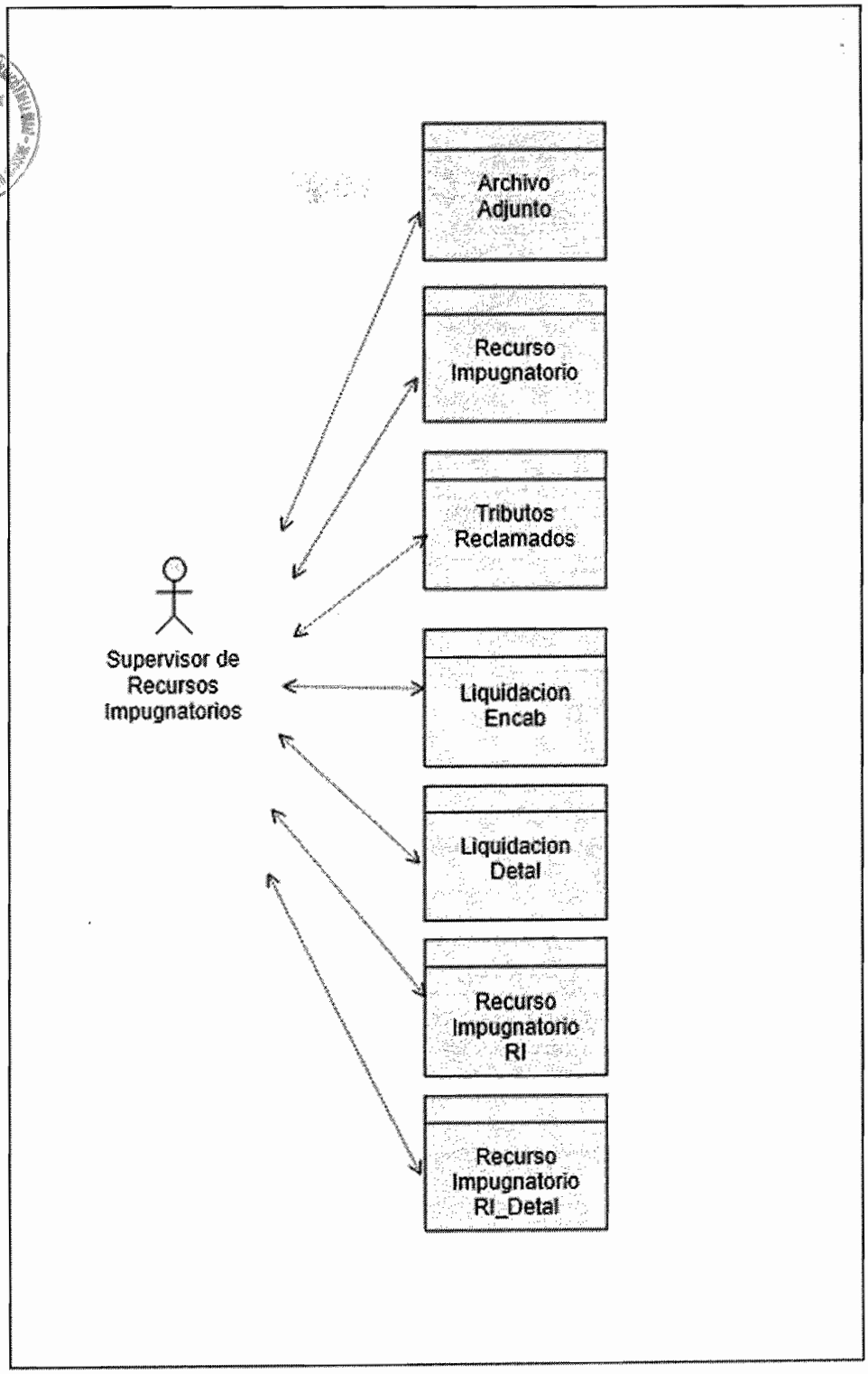

Figura 140 - Diagrama de Seguridad Recursos Impugnatorios (Supervisor de Recursos Impugnatorios)

Entidad	Contribuyente				Gestor de Cuenta (Software)				Gestor de Control de Deuda				Supervisor de Recursos Impugnatorios			
	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
ArchivoAdjunto	X	X	X	X					X	X	X	X		X		
RecursoImpugnatorio	X	X	X							X				X	X	
TributosReclamados	X	X								X				X		
LiquidacionEncab									X	X	X			X	X	
LiquidacionDetal									X	X	X	X		X		
RecursoImpugnatorioRI		X				X							X	X	X	
RecursoImpugnatorioRI_Detal		X				X							X	X	X	X

EP031 - PROCESO CONCURSAL

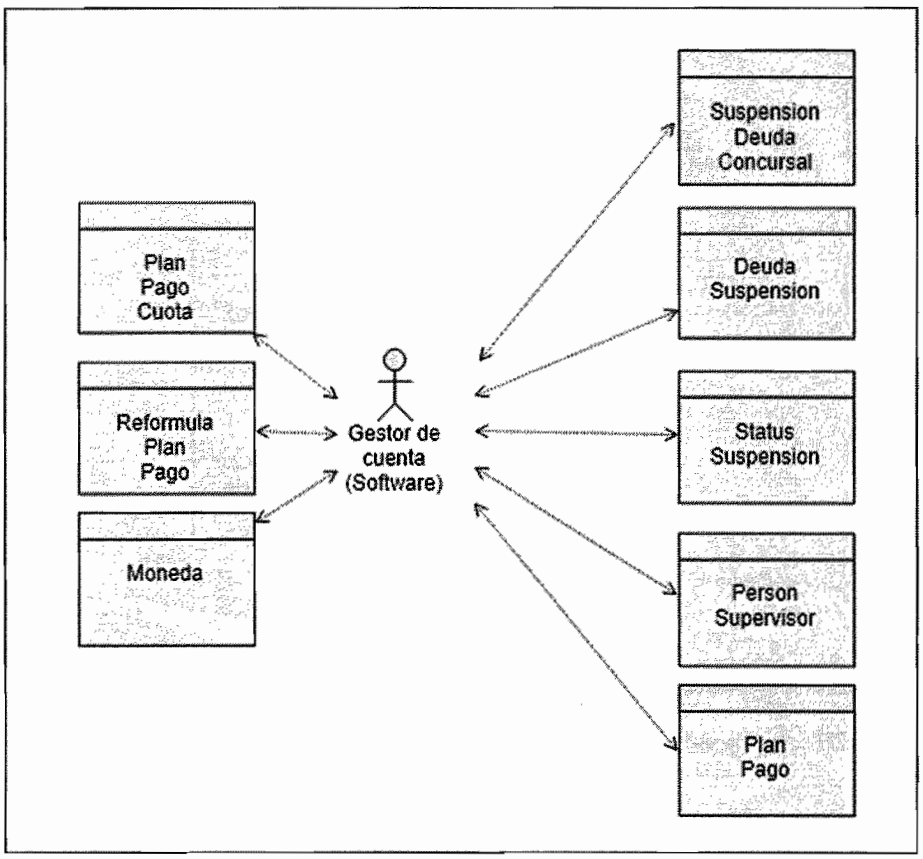

Figura 141 - Diagrama de Seguridad Proceso Concursal (Gestor de Cuenta - Software)

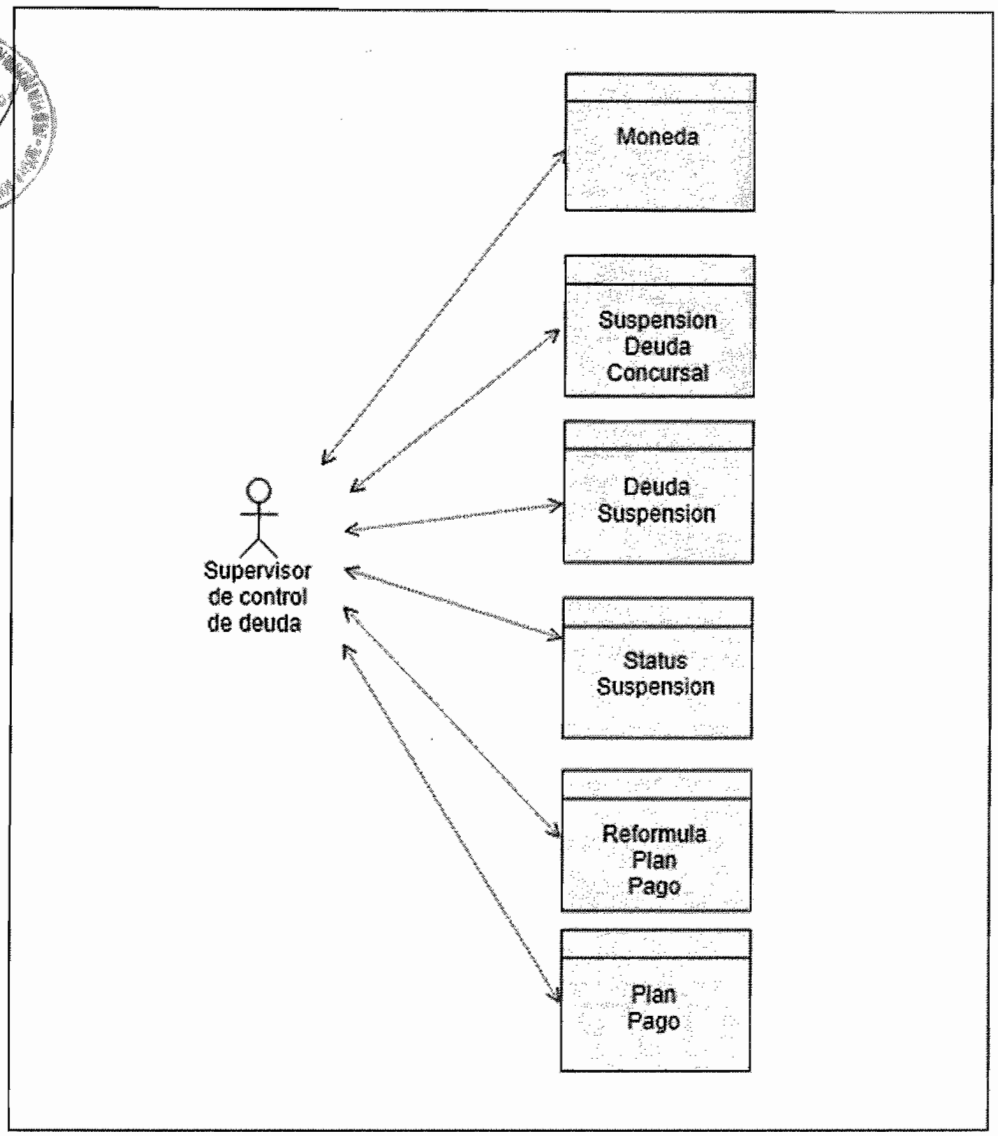

Figura 142 - Diagrama de Seguridad Proceso Concursal (Supervisor de Control de Deuda)

	Gestor de Cuenta (Software)				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R	U	D
SuspensionDeudaConcursal		X			X			
DeudaSuspension		X			X			
StatusSuspension		X				X		
PersonSupervisor		X						
PlanPago		X			X	X	X	
PlanPagoCuota	X	X						
ReformulaPlanPago		X			X			
Moneda		X				X		

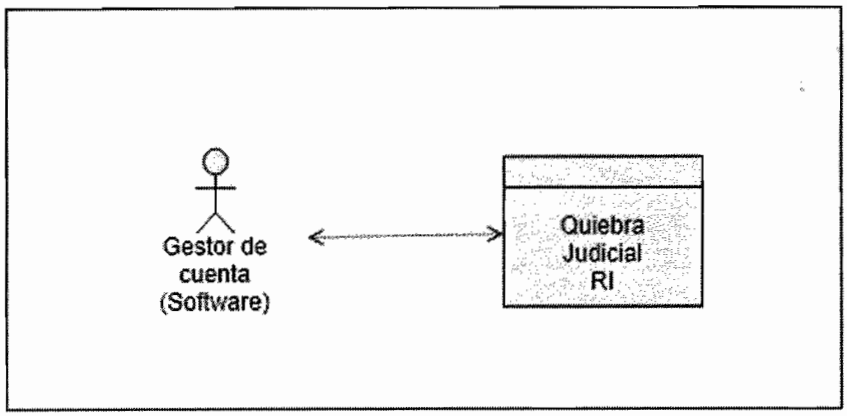

Figura 143 - Diagrama de Seguridad Quiebra Judicial (Gestor de Cuenta - Software)

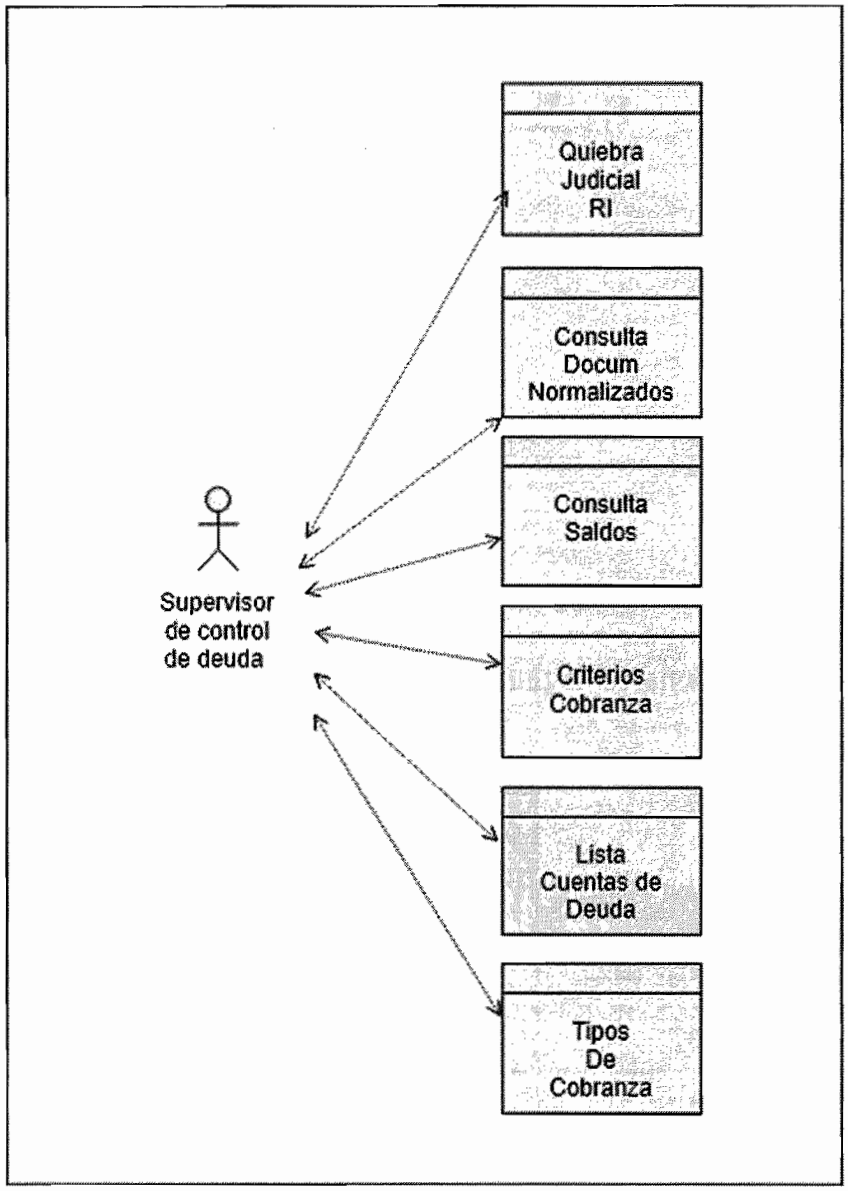

Figura 144 - Diagrama de Seguridad Quiebra Judicial (Supervisor de Control de Deuda)

	Gestor de Cuenta (Software)				Supervisor de Control de Deuda			
Entidad	C	R	U	D	C	R		D

QuiebraJudicialRI		X			X			
Consulta Docum Normalizados						X		
Consulta Saldos						X		
CriteriosCobranza						X		
Lista Cuentas de Deuda						X		
TiposDeCobranza						X		

EP033 - Cobranza coactiva

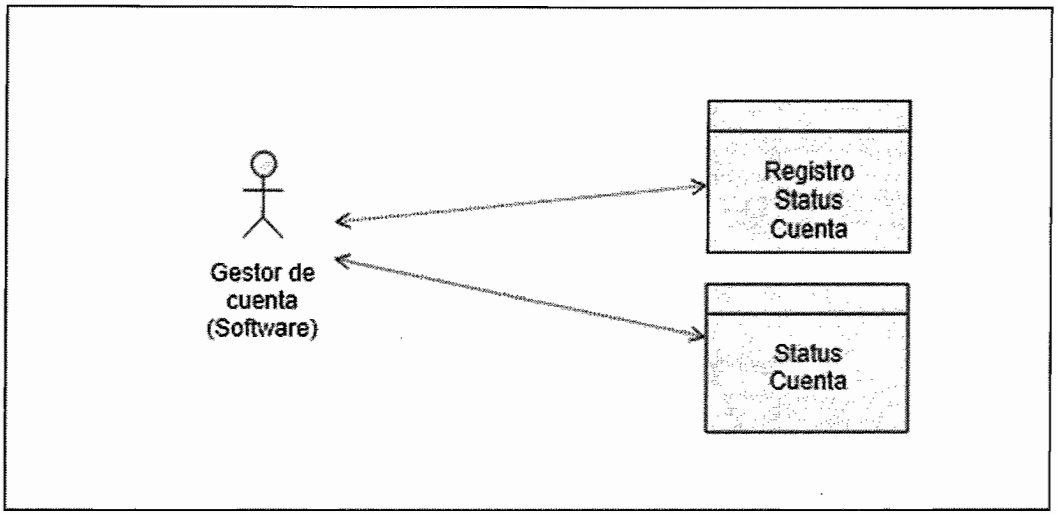

Figura 145 - Diagrama de Seguridad Cobranza Coactiva (Gestor de Cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
RegistroStatusCuenta	X	X	X	
StatusCuenta		X		

EP035 – Despacho

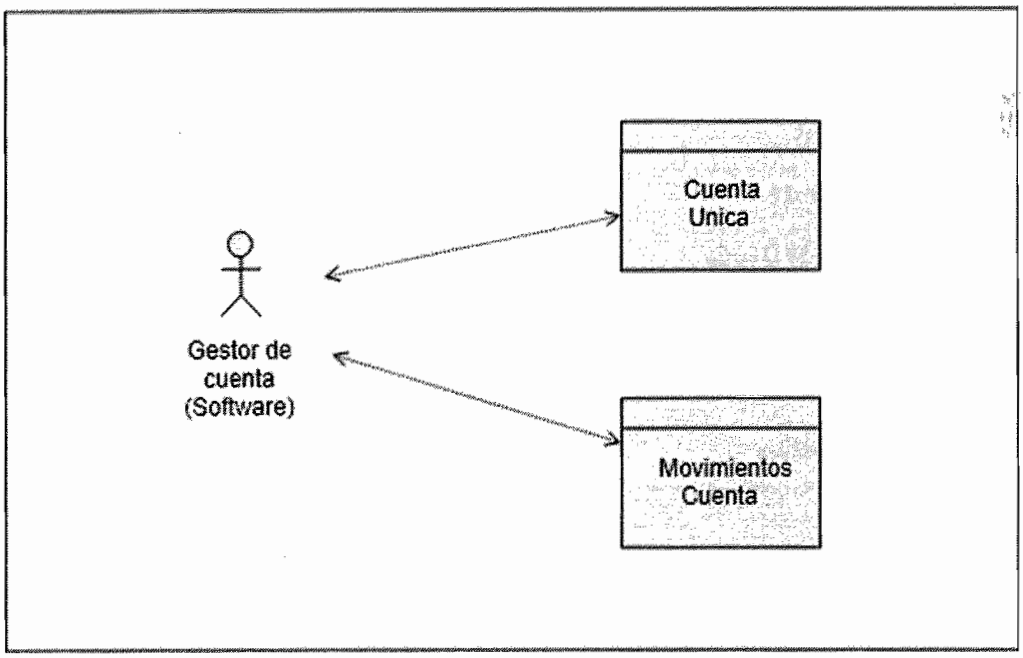

Figura 146 - Diagrama de Seguridad Despacho (Gestor de cuenta - Software)

	Gestor de Cuenta (Software)			
Entidad	C	R	U	D
MovimientosCuenta	X	X	X	
CuentaUnica	X	X	X	

EP037 – Infracciones

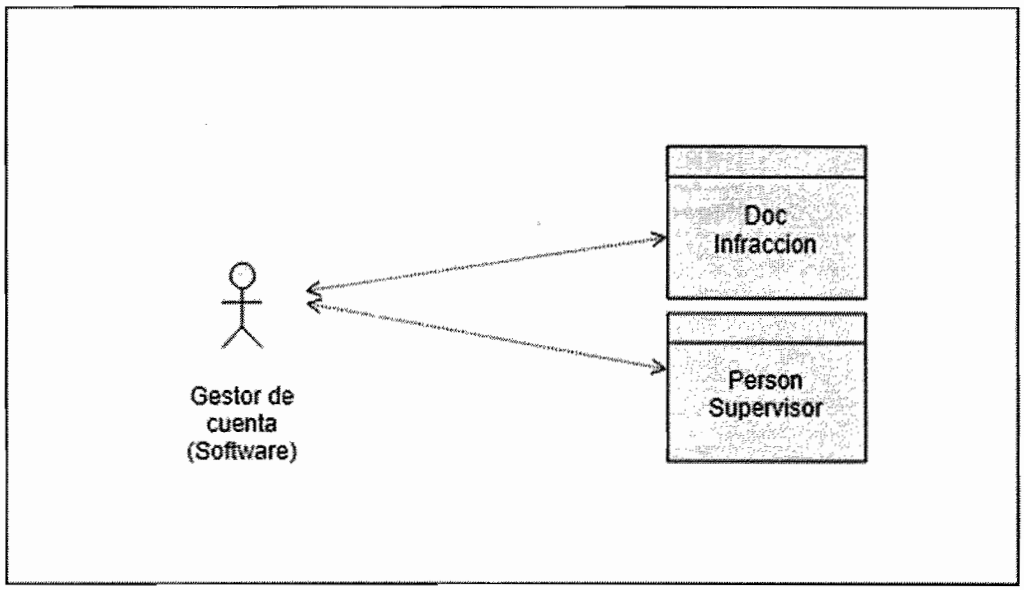

Figura 147 - Diagrama de Seguridad Infracciones (Gestor de Cuenta - Software)

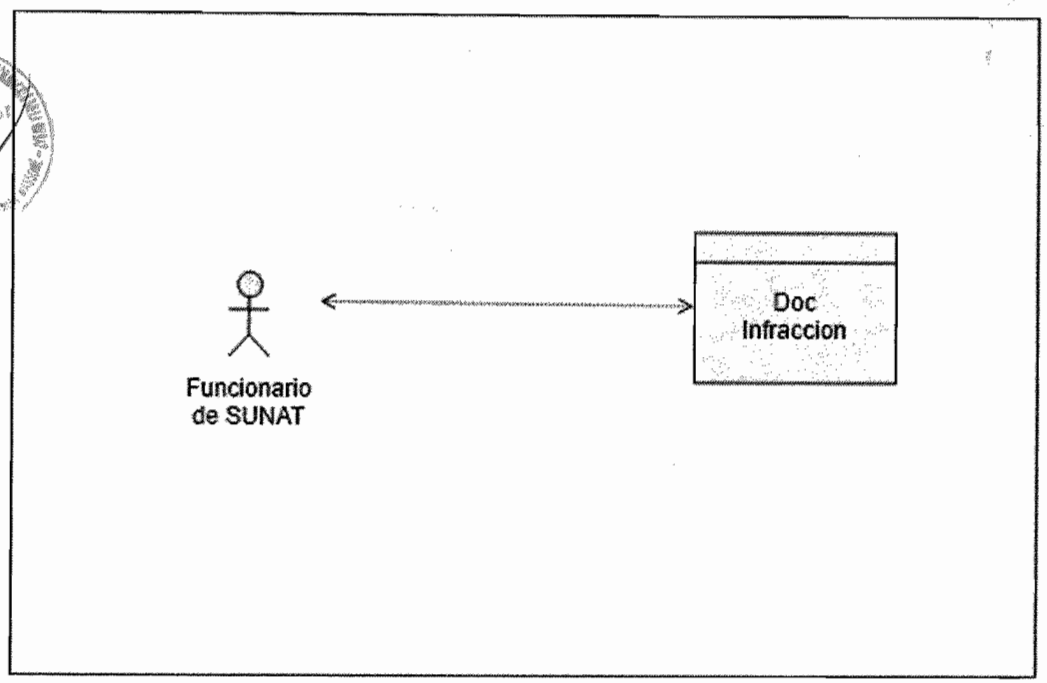

Figura 148 - Diagrama de Seguridad Infracciones (Funcionario de SUNAT)

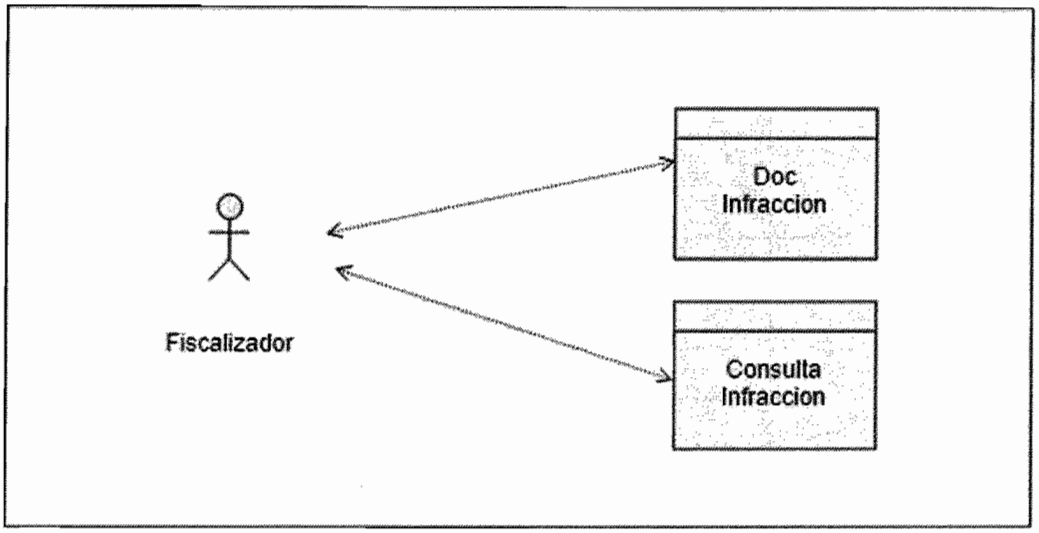

Figura 149 - Diagrama de Seguridad Infracciones (Fiscalizador)

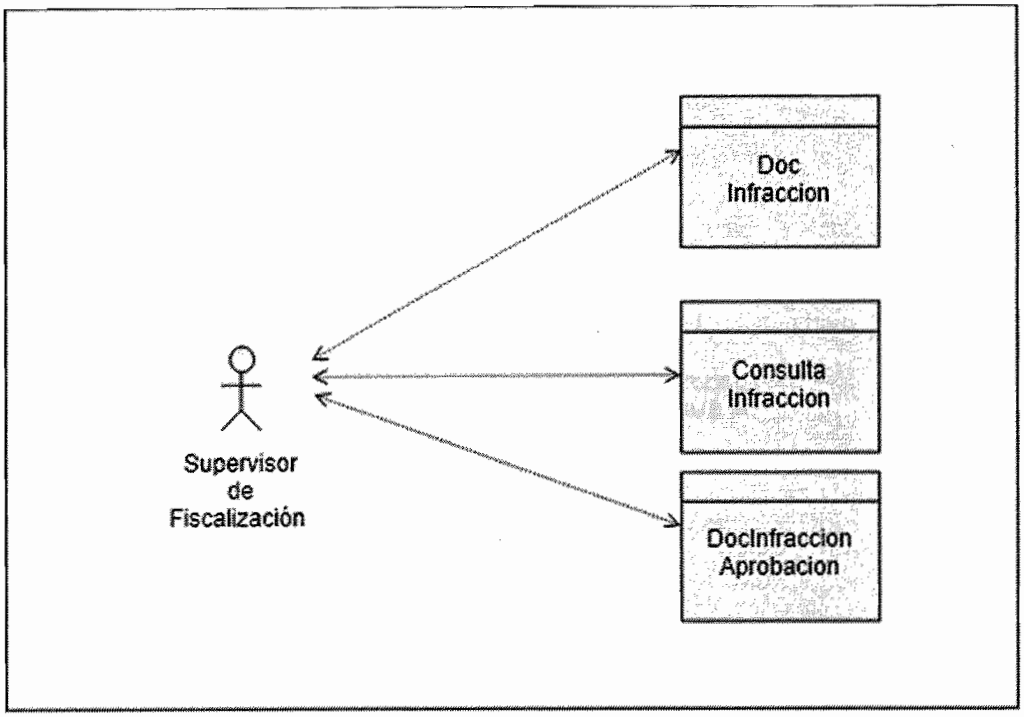

Figura 150 - Diagrama de Seguridad Infracciones (Supervisor de Fiscalización)

	Gestor de Cuenta (Software)				Funcionario de SUNAT				Fiscalizador				Supervisor de Fiscalización			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
DocInfraccion	X					X			X	X	X			X	X	
Consulta Infraccion										X				X		
DocInfraccionAprobacion													X			
PersonSupervisor		X														

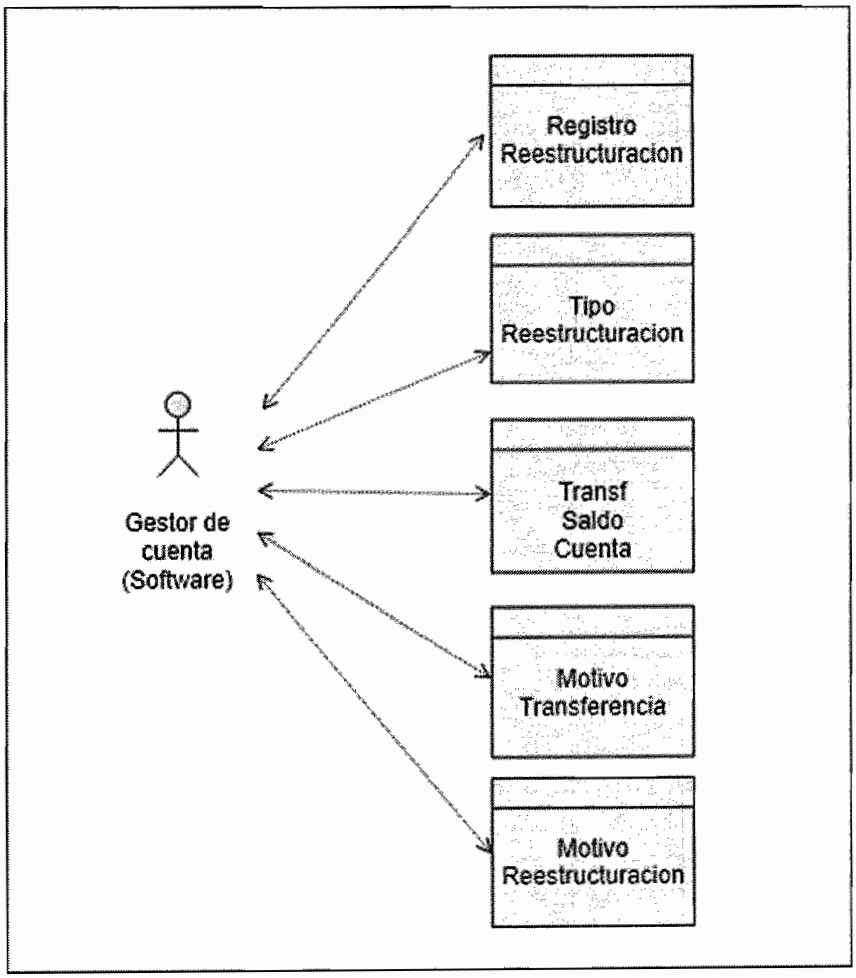

Figura 151 - Diagrama de Seguridad Reestructuración o Reorganización (Gestor de Cuenta - Software)

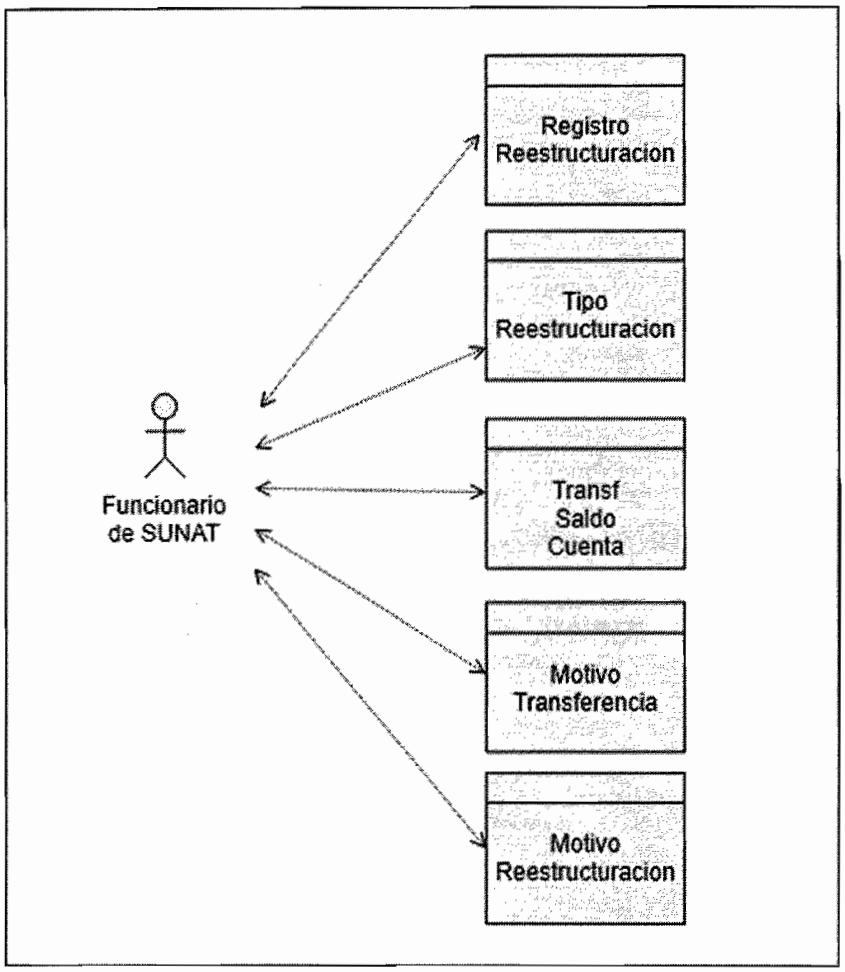

Figura 152 - Diagrama de Seguridad Reestructuración o Reorganización (Funcionario de SUNAT)

	Gestor de Cuenta (Software)				Funcionario de SUNAT			
Entidad	C	R	U	D	C	R	U	D
RegistroReestructuracion		X			X			
TipoReestructuracion		X				X		
TransfSaldoCuenta		X			X			
MotivoTransferencia		X			X			
MotivoReestructuracion		X			X			

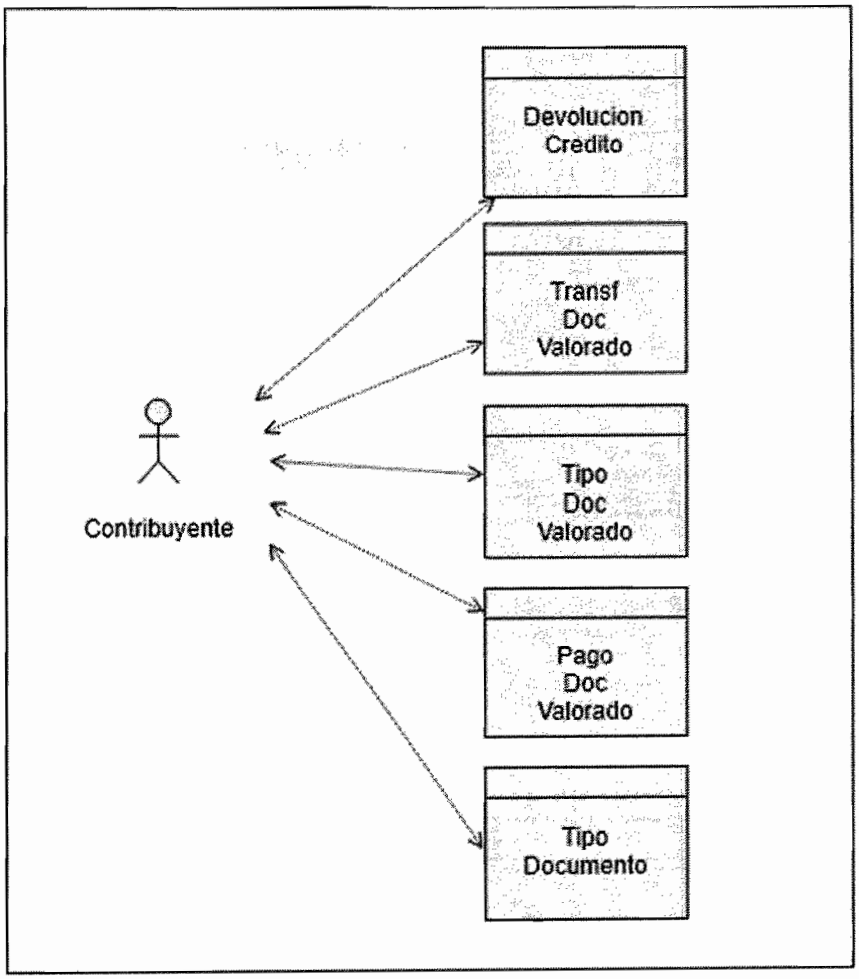

Figura 153 - Diagrama de Seguridad Documentos Valorados Electrónicos (Contribuyente)

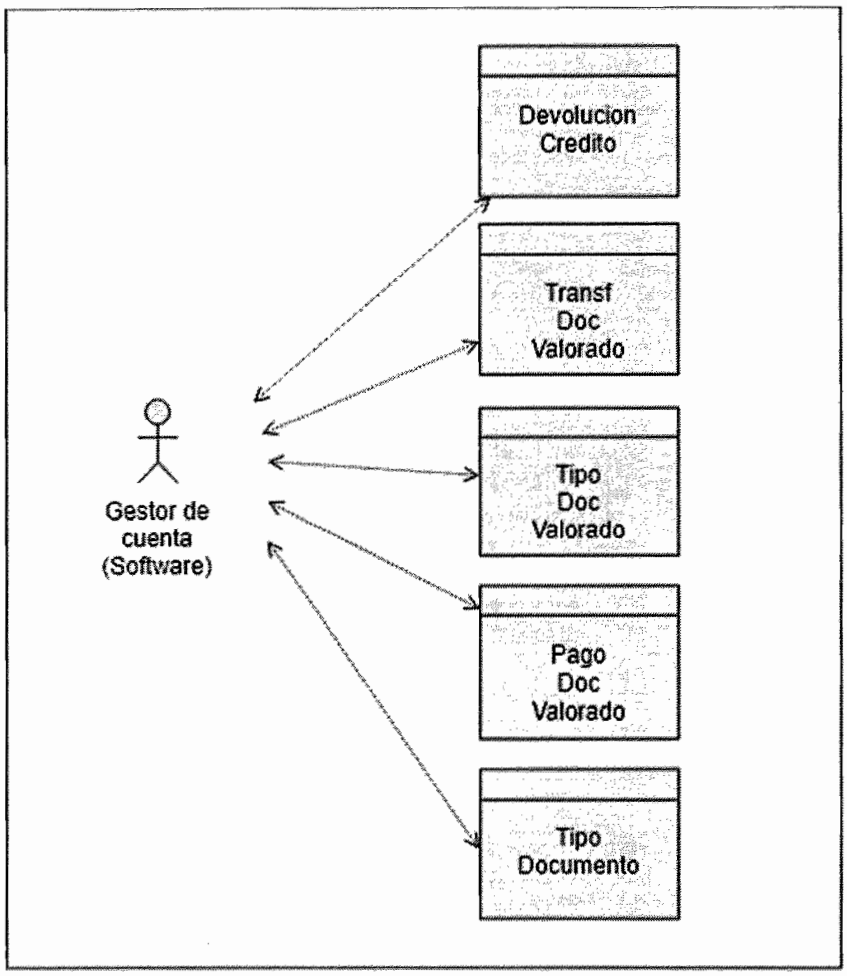

Figura 154 - Diagrama de Seguridad Documentos Valorados Electrónicos (Gestor de Cuenta - Software)

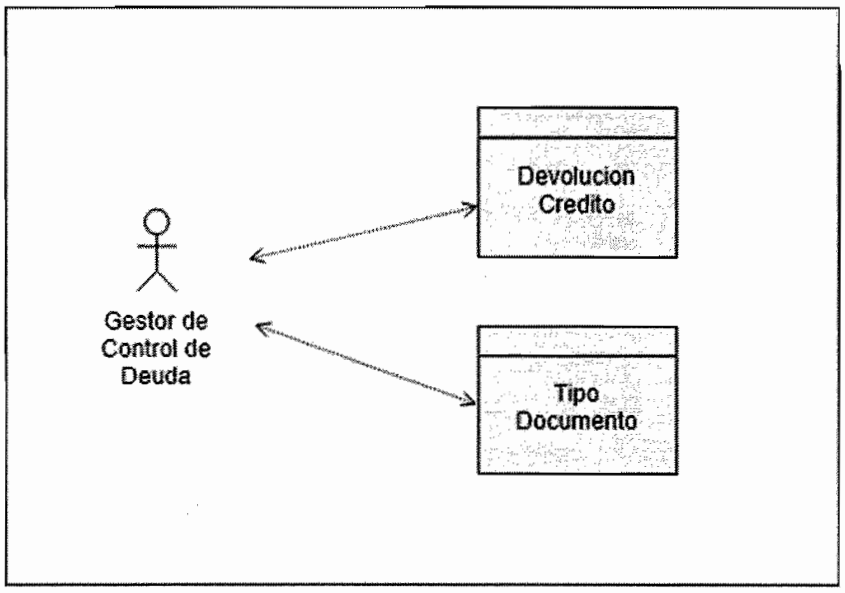

Figura 155 - Diagrama de Seguridad Documentos Valorados Electrónicos (Gestor de Control de Deuda)

Entidad	Contribuyente				Gestor de Cuenta (Software)				Gestor de Control de Deuda			
	C	R	U	D	C	R	U	D	C	R	U	D
DevolucionCredito		X				X			X			
TransfDocValorado	X	X				X						
TipoDocValorado		X				X						
PagoDocValorado	X	X				X						
TipoDocumento		X				X				X		

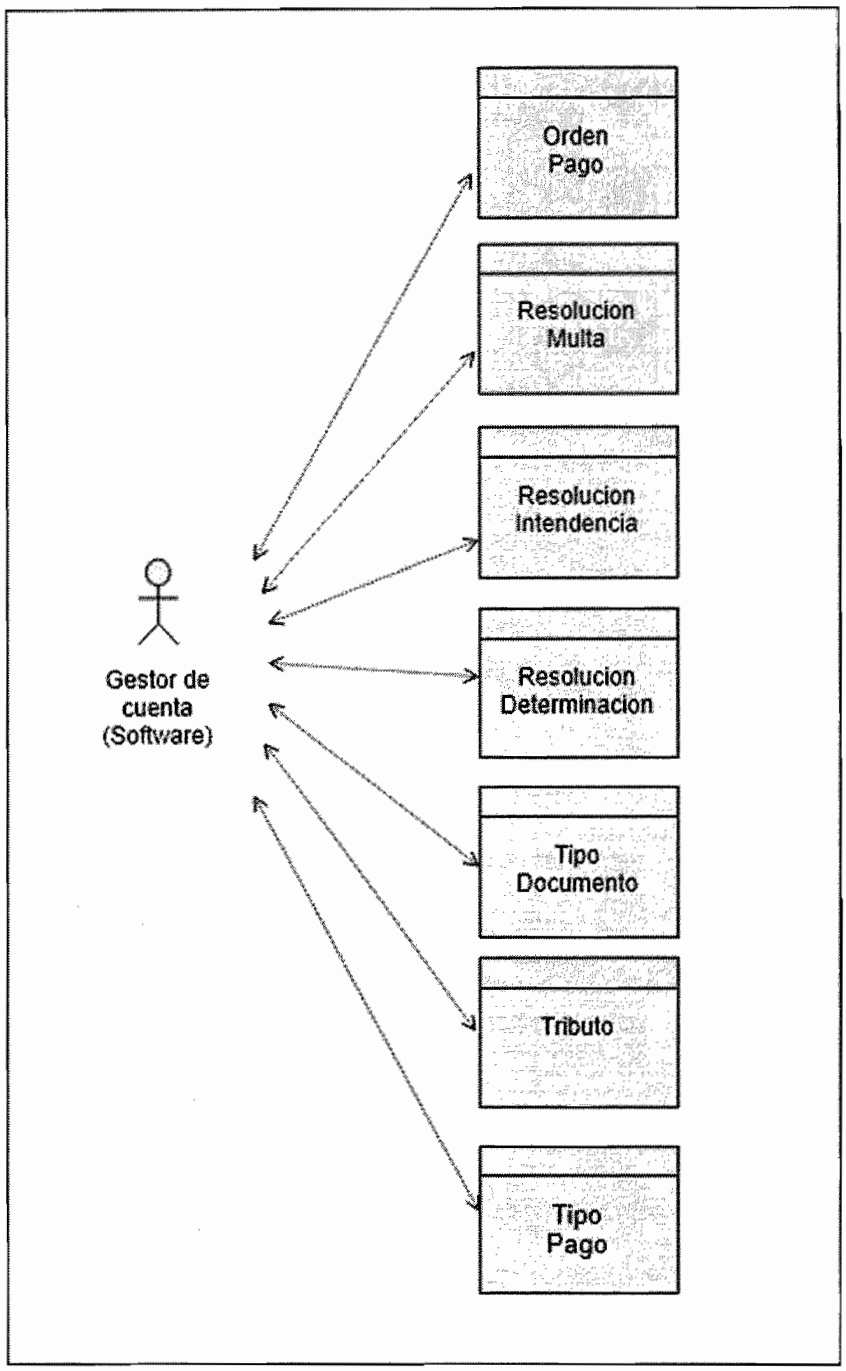

Figura 156 - Diagrama de Seguridad Registro Manual de Valores (Gestor de Cuenta - Software)

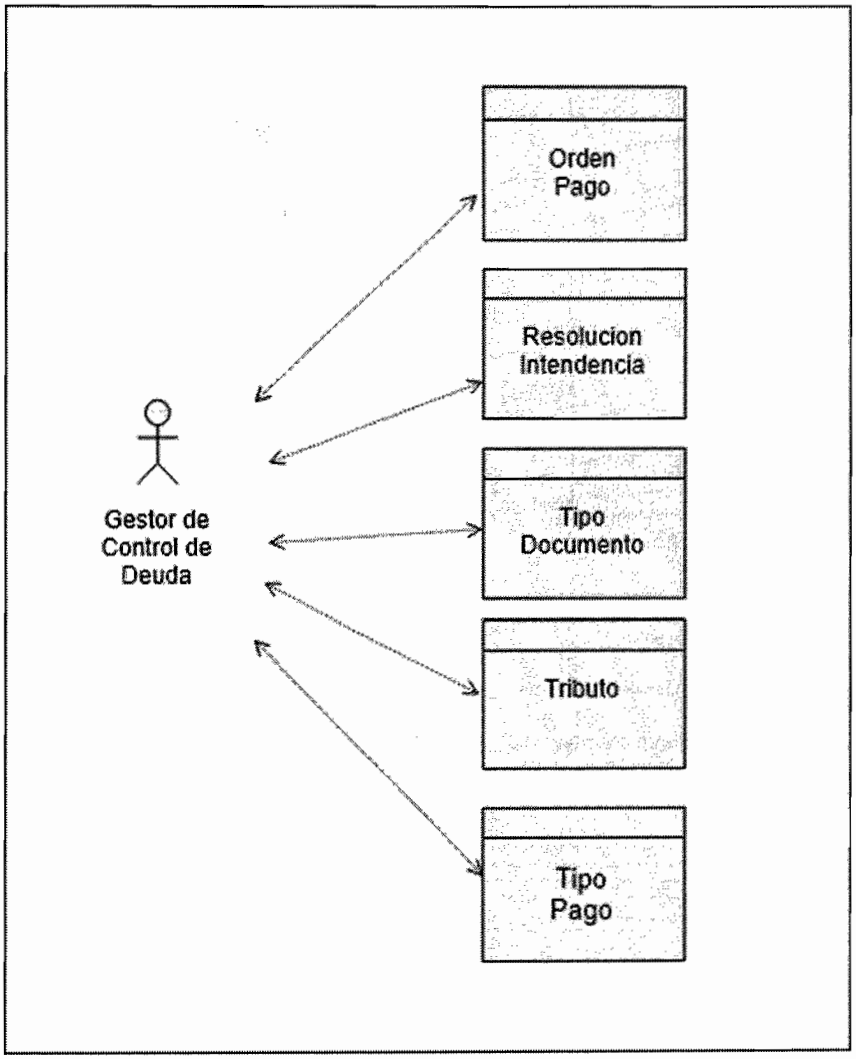

Figura 157 - Diagrama de Seguridad Registro Manual de Valores (Gestor de Control de Deuda)

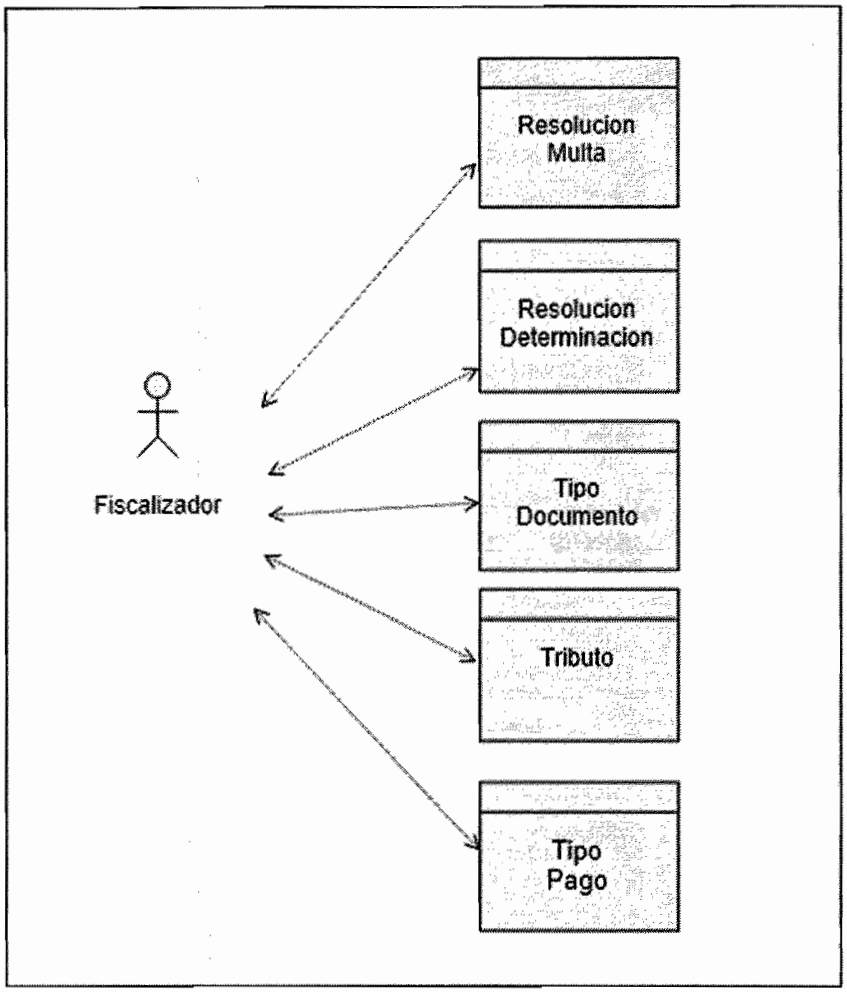

Figura 158 - Diagrama de Seguridad Registro Manual de Valores (Fiscalizador)

	Gestor de Cuenta (Software)				Gestor de Control de Deuda				Fiscalizador			
Entidad	C	R	U	D	C	R	U	D	C	R	U	D
OrdenPago		X			X	X						
ResolucionMulta		X							X	X		
ResolucionIntendencia		X			X	X						
ResolucionDeterminacion		X							X	X		
TipoDocumento		X				X				X		
Tributo		X				X				X		
TipoPago		X				X				X		