Universidad Nacional de La Plata
Facultad de Informática

Seguridad en entornos BPM:
Integrando firma digital en procesos con altos requerimientos de autenticación e integridad

Alumno
Ivan Grcevic

Directoras
Lic. Patricia Bazán - Lic. Paula Venosa

Tesis presentada para obtener el grado de Licenciado en Sistemas

Agradecimientos

Quiero dedicar especialmente esta tesina a mis directoras Patricia Bazán y Paula Venosa, por todo su apoyo, paciencia y ayuda, por guiarme y acompañarme en estos últimos años. A mi asesor profesional Jose Martinez Garro, por asesorarme durante mi investigación. También me gustaría agradecerle a mis amigos Luciano Lorenti y María Emilia Charnelli por acompañarme desde el inicio de la carrera y especialmente por haber sido mi apoyo incondicional en los últimos meses. Por último, agradecer a la Universidad Nacional de La Plata, por la gran oportunidad que brinda gratuitamente para la formación académica de nuevos profesionales.

Índice general

Índice de figuras

Capítulo 1

Introducción

En la presente tesina, me propongo integrar conceptos de criptografía y BPM (Business Process Management). La integración que propongo es en ambos sentidos. Por un lado, la seguridad en un entorno de BPM debe verse aumentada gracias a la criptografía. Por otro lado, la administración necesaria para mantener un esquema de seguridad de mecanismos criptográficos, debe verse simplificada al estar orientada a BPM e integrada con otros procesos de una determinada organización.

En particular, los mecanismos de seguridad y criptografía que se utilizan en este trabajo son las conexiones TLS, el protocolo HTTPS, la criptografía de clave pública y la firma digital. Como la administración de un esquema de criptografía de clave pública es compleja, se pretende también que la forma en que se integren con los procesos de BPM sea lo más transparente posible, reduciendo el impacto en la dificultad de uso de los sistemas de información.

1.1. Motivación

La informática es desde hace mucho tiempo una aliada importante para las organizaciones, pero originalmente los sistemas de software sólo mejoraban el rendimiento del nivel operativo y el almacenamiento y posterior acceso a la información. El concepto de BPM se encuentra muy extendido en el mundo de las organizaciones y hoy por hoy una gran cantidad de estas funcionan orientadas a procesos, es decir, planifican, documentan, ejecutan y mejoran sus procesos en un ciclo de mejora continua, y realizan su actividad como un conjunto de tareas relacionadas entre sí, asociadas cada una a un cierto rol y orquestadas de manera de cumplir un objetivo principal. [16]

Los BPMS (BPM Suites) son herramientas que van más allá de esto. Son piezas de software que pasan a ser transversales: atraviesan los niveles operativo, táctico y estratégico de la organización. Asisten en las distintas etapas del modelo de administración por procesos, desde el diseño de los procesos, pasando por la implementación, ejecución, orquestación y hasta la verificación del cumplimiento de los objetivos.

En los BPMS, típicamente la autenticación de los usuarios se realiza a través de un usuario y una contraseña. Cada uno tiene acceso a completar distintas tareas dependiendo de su grupo, rol y permisos.[4] Algunos usuarios pueden administrar aspectos del sistema, como los procesos, usuarios, grupos, permisos, e instancias. Muchas veces los BPMS también permiten también auditar el rendimiento de los distintos usuarios y grupos.[17]

La información es el bien más importante de toda organización, y su protección es vital para muchas de ellas, como aseguradoras, bancos, y prestadores de todo tipo de servicios, cuyos procesos requieren un

nivel de seguridad mayor que una simple autenticación a través de usuario y contraseña.[11] Los procesos de este tipo de sistemas requieren dar un paso más en cuanto a seguridad: tener la garantía de integridad de los datos que se intercambian y una autenticación más fuerte.[9]

Por otro lado, la firma digital es una tecnología bien conocida que brinda garantía de autenticidad, integridad y no repudio a los datos que se intercambian en sistemas distribuidos.[15]

Un BPMS funciona como un sistema distribuido, ya que múltiples usuarios acceden a él e interactúan participando como actores de los procesos, cada uno desde su dispositivo (sea una computadora, una tablet o un smartphone). Es frecuente el acceso desde distintos lugares físicos, incluso fuera de una red privada organizacional y sin estar conectado a una VPN. Parte del BPMS se ejecuta en cada dispositivo cliente, mientras que el núcleo y el motor de procesos se ejecuta en el o los servidores. Es de interés investigar cómo una tecnología como la firma digital puede enriquecer los procesos de negocio, especialmente aquellos en los cuales el tipo de seguridad provisto por la mayoría de los BPMS no es suficiente.[10]

Un proceso se compone de un conjunto de tareas o actividades llevadas a cabo por los participantes de las mismas. Este concepto llevado a un BPMS puede enriquecerse incorporando los conceptos de firma digital, que garantiza que los datos han llegado al sistema tal como fueron ingresados (integridad), que el usuario que completó la tarea es realmente quien dice ser (autenticación) y que no pueda evadir responsabilidades sobre la tarea completada (no repudio).

De esta manera no sólo se aplicaría la firma digital para proteger documentos como mensajes, facturas, transacciones comerciales, notificaciones, decretos. Este tipo de integración se podría llevar a cabo utilizando cualquier herramienta conocida para firmar un documento de tipo PDF y adjuntarlo en un proceso cualquiera un BPMS. En lugar de esto, se pretende que cualquier tarea completada, pueda ser firmada digitalmente por la persona que la completa, desde cargar datos de una venta hasta aprobar un presupuesto.

1.2. Objetivo

El objetivo de este trabajo consiste en realizar una investigación en los campos del BPM y la firma digital, centrada en el análisis de la integración de estas dos tecnologías, con el fin de que la firma digital enriquezca los componentes de BPM y a su vez los conceptos de BPM se puedan aplicar al sistema de gestión de seguridad, en particular al proceso de gestión de claves y su aplicaciòn para la firma. Se pretende plasmar las ideas planteadas durante la investigación en un proceso de ejemplo que ilustre este enriquecimiento y que permita analizar la aplicación de los estos conceptos.

Es de especial interés realizar el trabajo utilizando una herramienta de BPM de código fuente abierto, y a su vez mantener un equilibrio entre dos aspectos que suelen ser incompatibles: por un lado el nivel de transparencia y simplicidad para el usuario final y por otro lado un alto nivel de agregado de seguridad.

1.3. Estructura de esta tesina

Esta tesina está estructurada en 6 capítulos. El capítulo 1 es la presente introducción. En los capítulos 2 y 3 se provee un marco teórico para el resto del trabajo con el fin de introducir a todos los temas que son la base para los capítulos siguientes. Se abordan temas de criptografía en el capítulo 1 y de BPM en el capítulo 3.

En el capítulo 4 se desarrolla la idea de integración propuesta en esta introducción, teniendo en cuenta el objetivo propuesto. Además, se describe los procesos de un caso de estudio vinculado a una organización. Para este caso de estudio, realicé una entrevista a un empleado de la empresa, para entender su modelo de negocio y procesos fundamentales.

En el capítulo 5 se describe la implementación de una prueba de concepto basada en los procesos centrales del caso de estudio planteado en el capítulo 4. Esta implementación permitió, durante su desarrollo, volver sobre las ideas iniciales, pulir los conceptos esbozados y extraer conclusiones.

El capítulo 6 enuncia las conclusiones obtenidas y propone la posibilidad de realizar un trabajo futuro vinculado al tema.

Capítulo 2

Marco teórico y conceptual de criptografía y firma digital

La **criptografía** se ha definido tradicionalmente como el ámbito de la criptología que se ocupa de las técnicas de cifrado o codificado destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos ininteligibles a receptores no autorizados. Criptología es un término más general, y se refiere a la disciplina científica que se dedica al estudio de la escritura secreta, es decir, escritura de mensajes que, procesados de cierta manera, se convierten en difíciles o imposibles de leer por entidades no autorizadas.

La aparición de las **Tecnologías de la Información y la Comunicación** y el uso masivo de las comunicaciones digitales ha producido un número creciente de problemas de seguridad. Los mensajes que se intercambian a través de la red pueden ser interceptados, y por tanto, la seguridad de la información debe garantizarse. Este desafío ha generalizado los objetivos de la criptografía para ser la parte de la criptología que se encarga del **estudio de los algoritmos, protocolos, y sistemas que se utilizan para proteger la información** y dotar de seguridad a las comunicaciones y a las entidades que se comunican.

2.1. Conceptos de seguridad en sistemas

En seguridad informática, los tres conceptos principales son confidencialidad, integridad y disponibilidad, tal como se enuncia en el capítulo 1 de [15]. Este conjunto es conocido como la tríada CIA (Confidentiality, Integrity, Availability).

Confidencialidad se define como la preservación del acceso autorizado a la información, incluyendo medios para proteger la privacidad y la información privada.

Integridad se define como la protección contra la modificación inapropiada o destrucción de la información, incluyendo el aseguramiento del no repudio y autenticidad de la información.

Disponibilidad se define como el aseguramiento del acceso confiable y a tiempo a la información. Una pérdida de disponibilidad implica la interrupción del acceso o el uso de la información o del sistema que la manipula.

Se consideran tres niveles de impacto que tendría una brecha de seguridad respecto a uno o más de los conceptos mencionados. Estos son:

Bajo: Cuando el impacto puede tener un efecto adverso limitado en las operaciones de la organización, o sobre los recursos de la organización o sus individuos. La pérdida de confidencialidad, integridad o disponibilidad en este caso podría causar degradación de la capacidad de llevar a cabo la misión de la organización con un alcance y duración que no impida a la misma realizar sus funciones básicas, pero que conlleve eun efectividad notablemente reducida. Podría resultar en daños menores a los recursos de la organización. Podría resultar en una pérdida financiera menor. O en un daño menor a individuos.

Moderado: Un impacto moderado implicaría un efecto adverso serio en las operaciones de la organización, sus recursos o individuos. Por ejemplo una degradación significativa en las capacidades de llevar a cabo la misión de la organización, hasta un punto en el cual si bien la misma puede realizar sus funciones básicas, sufre una importante reducción en su efectividad. Podría resultar también en un daño significativo a los recursos de la misma, una significativa pérdida económica o un daño importante a individuos, sin involucrar la pérdida de vida o enfermedades serias.

Alto: Un impacto alto implicaría efectos adversos catastróficos para las operaciones de la organización, sus recursos o individuos. Por ejemplo, si el daño causara una degradación severa o pérdida total de la capacidad de llevar a cabo la misión de la organización, que implique la imposibilidad de realizar una o más de sus funciones básicas. También podría involucrar un daño mayúsculo a los recursos de la misma, una pérdida financiera masiva o un daño catastrófico a individuos involucrando pérdida de vidas o serias enfermedades con riesgo de muerte.

Las organizaciones que sufrirían mayor impacto ante una brecha en la seguridad de sus datos, son las que a su vez tienen mayores requerimientos de seguridad. Un ejemplo de requerimiento de seguridad bajo es el acceso confidencial a la información de directorio de una institución educativa estatal, datos de los estudiantes o listas departamentales. Esta información se encuentra típicamente disponible de manera prácticamente pública y se suele publicar en el sitio de la institución.

Un sitio web que consiste de un foro de discusión es un buen ejemplo de requerimientos de seguridad moderados. Si un hacker o un usuario registrado pudiera falsificar algunas de las entradas, esto produciría un daño moderado al administrador del sitio, generando pérdidas económicas y mala reputación.

Un ejemplo de requerimiento de seguridad alto es la información de alergias de pacientes de un hospital. Información imprecisa podría resultar en daños serios o en la muerte de un paciente y exponer al hospital a responsabilidad civil masiva. Si alguien que tenga acceso a manipular la información introdujera deliberadamente datos falsos para causar daño al hospital, la base debería ser restaurada rápidamente a un estado confiable, y debería ser posible rastrear el error para encontrar a la persona responsable.

Desafíos de la seguridad informática

La seguridad no es tan simple como le puede parecer a los novatos. Los requerimientos de seguridad parecen ser sencillos. De hecho, la mayoría de los requerimientos de servicios de seguridad pueden recibir etiquetas de una sola palabra que se explican por sí solas: confidencialidad, autenticación, no-repudio, integridad. Pero los mecanismos usados para alcanzar estos requerimientos pueden llegar a ser muy complejos, y entenderlos envuelve mucho razonamiento.

Para desarrollar un mecanismo de seguridad o algoritmo, uno debe siempre considerar posibles ataques a esas funciones. En muchos casos, ataques exitosos son diseñados viendo el problema de una forma completamente diferente, y por lo tanto explotando un punto débil inesperado.

Por ello, los procedimientos usados para proveer servicios de seguridad son frecuentemente contraintuitivos. Normalmente, un mecanismo de seguridad es complejo y desde la definición del requerimiento de seguridad particular, no resulta obvio que medidas elaboradas sean necesarias.

Además, estos mecanismos envuelven más de un algoritmo particular o protocolo. Y requieren que los participantes posean información secreta (claves por ejemplo), lo cual genera interrogantes en cuanto a la creación, distribución y protección de esa información secreta. Además, se depende de protocolos de comunicaciones que muchas veces pueden complicar la tarea de desarrollar un mecanismo de seguridad, debido por ejemplo a latencias variables que entran en conflicto con límites de tiempo intrínsecos a la seguridad. Hay una tendencia natural de parte de los usuarios y administradores de sistemas a percibir poco o nulo beneficio en la inversión en seguridad, hasta que ocurre un problema.

Muy a menudo la seguridad es algo que se piensa para después, para incorporarlo luego al sistema, y no se diseña desde el principio como una parte integrada del mismo.

Servicios de seguridad

El estándar X.800 define un servicio de seguridad como un servicio provisto por una capa de protocolo de los sistemas de comunicación que asegura una adecuada seguridad de los sistemas y los datos transferidos entre ellos. Hay cinco categorías de servicios de seguridad. Estos se pueden ver en la tabla 2.1.

Tabla 2.1: Servicios de seguridad

Autenticación	Asegura que la comunicación está siendo realizada por quien dice ser.
Control de acceso	Prevención del uso no autorizado de un recurso. Quién puede acceder, bajo qué condiciones y qué puede hacer con el recurso.
Confidencialidad	Protección de los datos de una lectura no autorizada.
Integridad	Aseguramiento de que los datos recibidos son exactamente los que se enviaron.
No repudio	Protección contra la negación de una de las entidades involucradas de haber participado en toda o parte de la comunicación.

Autenticación es un servicio relacionado a asegurar que la comunicación es auténtica. En el caso de un único mensaje, como una señal de alarma, la función del servicio de autenticación es asegurar al destinatario que el mensaje proviene del origen que asegura provenir. En el caso de una interacción prolongada, como la conexión de un cliente a un servidor, el servicio se asegura de que las dos entidades son auténticas, es decir, que cada entidad es quien dice ser. En segundo lugar, el servicio debe asegurar que la conexión no es interferida por un tercero que pueda simular ser uno de los participantes para transmitir o recibir mensajes.

Control de acceso, en el contexto de seguridad de redes, es la capacidad de limitar y controlar el acceso a los servidores y aplicaciones. Para lograr esto, cada entidad que intente acceder debe primero ser identificada, o autenticada, de forma que los derechos de acceso se otorguen a la misma.

Confidencialidad es la protección de los datos que son transmitidos contra ataques pasivos. Que ningún atacante pueda acceder al contenido de los datos de la transmisión. La otra arista de la confiden-

cialidad es la protección del flujo de tráfico del análisis de tráfico. Esto requiere que los atacantes no sean capaces de observar y analizar el origen y destino, frecuencia, longitud y otras características del tráfico.

Integridad es la certeza de que los datos recibidos son los mismos que se enviaron, y por tanto es la capacidad de detectar si fueron adulterados. Los mecanismos de seguridad lidian en este caso con ataques activos, por lo que interesa la detección más que la prevención.

No repudio evita que tanto el emisor como el receptor desconozcan un mensaje transmitido. Así, cuando un mensaje es enviado, el receptor puede probar que quien figura como emisor efectivamente lo envió. De la misma forma cuando un mensaje es recibido, el emisor puede probar que quien figura como receptor efectivamente lo recibió. [1]

Además de los cinco anteriormente mencionados, existe el servicio de disponibilidad. Este servicio de seguridad consiste en la propiedad de un sistema o recurso de un sistema de estar accesible y usable bajo demanda por una entidad autorizada, acorde a especificaciones de rendimiento. Una gran variedad de ataques pueden resultar en la pérdida o reducción de la disponibilidad. Algunos de estos ataques pueden ser contenidos o controlados por los sistemas de autenticación o encriptación, mientras que otros requieren algún tipo de interacción física para evitar o recuperar la pérdida de disponibilidad. Este servicio se encarga de los aspectos de seguridad relacionados a ataques de denegación de servicio.

2.2. Conceptos básicos de criptografía

Uno de los principales mecanismos de seguridad es la encriptación. La encriptación simétrica, o encriptación tradicional, también llamada encriptación de una sola clave. Era el único tipo de encriptación existente hasta que se desarrolló la encriptación de clave pública en la década de 1970. De los dos, sigue siendo por mucho el tipo de encriptación más usado. DES y AES son dos de los algoritmos más usados en cifrado simétrico.

Como conceptos principales de encriptación, debemos destacar los siguientes. **Texto plano** es el nombre que se le da al mensaje original, es decir, los datos inteligibles. Por otro lado, al mensaje codificado se le llama **texto cifrado**. El proceso de convertir texto plano en texto cifrado es conocido como **encriptación o cifrado**. El proceso de restaurar el texto plano a partir del texto cifrado es **desencriptación o descifrado**. El **algoritmo de cifrado** es el algoritmo que realiza varias sustituciones y transformaciones al texto plano. El **algoritmo de descifrado** es básicamente el algoritmo de cifrado ejecutado al revés. Tal esquema es conocido como **sistema criptográfico o sistema de cifrado**. Las técnicas utilizadas para descifrar un mensaje sin conocimiento de los detalles de cifrado caen en el área del **criptoanálisis**. Es lo que popularmente se conoce como “romper el código”. Lás áreas de criptografía y criptoanálisis juntas se llaman **criptología**. [15]

Los sistemas de criptografía se clasifican según el número de claves que utilizan. Si el emisor y el receptor utilizan la misma clave, el sistema se conoce como **simétrico**, de una sola clave, o de encriptación convencional. Si usan claves diferentes, el sistema se conoce como **asimétrico**, de dos claves o encriptación de clave pública.

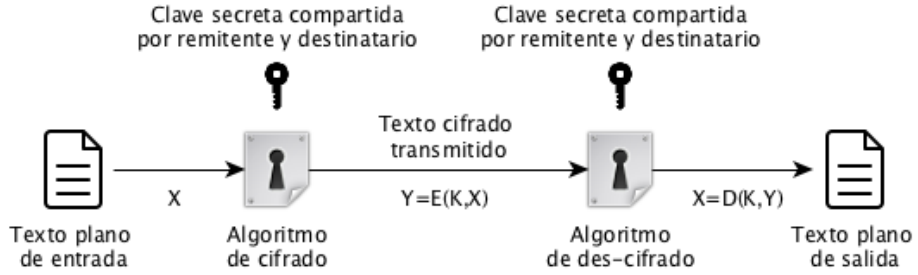

Figura 2.1: Criptografía simétrica

2.3. Criptoanálisis y ataque de fuerza bruta

Normalmente, el objetivo de atacar un sistema de encriptación es obtener la clave en uso en vez de simplemente recuperar el texto plano de un sólo texto cifrado. Hay dos enfoques generales para atacar un esquema de encriptación tradicional.

Los **ataques de criptoanálisis** dependen de la naturaleza del algoritmo y tal vez de cierto conocimiento de las características generales del texto plano o incluso algunos ejemplos de texto plano y texto cifrado. Los ataques de **fuerza bruta** prueban todas las claves posibles en un texto cifrado hasta que se obtiene una traducción inteligible a texto plano. En promedio, es necesario probar la mitad de las claves posibles para tener éxito.

Cualquiera de los dos ataques tiene un efecto catastrófico en caso de tener éxito. Todos los mensajes pasados y futuros encriptados con esa clave están comprometidos. [15]

2.4. Criptografía de clave pública

La encriptación asimétrica es una forma de sistema de criptografía en la cual la encriptación y la desencriptación se realizan usando diferentes claves, una pública y una privada. También se la conoces como encriptación de clave pública. Transforma el texto plano en texto cifrado usando una de las dos claves y un algoritmo de encriptación. Usando la otra clave y un algoritmo de desencriptación, se puede recuperar el texto plano del texto cifrado.

La criptografía de clave pública puede ser usada para obtener confidencialidad, autenticación, o ambas. El sistema criptográfico de clave pública más utilizado es RSA. La dificultad de atacar RSA se basa en la dificultad de obtener los factores primos de un número compuesto.

El desarrollo de la criptografía de clave pública es la mayor y tal vez la única verdadera revolución en toda la historia de la criptografía. Desde sus comienzos hasta los tiempos modernos, prácticamente todos los sistemas criptográficos se han basado en herramientas básicas de sustitución y permutación. Luego de milenios de trabajar con algoritmos que podían ser calculados a mano, el desarrollo de una máquina de encriptación y desencriptación fue un avance mayúsculo. Con la llegada de las computadoras, se llegó incluso más lejos y apareció DES. Pero incluso DES se seguía basando en simples herramientas de sustitución y permutación.

La criptografía de clave pública está basada en funciones matemáticas en lugar de sustitución y permutación. Aún más importante, la misma es asimétrica, involucrando el uso de dos claves separadas,

en contraste con la encriptación simétrica que usa una sola clave. El uso de dos claves ha profundizado los alcances en las áreas de confidencialidad, distribución de claves y autenticación. [15]

Un clásico concepto erróneo es que la criptografía de clave pública es menos propensa al criptoanálisis que la criptografía simétrica. De hecho, la seguridad de cualquier esquema de encriptación depende de la longitud de la clave y del trabajo computacional necesario para romper el cifrado. En principio, no hay nada en la criptografía simétrica y asimétrica que haga a una más fuerte que la otra ante ataques del tipo del criptoanálisis.

Otro clásico concepto erróneo común es que la encriptación de clave pública es una técnica de propósito general que ha transformado la encriptación simétrica en algo obsoleto. Por el contrario, a causa de la sobrecarga computacional que implica la criptografía de clave pública, no parece probable que la criptografía simétrica sea abandonada. La misma continuará siendo utilizada en varios escenarios.

Por último, se tiende a pensar que la distribución de claves es trivial usando criptografía de clave pública, comparada con el intercambio de una única clave secreta que ambos extremos deben conocer para utilizar criptografía simétrica. De hecho, se necesitan protocolos para la criptografía asimétrica, que involucran toda una infraestructura, un agente central, y procedimientos que no son en nada más simples o más eficientes que los requeridos para la criptografía simétrica.

Los conceptos más importantes de la criptografía asimétrica son las claves asimétricas, los certificados de clave pública, los algoritmos criptográficos de clave pública y la infraestructura de clave pública.

Las **claves asimétricas** son dos claves relacionadas, una pública y una privada, que son usadas para realizar operaciones complementarias, como la encriptación y desencriptación o la generación de firma y la verificación de la misma.

Los **certificados de clave pública** son documentos digitales emitidos por una Autoridad de Certificación y firmados de forma digital con la clave privada de dicha autoridad. Enlazan el nombre de quien suscribe a una clave pública, es decir, cada certificado indica que quien se suscribe identificado en el certificado posee una clave privada absolutamente secreta.

Los **algoritmos de clave pública**, o algoritmos criptográficos asimétricos, son algoritmos que usan dos claves relacionadas, una pública y una privada. Estas claves cumplen con la propiedad de que es imposible de manera computacional deducir la clave privada a partir de la clave pública.

Una **Infraestructura de Clave Pública** o **PKI** por sus siglas en idioma inglés, son un conjunto de políticas, procesos, plataformas de servidores, software y estaciones de trabajo utilizadas con el propósito de administrar certificados y pares de claves públicas y privadas, incluyendo la capacidad de emitir, mantener y revocar certificados de clave pública.

Sistemas criptográficos de clave pública

Los algoritmos asimétricos usan una clave para encriptar y otra clave diferente pero relacionada para desencriptar. Estos algoritmos tienen una característica importante: Es computacionalmente imposible determinar una de las dos claves, incluso conociendo la otra clave y el algoritmo.

Además, algunos algoritmos, como RSA, tienen una característica adicional: Cualquiera de las dos claves puede ser usada para encriptar, y la otra es usada para desencriptar.

Un esquema de encriptación de clave pública tiene seis ingredientes. **Plaintext o texto plano**: es el mensaje legible, los datos que se usarán como entrada del algoritmo, análogo al caso de la criptografía simétrica. **Algoritmo de encriptación**: realiza varias transformaciones en el texto plano. **Claves pública y privada**: es el par de claves seleccionadas de forma tal que una es usada para encriptar, y la otra para desencriptar. **Ciphertext o texto cifrado**: es el mensaje devuelto producto del algoritmo. Depende del texto plano y la clave usada para encriptar. Dado un mensaje, dos claves diferentes producen dos textos cifrados diferentes. **Algoritmo de desencriptación**: este algoritmo toma el texto cifrado y la clave complementaria a la clave usada para la encriptación y como resultado provee el texto plano original.

Con este enfoque, todos los participantes tienen acceso a las claves públicas. Las claves privadas se generan localmente y se mantienen sin distribuir. Siempre y cuando la clave privada de un usuario permanezca protegida y en secreto, podrá recibir mensajes confidenciales de forma segura. En cualquier momento, un sistema puede cambiar su clave privada y publicar la nueva clave pública para reemplazar la anterior clave que todos lo demás actores conocían. [15]

Las firmas digitales utilizan también criptografía de clave pública, aunque al contrario del enfoque explicado anteriormente, la clave privada de un usuario se utiliza para generar la firma y la clave pública es usada por los destinatarios para corroborar que el mensaje provenga del usuario que declara haberlo enviado. La firma digital no persigue la confidencialidad, por lo cual la generación de la firma consiste en generar un resumen del mensaje original, de un tamaño fijo, y encriptar ese resumen. Este resumen se conoce como Hash.

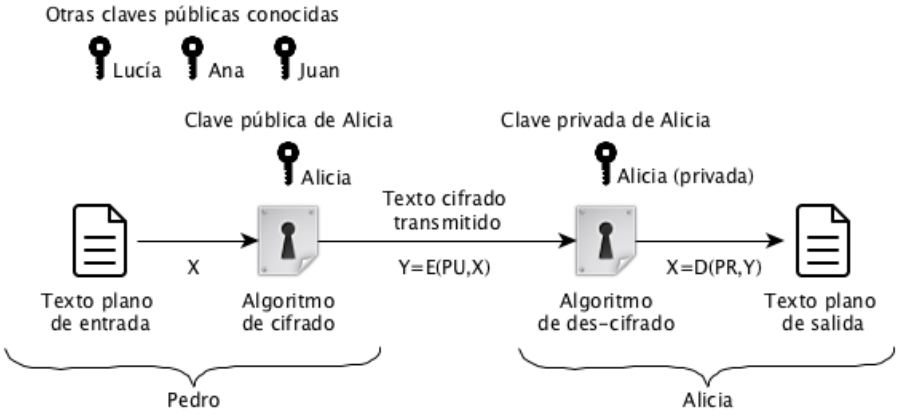

Figura 2.2: Encriptación con clave pública

2.5. Firmas Digitales

El desarrollo más importante en el trabajo sobre criptografía de clave pública es la firma digital. La firma digital provee un conjunto de capacidades que sería difícil implementar de otra manera.

Una firma digital es un mecanismo de autenticación que permite al creador del mensaje adjuntar un código que actúa como firma.

La firma es formada tomando el hash del mensaje original y encriptándolo con la clave privada del emisor. La firma garantiza el origen y la integridad del mensaje.

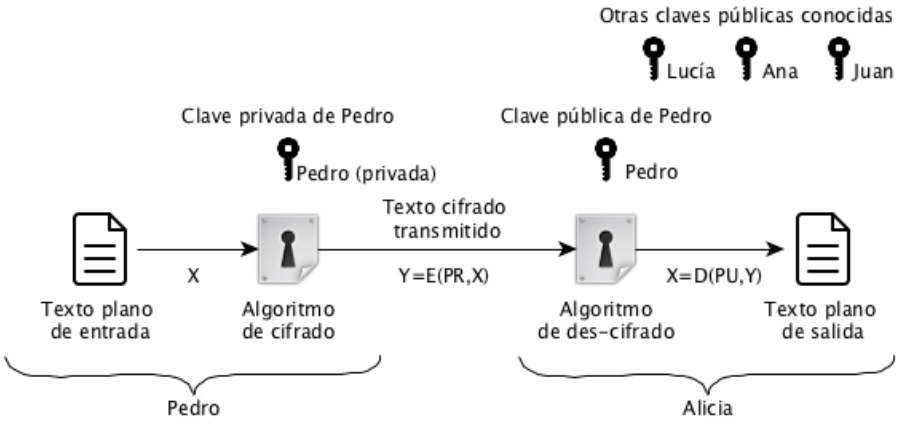

Figura 2.3: Encriptación con clave privada

La autenticación de mensaje protege a dos partes que intercambien mensajes de terceros. Pero no los protege al uno del otro. Muchas formas de disputa pueden darse entre dos actores que se comunican.

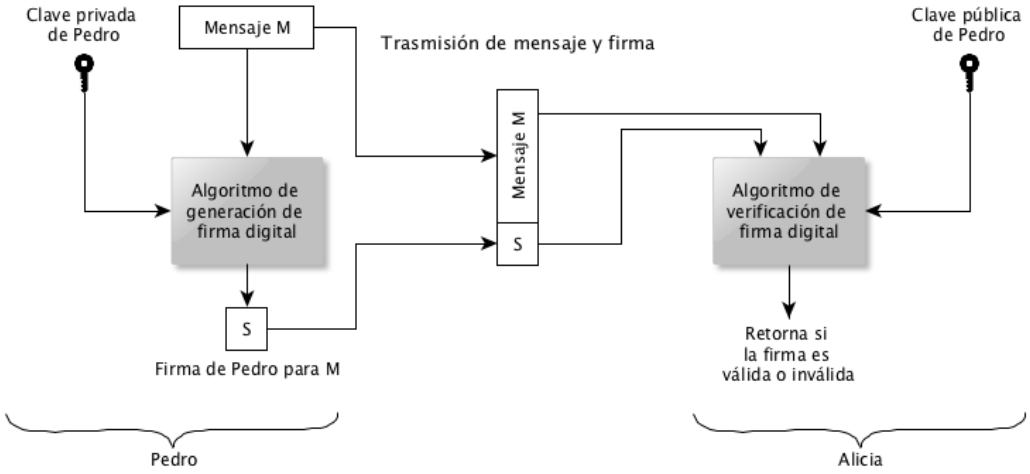

Figura 2.4: Modelo genérico del proceso de Firma Digital

Por ejemplo, supongamos que Juan envía un mensaje autenticado a María, usando un esquema con criptografía simétrica en lugar de firma digital. Consideremos las siguientes disputas que podrían darse.

María puede falsificar un mensaje diferente y declarar que provino de Juan. María simplemente tendría que crear un mensaje y adjuntar un código de autenticación creado usando la clave secreta que ambos comparten.

Juan puede negar haber enviado un mensaje. Dado que es posible que María falsifique un mensaje, no es posible probar que Juan realmente haya enviado algún mensaje.

La preocupación ante cualquier de estos posibles escenarios es totalmente legítima. Un ejemplo más concreto del primer escenario es el siguiente: una transferencia electrónica de fondos tiene lugar, y el receptor aumenta el monto transferido y declara haber recibido ese monto mayor de la otra persona. Un ejemplo más concreto del segundo escenario es un mensaje de correo electrónico que contiene instrucciones para un corredor de valores de la bolsa que luego termina teniendo consecuencias negativa. El remitente del correo electrónico podría fingir que nunca envió dicho mensaje.

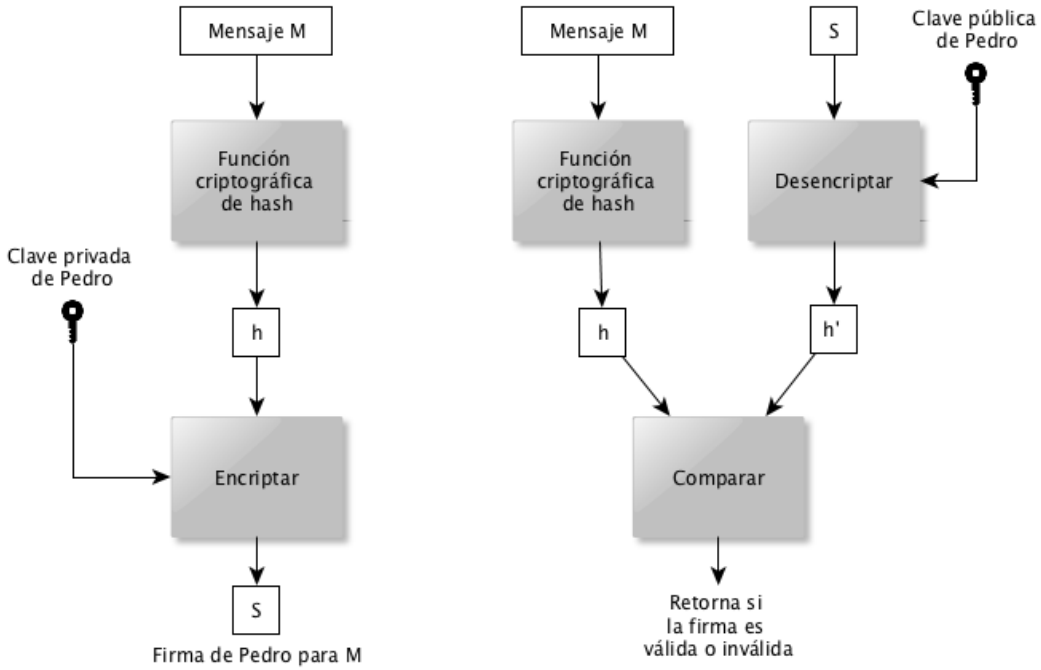

Figura 2.5: Representación simplificada de los elementos esenciales del proceso de firma digital

En situaciones en las que no hay confianza total entre el emisor y el receptor, se necesita algo más que autenticación de mensaje. La solución más atractiva a este problema es la firma digital. La firma digital debe tener las siguientes propiedades:

- Debe verificar el autor y la fecha y hora de la firma.
- Debe autenticar los contenidos del mensaje al momento de la firma.
- Debe ser verificable por terceros, para resolver disputas.

Por lo tanto, la firma digital incluye la función de autenticación de mensaje. Podríamos formular los siguientes requerimiento para una firma digital.

- Debe ser un patrón de bits que depende del mensaje que está siendo firmado.
- Debe usar alguna información que sea única para el emisor para prevenir tanto falsificación como repudio.
- Debe ser relativamente fácil producir la firma.
- Debe ser relativamente fácil reconocer y verificar la firma.
- Debe ser computacionalmente imposible falsificar la firma, ya sea construyendo un nuevo mensaje para una firma existente o construyendo una firma fraudulenta para un mensaje existente.
- Debe ser práctico guardar una copia de la firma digital en el almacenamiento de un sistema.

Firma digital directa es un esquema de firma digital que involucra solo la comunicación entre un origen y un destino. Se asume que el destinatario conoce la clave pública del emisor. [15]

La validez de estos esquemas depende de la seguridad de la clave pública del emisor. Si éste desea negar haber enviado un determinado mensaje, puede aseverar haber perdido la clave privada o haber sufrido el robo de la misma y alegar que un tercero utilizó su clave para falsificar su firma. Controles

administrativos relacionados a la seguridad de las claves privadas pueden ser utilizados para frustrar o al menos debilitar esta táctica, pero la amenaza sigue latente en cierto grado. Un ejemplo es solicitar que cada mensaje firmado incluya la fecha y hora en la que fue enviado y que todos los emisores reporten de inmediato a una autoridad central de administración de claves privadas si la seguridad de su clave está comprometida. Otra amenaza es que una clave privada puede efectivamente ser robada a X en un momento T. El oponente puede entonces enviar un mensaje firmado con la firma de X y estampado con la fecha y hora posterior a T. La técnica para lidiar con esta amenaza es similar a la anterior. Está universalmente aceptado el uso de certificados digitales y autoridades de certificación centralizadas que actúan como terceros en las disputas.

Es debido a este tipo de amenazas que en los entornos de firma digital y criptografía de clave pública es fundamental la administración y distribución de claves.

2.6. Administración y distribución de claves

La distribución de claves es la función que entrega una clave a dos entidades que desean intercambiar datos encriptados de manera segura. Un mecanismo o protocolo es necesario para proveer la seguridad de las claves distribuidas. Esto involucra frecuentemente el uso de claves maestras, que se usan con una frecuencia mínima y tienen una duración prolongada, y claves de sesión, que son generadas y distribuidas para el uso temporal entre dos partes.

Los esquemas de criptografía de clave pública sólo son seguros si se garantiza la autenticidad de la clave pública. X.509 define el formato para certificados de clave pública. Este formato es ampliamente usado en una gran variedad de aplicaciones.

Una **infraestructura de clave pública (PKI)** se define como un conjunto de hardware, software, personas, políticas y procedimientos necesarios para crear, administrar, almacenar, distribuir, y revocar certificados digitales basados en criptografía asimétrica.

El escenario más simple es el intercambio de claves entre dos partes, usando criptografía simétrica, es decir compartiendo una clave privada para transmitir las claves públicas. A y B pueden:

1. Intercambiar una clave privada de manera presencial
2. Un tercero puede seleccionar una clave y entregarla físicamente a cada uno
3. Si A y B ya usaron otras claves antes, pueden intercambiar una nueva encriptándola usando las claves antiguas
4. Si A y B tienen ambos una conexión encriptada con un tercero C, C puede enviarle a cada uno las claves que necesitan de forma segura.

Como vemos los protocolos para la distribución de claves no son menores, y lejos están de ser sencillos. A simple vista podemos identificar varias dificultades con estos esquemas. La opción número 4 parece ser la más apropiada, y es la que ha sido más ampliamente adoptada en el mundo real. En este esquema, un centro de distribución es responsable de distribuir las claves entre los pares.

La distribución basada en un **organismo central** está basada en la jerarquía de claves. Como mínimo, consideramos dos niveles de claves: las **claves de sesión**, claves temporales que se utilizan para intercambiar datos entre los sistemas finales, y **claves maestras** que son utilizadas para la comunicación

entre el organismo central de distribución y cada sistema o usuario final, es decir para la distribución de las claves de sesión. [15]

No es necesario limitar la función de distribución a un único Organismo Central de Distribución de claves, puede existir una jerarquía de estos. Un esquema jerárquico minimiza el esfuerzo involucrado en la distribución de claves maestras, y limita el daño que podría producir el descubrimiento de una clave maestra o un Organismo Central con fallas o fraudulento.

Las claves de sesión en este tipo de esquemas típicamente duran lo que dura la conexión y el intercambio de mensajes entre dos partes. Para cada nueva conexión o nueva conversación, se necesita generar una nueva clave de sesión. Para protocolos sin conexión, no hay un inicio o fin explícito de la comunicación, por lo cual no es obvio cuán frecuentemente es necesario cambiar las claves de sesión. Cuanto menos tiempo transcurra y menos mensajes se intercambien con una sola clave de sesión, se minimiza el impacto de un posible ataque.

Distribución de claves públicas

En el contexto de la criptografía asimétrica y la firma digital, los protocolos de distribución de claves tienen mayor complejidad y tienden a basarse en estándares y entidades de administración de claves reconocidas. La clave para las más avanzadas **Infraestructuras de clave pública** son los **certificados**.

Dado que es inviable consultar una entidad de certificación confiable cada vez que se necesite constatar la identidad de un mensaje, los certificados proveen un mecanismo para que los participantes intercambien sus claves de manera segura, como si las obtuvieran de la entidad, pero sin comunicarse con ella. En esencia, un certificado consiste en una clave pública, un identificador del dueño, y todo ese bloque está firmado por ese tercero de confianza.

Cada usuario puede presentar de manera segura su clave pública a una **entidad de certificación**, o **CA** por sus siglas en inglés, y obtener un certificado. Estas entidades son organizaciones en las que los usuarios confían. Luego, un usuario puede publicar su certificado. Cualquiera que necesite la clave pública del mismo puede obtener su certificado y constatar su validez por la firma de la entidad de certificación. [15]

Podemos enunciar los siguientes requerimientos para el esquema:

1. Cualquier participante puede leer un certificado para determinar el nombre y la clave pública del dueño del mismo.
2. Cualquier participante puede verificar que el certificado fue otorgado por la entidad de certificación y no es una falsificación.
3. Sólo la entidad de certificación puede crear y actualizar certificados.
4. Cualquier participante puede verificar la vigencia del certificado.

La aplicación para la generación de un formulario debe ser en persona o por algún medio de comunicación autenticado y seguro. Para un participante A, la autoridad provee un certificado de la forma:

$C_A = Encrypt(PR_{CA}, [T||ID_A||PU_A])$

donde PRCA es la clave privada usada por la autoridad y T es un timestamp o estampa de tiempo. Luego A envía el certificado a otro participante, quien lee y verifica el certificado de la siguiente manera:

$Decrypt(PU_{CA}, C_A) = (T||ID_A||PU_A)$

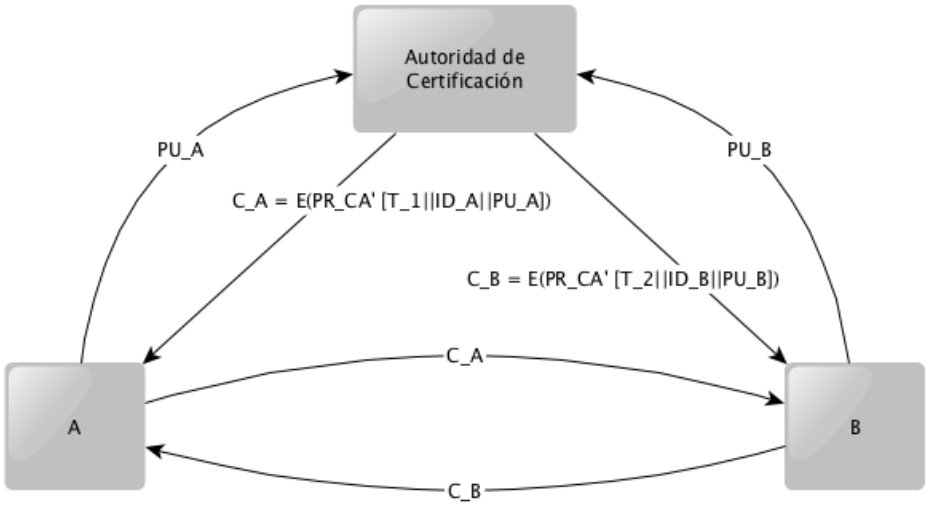

Figura 2.6: Distribución de claves públicas basada en una autoridad de certificación

El receptor usa la clave pública de la autoridad de certificación, PU_{CA}, para desencriptar el certificado. Dado que el certificado es legible solo usando la clave pública de la autoridad, se verifica que el certificado proviene de la misma. Los elementos ID_A y PU_A proveen al receptor de la clave pública y el nombre del emisor.

La estampa de tiempo corrobora la validez del certificado, contrarresta el siguiente escenario. La clave privada de A es obtenida por un adversario. A genera un nuevo par de claves privada/pública y aplica ante la autoridad de certificación para un nuevo certificado. Mientras tanto, el adversario envía el viejo certificado a B. Luego B envía mensajes encriptados usando la clave pública comprometida, y el adversario puede leer esos mensajes. En este contexto, el compromiso de la clave privada es comparable con la pérdida de la tarjeta de crédito. El dueño puede cancelar el número de tarjeta de crédito pero está en riesgo hasta que todos los posibles actores estén al tanto de que la tarjeta está obsoleta. Por ello, la estampa de tiempo sirve como una fecha de vencimiento. Si un certificado es suficientemente antiguo, se asume expirado. El estándar X.509 se ha vuelto universalmente aceptado para dar formato a certificados de clave pública. El mismo es usado en la mayoría de las aplicaciones de seguridad en redes, incluyendo seguridad IP y Seguridad de la Capa de Transporte, o TLS por sus siglas en inglés.

X509 Standard

X.509 es parte de X.500, una serie de recomendaciones que define el servicio de un directorio. Un directorio es, un servidor o un conjunto distribuido de servidores que mantiene una base de datos con información de los usuarios.

X.509 define un framework para proveer servicios de autenticación de un directorio X.500 a sus usuarios. Los directorios mantienen los certificados de los usuarios. Cada certificado contiene la clave pública de un usuario y está firmado con la clave privada de una entidad de certificación de confianza. Adicionalmente, X.509 define protocolos de autenticación basados en certificados de clave pública.

X.509 es un estándar importante porque la estructura de certificados y los protocolos de autenticación en él definidos son usados en una gran variedad de contextos, que incluyen S/MIME, Seguridad IP y SSL/TLS.

X.509 fue emitido inicialmente en 1988. El estándar fue revisado subsecuentemente para atacar algunos de los problemas de seguridad que se encontraron con el tiempo. Una recomendación revisada fue lanzada en 1993, la versión 2. Una tercera versión fue emitida en 1995 y revisada en los años 2000 y 2008.

X.509 está basado en el uso de criptografía de clave pública y firmas digitales. El estándar no dicta el uso de un algoritmo específico pero recomienda RSA. Se asume que el esquema de firma digital requiere el uso de una función de hash. Nuevamente, el estándar no dicta un algoritmo de hash específico.

Certificados

El corazón del esquema X.509 es el esquema de certificados de clave pública asociados a cada usuario. Se asume que estos certificados fueron creados por alguna entidad de certificación de confianza y colocados en el directorio por la entidad o por el usuario. El directorio en sí mismo no es responsable por la creación de las claves públicas o por la función de certificar, simplemente provee una ubicación fácilmente accesible para que los usuarios obtengan certificados.

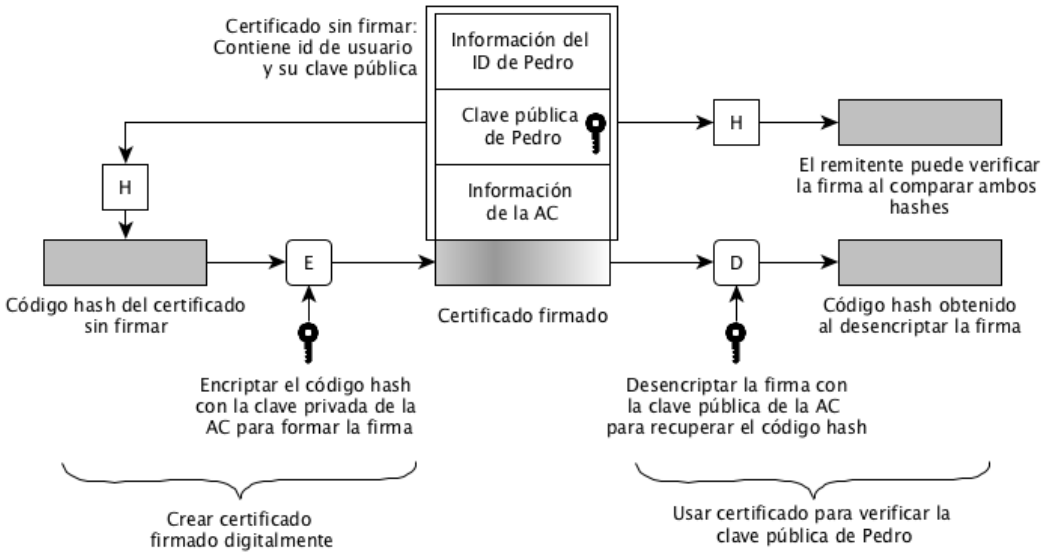

Figura 2.7: Uso de un Certificado de Clave Pública

La figura 2.8 muestra el formato de los certificados X.509.

La CA firma el certificado con su clave privada. Si un usuario conoce la clave pública de la entidad, puede verificar que el certificado fue firmado por la misma y que es válido.

Obteniendo un Certificado de Usuario

Los certificados de usuarios generados por una CA tienen las siguientes características:

- Cualquier usuario con acceso a la clave pública de la CA puede verificar la clave pública del usuario que fue certificado.
- Ninguna otra entidad que no sea la CA puede modificar el certificado sin que sea detectado. Dado que no se pueden falsificar, son colocados en un directorio sin la necesidad de realizar esfuerzos especiales para proteger dicho directorio.

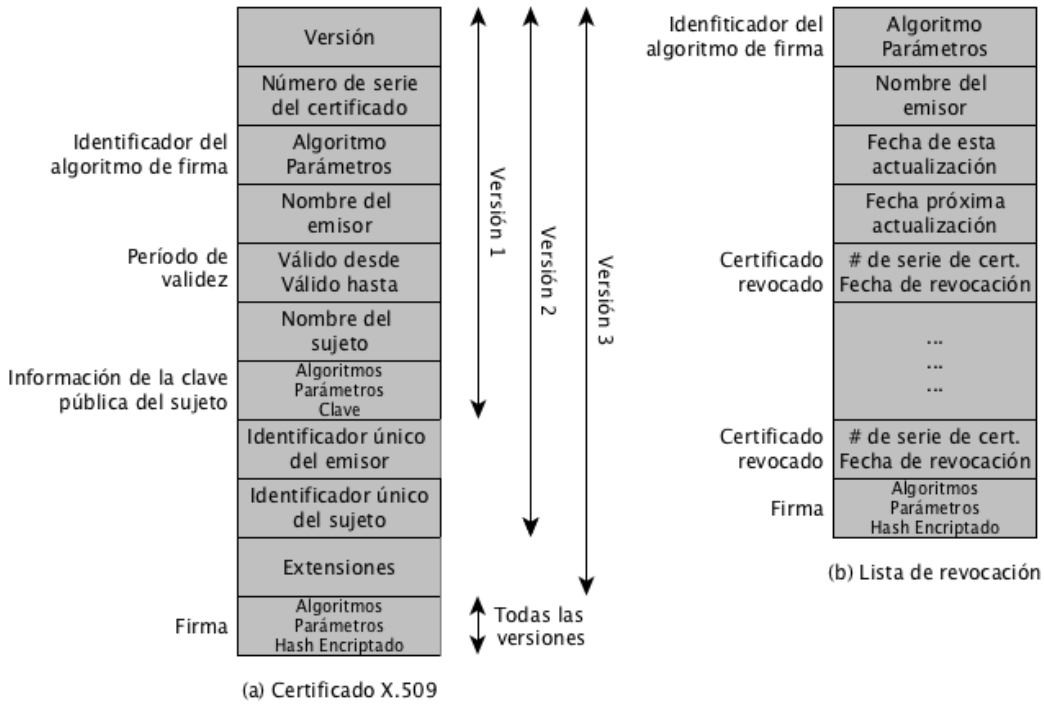

Figura 2.8: Formatos X.509

No todos los usuarios suscriben a la misma CA. Es necesaria una distribución, por lo cual hay una gran cantidad de ellas. Es por esto que existen las denominadas cadenas de certificados. El esquema propone que un certificado puede estar firmado por una CA_3, cuyo certificado a su vez está firmado por una CA_2, cuyo certificado está firmado por la CA_1.

No todos los sistemas tienen confianza en todas las entidades. Pero las más importantes a nivel mundial son conocidas por todos, y se dispone de los certificados y de su clave pública. Por lo tanto, independientemente de que dos usuarios intercambien sus certificados para comunicarse, y los mismos hayan sido emitidos por CAs de niveles inferiores de la jerarquía, estos usuarios podrán confiar en esa comunicación a través del uso de la cadena.

El usuario A utilizará la clave pública de la CA_1 en la que confía para leer el certificado de la CA_2 de menor nivel que firmó el certificado del usuario B. Una vez obtenida la clave pública de CA_2 y sabiendo que es confiable, puede leer y verificar el certificado del usuario B.

Revocación de certificados

Cada certificado incluye su período de validez, como una tarjeta de crédito. Normalmente, un certificado es emitido poco antes de la fecha de vencimiento del anterior. Además, es necesario disponer de un mecanismo de revocar un certificado antes de que venza, por cualquier motivo que comprometa la seguridad. Por ejemplo:

- Si se sospecha que la clave privada del usuario podría haber sido obtenida por un atacante.
- Si el usuario ya no puede mantener el certificado emitido porque cambian las políticas de la CA o porque el mismo cambia su nombre.
- Se cree que la seguridad de la CA puede estar comprometida.

Cada CA debe mantener una lista de todos los certificados no expirados que hayan sido revocados. Estas listas son publicadas en los directorios igual que los certificados. La razón por la cual no es necesario mantener una lista de los certificados revocados cuando ya vencieron es que la fecha de vencimiento implica que ya no son válidos por sí sola.

Cada Lista de Revocación de Certificados, o CRL por sus siglas en inglés, consiste del nombre de la entidad que emite, es decir la CA, la fecha en que fue publicada, y la lista de revocaciones. Cada revocación está compuesta de un número de serie de un certificado y la fecha de revocación del mismo. Dado que los números de serie son únicos para cada CA, no es necesario incluir más información.

Cuando un usuario final recibe un certificado en un mensaje, debe determinar si el certificado fue revocado antes de confiar en él. Podría consultar esta información en el directorio cada vez que lo necesita, pero en pos de evitar retrasos y posibles costos asociados con la búsqueda en directorios, es común que los sistemas y usuarios mantengan una copia local de los certificados revocados, y la actualicen frecuentemente.

Esta sección fue elaborada utilizando como referencia [15], [2] y [7].

2.7. Seguridad a nivel capa de transporte

Los usuarios de sistemas web no están entrenados en cuestiones de seguridad, ni tienen por qué estarlo. Es por ello que si bien los navegadores web son simples de utilizar, los servidores web sencillos de configurar y las aplicaciones web relativamente fáciles de desarrollar, la web puede enfrentar al usuario con amenazas de seguridad día a día.

Un grupo de estas amenazas son ataques pasivos, como el espionaje de información del tráfico de red, mientras que otro grupo son ataques activos, como el fraude, alteración de mensajes en tránsito entre el cliente y el servidor o alteración de la información de un sitio web.

Otra posible clasificación para las amenazas de seguridad es por la ubicación de las mismas. Hay amenazas en los clientes, en los servidores y en el tráfico de red. Dado que el tráfico de la mayoría de los sistemas web se da a través del protocolo de la capa de transporte **TCP** "**Transmission Control Protocol**", implementar seguridad a este nivel es una buena alternativa para enfrentar estas amenazas.

SSL o Secure Sockets Layer y su sucesor TLS o Transport Layer Security, son protocolos criptográficos de seguridad que se utilizan a nivel capa de transporte y capa de aplicación de la web, por encima de TCP. La gran mayoría de los navegadores web implementan estos protocolos.[14]

Hay dos conceptos claves que componen SSL/TLS. Una **conexión** es un transporte que provee una relación par-a-par y es transitoria. Cada conexión es asociada a una sesión. Una **sesión** es una asociación entre un cliente y un servidor. Las sesiones son creadas por el Handshake Protocol. Las sesiones definen un conjunto de parámetros de seguridad que serán compartidos a lo largo de múltiples conexiones. Las sesiones evitan la costosa negociación de nuevos parámetros de seguridad en cada conexión. [15]

SSL Record Protocol provee dos servicios a las conexiones SSL: confidencialidad e integridad de mensajes. La confidencialidad se logra por una clave secreta que es intercambiada durante el Handshake Protocol, usada para encriptar mensajes. La integridad se logra a través de otra clave secreta definida durante el protocolo Handshake, usada para crear MACs, es decir Códigos de Autenticación de Mensajes.

Tabla 2.2: Comparación de amenazas en la Web

	Amenazas	Consecuencias	Prevención
Integridad	▪ Modificación de datos del usuario ▪ Troyano en el navegador ▪ Modificación de memoria ▪ Modificación de mensaje en tránsito	▪ Pérdida de información ▪ Compromiso de la máquina ▪ Vulnerabilidad a otros ataques	▪ Sumas de comprobación criptográficas (hash)
Confidencialidad	▪ Espionaje de la red ▪ Robo de información del servidor ▪ Robo de información del cliente	▪ Pérdida de información ▪ Pérdida de privacidad	▪ Encriptación
Denegación de servicio	▪ Detener ejecuciones de programas ▪ Inundar un servidor con requests genéricos ▪ Llenar el disco o la memoria ▪ Aislar una máquina con ataques DNS	▪ Destructivo ▪ Molesto ▪ Impide a los usuarios realizar su trabajo	▪ Difícil de prevenir ▪ Cuotas de utilización de ▪ servicios por intervalos de tiempo
Autenticación	▪ Suplantación de identidad de usuarios legítimos ▪ Falsificación de datos	▪ Falsa autenticación ▪ Considerar como verdadera información falsa	▪ Firma digital ▪ Otras técnicas criptográficas

Se dividen los datos a transmitir en fragmentos, cada fragmento es comprimido, se le agrega un MAC al final, y todo el conjunto se encripta. Al resultado se le agrega un encabezado SSL.

Change Cipher Spec Protocol es uno de los tres protocolos específicos de SSL que utilizan el SSL Record Protocol. Consiste en un sólo mensaje de un único byte con valor 1. El propósito de este mensaje es notificar que los próximos mensajes serán protegidos bajo las condiciones recién acordadas. Este mensaje se envía luego de finalizar el SSL Handshake Protocol.

Alert Protocol se utiliza para intercambiar mensajes de alertas entre los pares que se comunican sobre una sesión. Cada mensaje consiste de dos bytes. El primer byte toma el valor 1 **warning** o 2 **fatal** para convenir la severidad del mensaje. Si el nivel es fatal, inmediatamente se termina la conexión, y si bien otras conexiones de la sesión pueden continuar, no se crearán nuevas conexiones en esta sesión. El segundo byte contiene un código que indica la alerta específica. Por ejemplo unexpected_message indica que se recibió un mensaje inapropiado, y bad_certificate indica que el certificado recibido está corrupto.

Handshake Protocol permite al servidor y al cliente autenticarse uno con el otro y negociar algoritmos de encriptación y generación de MACs y las claves criptográficas que se usarán para proteger los datos en cada transmisión SSL. Este protocolo es utilizado antes que cualquier dato de aplicación sea transmitido.

HTTPS

HTTPS se refiere a la combinación de HTTP y SSL/TLS para implementar una comunicación segura entre el cliente web y el servidor web. La capacidad de manejar HTTPS está incorporada en todos los navegadores modernos. Su uso depende de que el servidor soporte la comunicación HTTPS.[14]

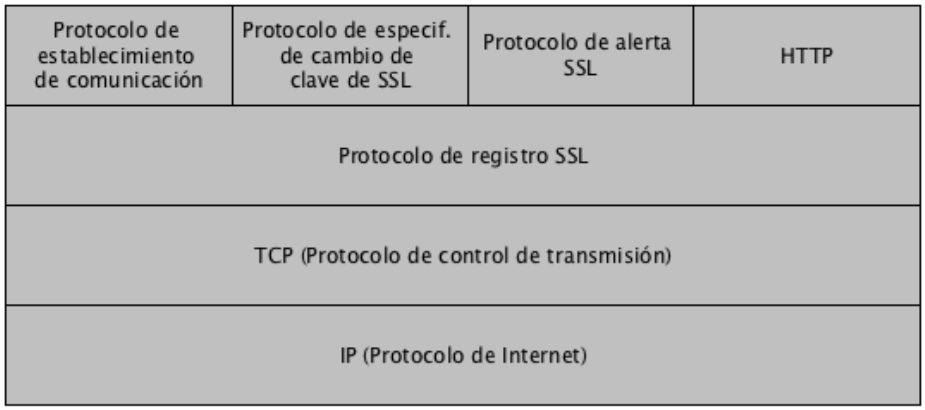

Figura 2.9: Pila de protocolos SSL/TLS

La principal diferencia que puede percibir el usuario del navegador web cliente es que la dirección del URL (uniform resource locator) comienza con https:// en lugar de http://. Una conexión HTTP normal usa el puerto 80, mientras que si se especifica HTTPS, se usará el puerto 443, que invoca TLS. [15]

Cuando se usa HTTPS, los siguientes elementos de comunicación son encriptados.

- El URL del recurso solicitado
- Contenidos del recurso
- Contenidos de los formularios web rellenados por el usuario
- Cookies enviadas por el navegador al servidor y del servidor al navegador
- Contenidos de los encabezados HTTP (HTTP headers)

En HTTPS, la entidad que actúa como cliente HTTP actúa también como cliente TLS. El cliente inicia la conexión al servidor en el puerto apropiado y envía el mensaje para iniciar el TLS Handshake Protocol. Cuando finaliza el handshake TLS, el cliente puede enviar el primer request HTTP. Los datos HTTP se envían como datos de aplicación TLS.

Existen tres niveles de protocolos en este tipo de interacción. HTTP en el nivel superior, que envía un pedido de conexión a la capa media. La capa media en este caso es TLS/SSL, donde se establece la sesión TLS entre el cliente y servidor, y posiblemente múltiples conexiones. Las conexiones se establecen sobre TCP, en el nivel inferior del esquema.

Con **HTTPS**, tenemos la garantía de varios **servicios de seguridad**. La conexión TLS es privada porque se utiliza criptografía simétrica para encriptar los datos, esto asegura la **confidencialidad**. La identidad de las partes es autenticada usando criptografía simétrica con TLS, aunque en aplicaciones web normalmente sólo se requiere confiar en la identidad del servidor. Esto provee el servicio de **autenticación**, aunque sólo sea el cliente el que puede confiar en el servidor y no a la inversa. Por último, dado que previo a ser encriptado cada bloque de datos es concatenado con un MAC, la conexión asegura la **integridad** de los mensajes.

La autenticación del lado cliente en HTTPS no está difundida y no es común en las aplicaciones web. Se puede recurrir a otros mecanismos de autenticación del cliente, como la firma digital.

2.8. Conclusiones

Los distintos servicios de seguridad y su nivel de impacto en el funcionamiento de una determinada organización son la base principal del análisis a realizar sobre cualquier conjunto de procesos de negocio. Los mecanismos mencionados en este capítulo nos permiten comprender mejor qué herramientas tenemos disponibles a la hora de mejorar la calidad de estos servicios. Su implementación mejora la calidad de la seguridad, y el costo radica en la mayor complejidad del sistema y la menor facilidad de uso para los usuarios. La forma en que la criptografía y los sistemas de BPM se integren debe mantener una relación conveniente entre la facilidad de uso y el nivel de seguridad deseado.

Capítulo 3

Marco teórico y conceptual de Business Process Management

Business Process Management ha recibido considerable atención recientemente de parte tanto de la comunidad de administración de negocios como de la de ciencias de la computación. Los miembros de estas comunidades se caracterizan por tener antecedentes educativos e intereses muy distintos. Incrementar la satisfacción del cliente, reducir costos del negocio, y establecer nuevos productos y servicios a bajo costo son aspectos importantes del Manejo de Procesos desde el punto de vista de la administración de negocios. La comunidad del software está interesada en proveer sistemas de software robustos y escalables. Dado que los procesos de negocios son realizados en complejos escenarios de tecnologías de la información, la integración de los sistemas de información existentes es una base importante para la realización técnica de los procesos de negocios.

Business process management se basa en la observación de que cada producto que la compañía provee al mercado es el resultado de un número de actividades realizadas. Los procesos de negocio son el instrumento clave para organizar estas actividades y para mejorar la comprensión de sus interrelaciones.

Las tecnologías de la información en general y los **sistemas de información** en particular merecen un rol importante en Business process management, porque cada vez más actividades llevadas a cabo por una compañía son soportadas por sistemas de información. Es decir que las actividades de BPM pueden ser realizadas por los empleados de la compañía manualmente o con la ayuda de sistemas de información. También hay actividades que pueden ser realizadas de manera totalmente automática por los sistemas de información, sin participación humana alguna. Una compañía puede alcanzar sus **objetivos de negocio** de manera eficiente y efectiva sólo si la gente y otros recursos empresariales, como los sistemas de información, interactúan de la manera apropiada. Los procesos de negocio son un concepto importante para facilitar esta colaboración efectiva.

3.1. Introducción a procesos de negocios y ciclos de vida

Un proceso de negocio consiste en un conjunto de actividades que son realizadas en coordinación en un ambiente organizacional y técnico. Estas actividades en conjunto cumplen un objetivo de negocio. Cada proceso de negocio pertenece a una sola organización, pero puede interactuar con otros procesos ejecutados por otras organizaciones. [16]

La base de **business process management** es la representación explícita de los procesos de negocio con sus actividades y restricciones de ejecución entre ellas. Una vez que los procesos de negocio están definidos, pueden ser sujeto de análisis, mejora y promulgación o puesta en marcha.

Business process management incluye conceptos, métodos y técnicas para dar soporte al diseño, administración, configuración, puesta en marcha y análisis de los procesos de negocio.

Un **modelo de proceso de negocio** consiste de un conjunto de modelos de actividades y restricciones de ejecución entre ellas. Una **instancia de proceso de negocio** representa un caso concreto en el negocio operacional de una compañía, y consiste de instancias de actividades. Cada modelo de proceso actúa como un plano para un conjunto de instancias de procesos de negocio, y cada modelo de actividad actúa como un plano de un conjunto de instancias de actividades.

Los **modelos de proceso de negocio** son los elementos principales para implementar **BPM**. Esta implementación puede ser hecha a través de reglas y políticas organizacionales, pero también puede ser hecha por un software, usando un **BPMS (Business Process Management Systems)**.

Un **BPMS** es un software genérico que es manejado por representaciones explícitas de procesos y coordina la realización de dichos procesos. Esta herramienta asiste a las organizaciones con tecnología que facilita la puesta en marcha de proceso de negocio que de otro modo sería manual y guiada por el conocimiento del personal de la compañía y asistidos por las regulaciones y procedimientos organizacionales presentes.

Ejemplo de proceso de negocio: su representación, coreografía y orquestación

Un buen ejemplo es un proceso de orden de compra, porque tiene claridad y su complejidad es reducida. En este proceso, una orden es recibida, una factura es enviada, un pago es recibido y los productos ordenados son enviados. Esta representación textual enumera de actividades del proceso de negocio, pero no hace explícito el orden de acuerdo al cual estas actividades son realizadas. **Notaciones gráficas** son mucho más apropiadas para expresar el orden entre actividades de un proceso de negocio. El proceso consiste en un conjunto de actividades que se llevan a cabo en forma coordinada. La coordinación entre las actividades es lograda por una representación explícita del proceso y las restricciones de ejecución.

Hay varias notaciones gráficas para modelar procesos de negocio, pero su esencia es muy similar. En la notación BPMN -o Business Process Model and Notation-, las actividades se representan con rectángulos redondeados, y llevan un nombre en el centro. Los eventos se usan para marcar por ejemplo el principio y el final del proceso. Los eventos se representan con círculos. Un evento puede ser marcado con un símbolo indicando el tipo del evento. La ejecución de las actividades se expresa con flechas direccionales.

La ramificación y unión de los nodos es representada con diamantes que pueden ser marcados con diferentes símbolos. Actividades concurrentes pueden ser ejecutadas en cualquier orden, y se permite cualquier superposición temporal entre las actividades concurrentes. Un diamante con múltiples flechas entrantes y una sola saliente es un nodo de unión, que mezcla las ramas precedentes.

Una descripción más completa y profunda de BPMN puede encontrarse en el **anexo A**.

La compañía recibirá varias órdenes, cada una de las cuales puede ser procesada como se describe en el diagrama. Esta observación da lugar a conceptos importantes en BPM: **modelos de procesos de negocio** e **instancias de procesos de negocio**.

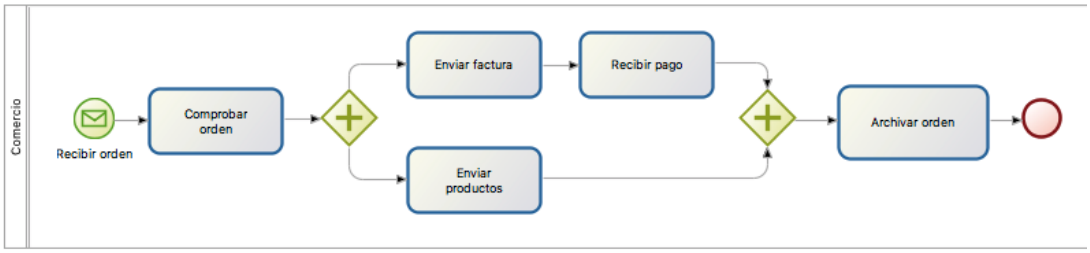

Figura 3.1: Proceso simple de venta de un comercio

Dado que los procesos de negocio por definición se llevan a cabo en una sola organización, el orden de las actividades puede ser controlado por un BPMS como un componente de software central ejecutado por la compañía. Este control centralizado es muy similar al director que controla a los músicos de una orquesta, por lo cual los procesos de negocio también son llamados **orquestaciones de procesos**.

Este proceso de ejemplo interactúa con el proceso de negocio del comprador correspondiente a la venta. El comprador envía una orden, recibe la información para el pago, liquida la factura y recibe los productos solicitados.

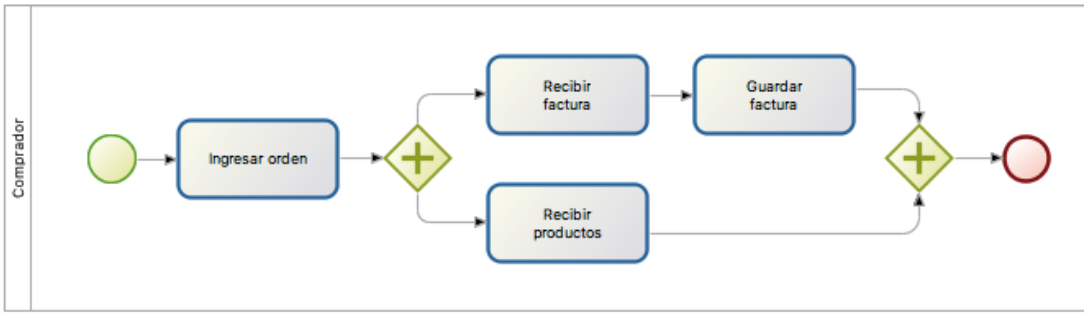

Figura 3.2: Proceso de compra desde el punto de vista del comprador

Los procesos de negocio pueden interactuar entre ellos. Los procesos de negocio del vendedor y del comprador, por ejemplo, pueden interactuar entre ellos de la siguiente forma.

1. El comprador envía un mensaje de orden de compra al vendedor
2. El vendedor recibe el mensaje en un evento de inicio. La información es extraída de la orden.
3. El vendedor envía una factura y envía los productos ordenados.
4. El comprador recibe la factura.
5. El comprador liquida la factura.
6. El comprador recibe los productos.

Las actividades en interacción entre el proceso del vendedor y el proceso del comprador se relacionan con flechas punteadas, que representan flujo de mensajes. El flujo de mensajes puede referirse a mensajes electrónicos o a la transportación de objetos físicos como los productos ordenados.

Las interacciones entre un conjunto de procesos de negocio se especifican en una **coreografía de procesos**. El término coreografía indica la ausencia de un agente central que controla las actividades entre los procesos involucrados. La interacción se logra a través del envío y recepción de mensajes.

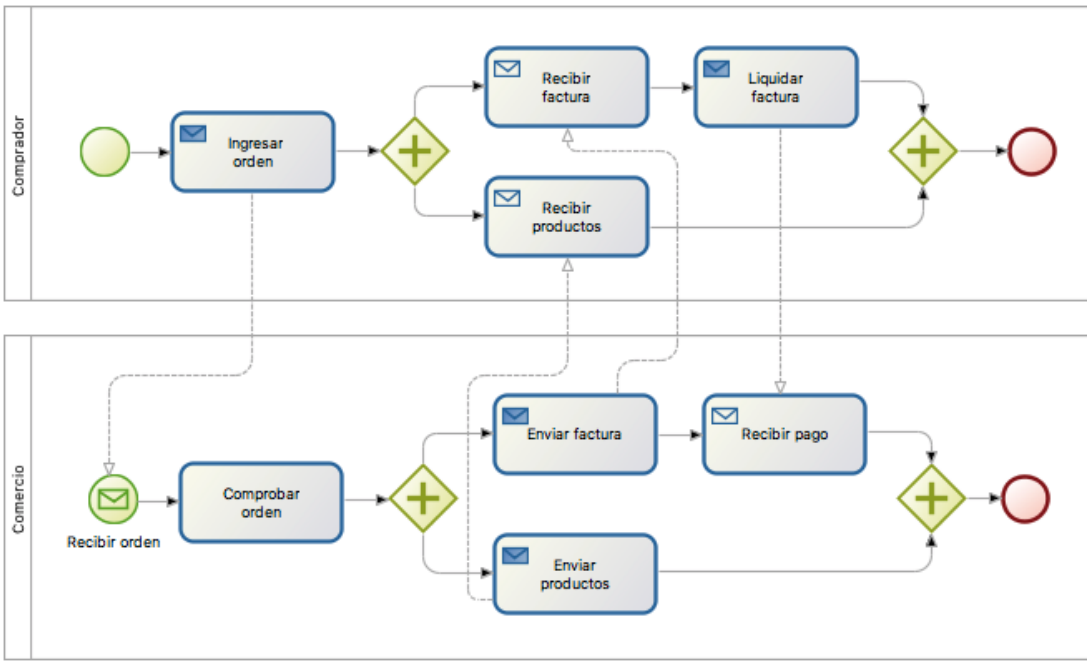

Figura 3.3: Coreografía de procesos de negocio

Esta coreografía de procesos permite múltiples implementaciones concretas, en las cuales el nivel de soporte de software puede variar. Este proceso captura perfectamente formas tradicionales de ordenar productos que no son soportadas por sistemas de información.

Muchas partes de los procesos de negocio mostrados en la figura pueden ser implementadas por sistemas de software. El comprador podría usar un navegador web para buscar en un catálogo en línea del vendedor, colocar productos en su carro de compras, proveer una dirección de facturación y presionar el botón de envío.

La definición de procesos de negocio y sus interacciones no prescribe estrategias o plataformas para la implementación. La realización de los procesos de negocio por parte de los participantes puede cambiar sin afectar la interacción entre los procesos.

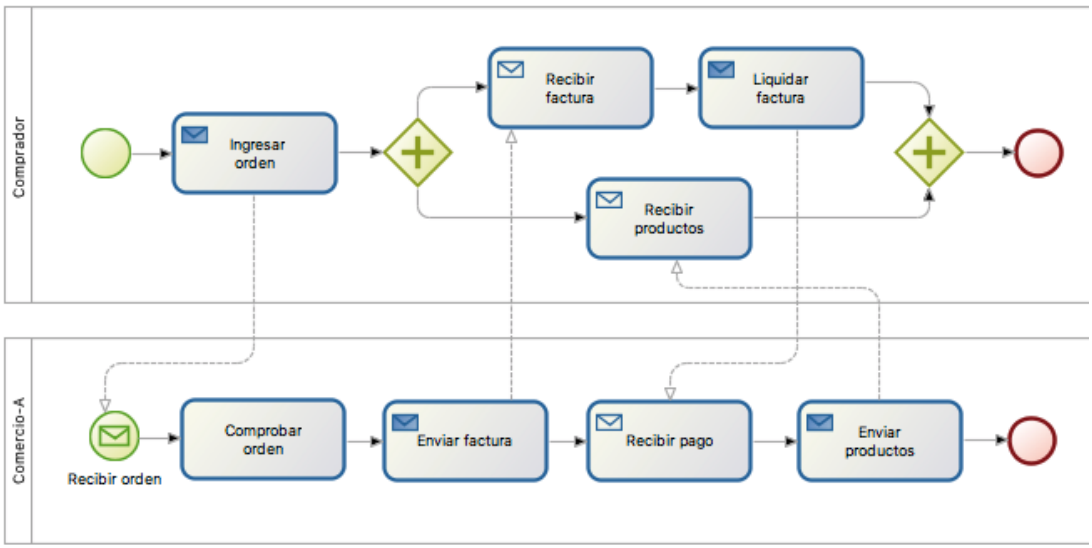

Figura 3.4: Variante del proceso del vendedor

En este nuevo ejemplo, el proceso de negocio del vendedor realiza actividades de forma secuencial. Ya no hay actividades concurrentes. Se introduce la siguiente regla: un producto es enviado sólo luego de haber recibido el pago correspondiente. Este es un enfoque que protege al vendedor de compradores fraudulentos. Dicho proceso trabaja perfectamente en coordinación con el proceso del comprador, que no ha sido modificado. Sin embargo, el tiempo de ejecución puede ser un poco mayor que en el primer caso, dado que algunas actividades podían ser realizadas de forma paralela.

Diferentes **niveles de abstracción** pueden ser identificados en BPM, desde objetivos de negocio de alto nivel y estrategias de negocio hasta procesos de negocio implementados.

Al nivel más alto, se especifican los **objetivos de negocio y las estrategias**. Los objetivos de negocio se refieren a los objetivos a largo plazo de la compañía, mientras que las estrategias se refieren a los planes para alcanzar dichos objetivos.

En el segundo nivel se encuentran los **procesos de negocio organizacionales**. Son procesos de alto nivel que normalmente se especifican de manera textual por sus entradas, salidas, resultados esperados y dependencias con otros procesos organizacionales.

Mientras que los procesos organizacionales caracterizan la funcionalidad del negocio de grano grueso, hay múltiples procesos de negocio operacionales que contribuyen a un proceso de negocio organizacional. En un **proceso de negocio operacional**, las actividades y sus relaciones se especifican, pero los aspectos de implementación de dicho proceso no se tienen en cuenta. Los procesos de negocio operacionales se especifican en modelos de procesos.

Los procesos de negocio operacionales son la base para el desarrollo de procesos de negocio implementados. Éstos contienen información de la ejecución de las actividades de los procesos y el ambiente técnico y organizacional en el cual serán ejecutadas.

3.2. Ciclo de vida de los procesos

El **ciclo de vida de los procesos** consiste de fases que se relacionan unas con otras. Las fases están organizadas en una estructura cíclica, mostrando dependencias lógicas. Estas dependencias no implican un orden temporal en el cual las fases deben ser ejecutadas. Muchos diseños y actividades de desarrollo son conducidos durante cada una de estas fases, y los enfoques incrementales y evolutivos que involucran actividades concurrentes son muy comunes.

El ciclo de vida de los procesos comienza en la **fase de Diseño y Análisis**, en la cual se conducen encuestas de los procesos de negocio y el ambiente técnico y organizacional. En base a estas encuestas, los procesos de negocio se identifican, revisan, validan y representan en modelos.

Modelos de proceso explícitos expresados en notación gráfica facilitan la comunicación, para que diferentes participantes puedan comunicarse de forma eficiente, refinar los modelos y mejorarlos.

Una vez que el diseño original del proceso de negocio es desarrollado, necesita ser validado. Un instrumento útil para validar procesos es un taller en el cual las personas involucradas discuten acerca de los mismos. Las técnicas de simulación pueden ser usadas para dar soporte a la validación, porque al simular los procesos se pueden dejar al descubierto deficiencias en los modelos.

La siguiente fase es la **Configuración**. Una vez que el modelo de proceso es diseñado y verificado, el proceso de negocio debe ser implementado. Puede implementarse sin utilizar un BPMS. No obstante,

en caso de que un software dedicado sea usado para desarrollar el proceso de negocio, la plataforma de implementación se elige durante la fase de configuración. El modelo de proceso es enriquecido con información técnica que facilita la puesta en marcha del mismo a través del BPMS.

Una vez que el sistema está configurado, la implementación del proceso de negocio debe ser probada. A nivel proceso, es importante realizar pruebas de rendimiento e integración para detectar posibles problemas de ejecución durante la fase de configuración. Una vez que la subfase de pruebas es completada, el sistema se despliega en el ambiente destinado para su funcionamiento.

La tercera fase es la **puesta en marcha**. Esta fase está compuesta por la ejecución del proceso. Se crean instancias del proceso para satisfacer los objetivos de negocio de la compañía. La iniciación de instancias de proceso se produce por un evento definido, por ejemplo, la recepción de una orden de compra enviada por un cliente.

Un componente de monitoreo del BPMS visualiza el estado de las instancias de procesos. El monitoreo de procesos es un mecanismo importante para proveer información precisa acerca del estado de las instancias. Esta información es valiosa por ejemplo para responder a un pedido de un cliente que consulta acerca del estado actual de su caso.

Las técnicas de visualización pueden estar basadas en colores, de forma que por ejemplo, una actividad habilitada se muestra verde, una instancia en ejecución se muestra azul y una instancia finalizada o completada se muestra en color gris. La mayoría de los BPMS proveen información de monitoreo basada en el estado de los procesos de negocio activos.

La última fase del ciclo es la **fase de Evaluación**. En ella se utiliza la información disponible para evaluar y mejorar los modelos de procesos de negocio y sus implementaciones. Los logs de ejecución se evalúan usando técnicas de process mining y monitoreo de actividades de negocio. Estas técnicas pretenden identificar la calidad de los modelos de procesos y la adecuada ejecución del ambiente.

Por ejemplo, el monitoreo de actividades de negocio podría identificar que una cierta actividad tarda demasiado tiempo debido a una falta de recursos necesarios para realizarla. Dado que esta información es útil además para simular procesos, estas dos fases están íntimamente relacionadas.

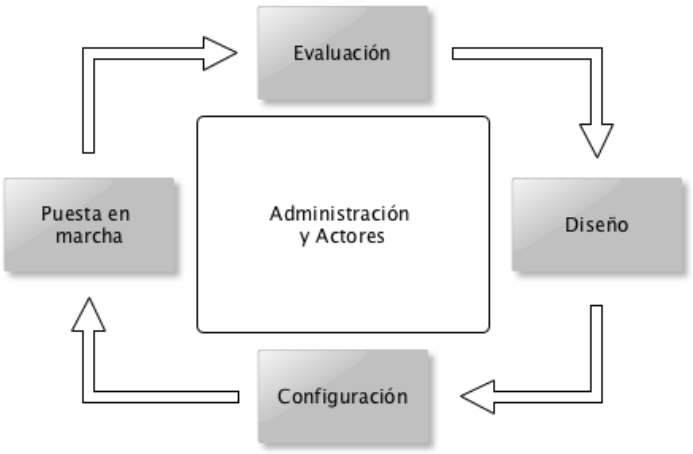

Figura 3.5: Ciclo de vida de los procesos de negocio

3.3. Automatización de procesos y BPMS

En esta sección se realiza una introducción a los **Sistemas de Gestión de Procesos de Negocio**, o **BPMS** por sus siglas en inglés, que consisten en herramientas de desarrollo integrales para la definición de modelos de proceso, de los roles involucrados, el mapeo de datos, integración de servicios, diseño de interfaces visuales para la interacción humana, diseño de coreografía y orquestación de procesos y la ejecución de los mismos.

La característica más elementalmente distintiva de un BPMS es que es un sistema informático consciente de los procesos, pensado para los procesos. Existen distintos BPMS en el mercado, con una gran variedad de funciones, desde simples sistemas que solo se encargan del diseño y automatización de procesos a sistemas más complejos que involucran inteligencia de procesos, procesamiento de eventos complejo, monitoreo, funcionalidades SOA, integración con aplicaciones de terceros y redes sociales. A pesar de la gran variedad de funcionalidades que pueden ofrecer, la principal característica que hace a un BPMS es la automatización de procesos de negocio.[8]

Los procesos se describen de forma tal que los mismos BPMS pueden realizar la ejecución de los mismos.

Arquitectura de un BPMS

La siguiente figura muestra la arquitectura de un BPMS, concretamente el motor de ejecución, la herramienta de modelado de procesos, el manejador de listas de tareas y la herramienta de administración y monitoreo.

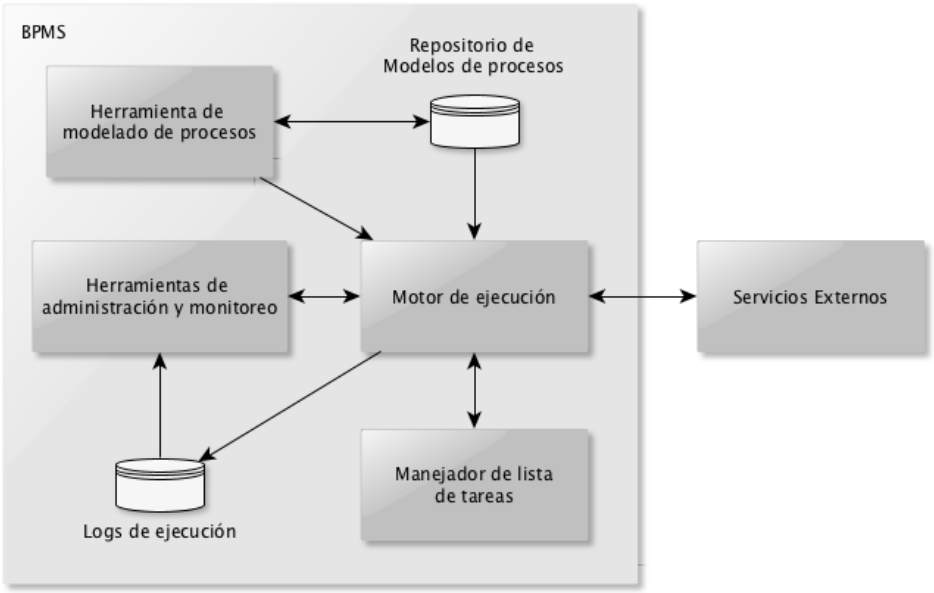

Figura 3.6: Arquitectura de un BPMS

El **Motor de ejecución** es central a un BPMS. Provee diferentes funcionalidades incluyendo: la capacidad para crear instancias ejecutables de procesos, la habilidad para distribuir trabajo entre los participantes de los procesos para ejecutar un proceso de principio a fin y la habilidad de obtener y almacenar automáticamente datos necesarios para la ejecución del proceso y delegar actividades a aplicaciones de software a lo largo de la aplicación. Adicionalmente, el motor está continuamente monitoreando

el progreso de las diferentes instancias y coordinando cuáles actividades se deben atender a continuación a través de la generación de ítems de trabajo, es decir instancias de actividades de procesos que deben ser atendidas para instancias de proceso específicas. Los **ítems de trabajo** son asignados a recursos que estén calificados y autorizados a llevar adelante cada tarea.

La **herramienta de modelado** ofrece funcionalidades como la habilidad de los usuarios de crear y modificar modelos de proceso y la habilidad de generar anotaciones agregar definiciones en los procesos con datos adicionales como entradas y salidas de datos, participantes, reglas de negocio, o indicadores de performance claves asociados a una actividad. Adicionalmente, ofrece la habilidad de almacenar, compartir y recuperar modelos de proceso de negocio de un **repositorio de modelos de proceso**. Un modelo de proceso puede ser **desplegado** en el motor para ser ejecutado. Esta acción puede ser realizada directamente desde la herramienta de modelado o desde el repositorio. El motor usa el modelo de proceso para determinar el orden lógico y temporal de las actividades de un proceso que deben ser ejecutadas. En este sentido, determina cuáles ítems de trabajo deben ser generados y a quiénes deben ser asignados o cuáles servicios externos deben ser llamados.

El **manejador de lista de tareas** es un componente de los BPMS a través del cual por un lado los participantes acceden a tareas disponibles para ser realizadas, y por otro lado se comprometen a ellas. El motor de ejecución mantiene un registro de los ítems de trabajo que están pendientes y los disponibiliza a través de listas de tareas pendientes para cada participante individualmente. La lista de tareas pendientes de un BPMS estándar puede imaginarse como una bandeja de entrada, similar a las de los sistemas de correo electrónico. A través de una bandeja de entrada, los participantes pueden ver qué tareas están listas para ser realizadas. El manejador de lista de tareas suele usar formularios electrónicos para el ingreso y lectura de datos. Cuando un ítem de trabajo es seleccionado e iniciado por un participante desde su lista de tareas, el correspondiente formulario se muestra en su pantalla. Este paso es llamado check-out. Los participantes pueden ingresar datos, y marcar la completitud de la tarea. Este último paso se llama check-in. Luego, el motor determina el próximo ítem de trabajo que debe realizarse para el caso en cuestión. A menudo los participantes pueden ejercer cierto control sobre la lista de ítems de trabajo, por ejemplo con respecto al orden en el cual son mostrados los ítems y la prioridad que asignan a cada uno. Además, el manejador normalmente soporta la suspensión temporal de la realización de un ítem y la capacidad de delegar el trabajo a alguien más. Qué funciones concretas estén disponibles depende del BPMS en cuestión y de su configuración específica. Es muy común personalizar los manejadores de tareas, por ejemplo acorde al diseño corporativo, para fomentar el uso eficiente y la aceptación dentro de la organización.

Puede ser útil invocar **servicios externos** durante la ejecución de procesos de negocio. En muchos procesos de negocio, hay actividades que no son ejecutadas de forma manual, al menos no completamente. En estos casos, el motor de ejecución puede simplemente llamar a una aplicación externa, por ejemplo para recibir un informe acerca del análisis crediticio de un determinado cliente. La aplicación externa debe exponer una interfaz de servicio con la cual el motor pueda interactuar. Por ello, simplemente llamamos a estas aplicaciones servicios externos. El motor de ejecución provee al servicio externo de los datos que necesite para realizar su tarea y al completar la petición, el servicio retorna la salida al motor. Esto es almacenado en el registro de ejecución. Algunas actividades no son ni completamente manuales ni completamente automáticas. En cambio, deben ser realizadas por participantes humanos con un cierto

grado de soporte automático. Para esta categoría de actividades, el motor de ejecución va a invocar a los servicios apropiados con los parámetros apropiados, en el momento en que el participante seleccione un ítem de trabajo para comenzar a llevarlo a cabo. Un típico ejemplo sería la invocación de un Sistema de Manejo de Documentos (DMS) para mostrar al participante un documento que es importante para realizar la tarea.

Las **herramientas de administración y monitoreo** son las herramientas necesarias para la administración de todas las cuestiones operacionales de un BPMS. Debemos considerar, como primer ejemplo, la disponibilidad de los participantes. Si una persona se encuentra incapacitada para trabajar debido a una enfermedad o a un período de vacaciones, el BPMS debe tener registro de ello para evitar asignarle ítems de trabajo. Las herramientas de administración se requieren también para lidiar con situaciones excepcionales, por ejemplo eliminar ítems de trabajo obsoletos del sistema. También deben estar equipadas con la función de **monitoreo de procesos**. Uno puede utilizar estas herramientas para monitorear el rendimiento de los procesos de negocio en ejecución, en particular con respecto al progreso de las instancias individuales. Estas herramientas pueden totalizar datos de las diferentes instancias, como tiempo promedio de instancia, porcentaje de tareas atrasadas, y otras métricas relevantes. El BPMS registra la ejecución de un modelo de proceso paso por paso. Los eventos relacionados a la ejecución se almacenan y pueden ser exportados en forma de **registros de ejecución**. Algunas herramientas de monitoreo pueden analizar datos históricos de los registros de ejecución y compararlos con datos en vivo.

La arquitectura general descripta hasta aquí sustenta el funcionamiento de cualquier BPMS.[8]

3.3.1. Ventajas de la utilización de un BPMS

Hay cuatro grandes categorías de ventajas que hacen atractivo para las organizaciones el uso de un BPMS: reducción de carga de trabajo, integración flexible de sistemas, ejecución transparente y aplicación de reglas.[8]

La **reducción de carga de trabajo** se produce porque parte del trabajo lo hace automáticamente el BPMS. En primer lugar, se encarga de trasportar el trabajo él mismo, lo cual elimina la necesidad de cadetes, data-entry, correo, y demás medios para transportar información y tareas pendientes. En segundo lugar, se reduce ampliamente el trabajo de coordinación, ya que el BPMS usa el modelo de proceso para determinar qué actividades deben ser realizadas a continuación y en qué orden. Por último, se reduce el trabajo de recolectar toda la información adicional necesaria para realizar una determinada tarea. Esto puede depender, dependiendo del tipo de procesos de la organización, de la implementación de un sistema de manejo de documentos además del BPMS.

La **integración flexible de sistemas** es el argumento más mencionado para comenzar a utilizar un BPMS. Los BPMS introducen un soporte genérico para el área de lógica de procesos. Los BPMS hacen mucho más fácil el manejo de la lógica de procesos, porque podemos cambiar el diseño de un proceso sin cambiar el código de las aplicaciones que lo implementan y a la inversa podemos cambiar parte de la implementación sin afectar la lógica de negocio. Los BPMS proveen el adhesivo para unir sistemas separados y permitir que colaboren entre sí con un objetivo de negocio.

La **transparencia en la ejecución** es una característica que no se ve a simple vista en un BPMS. Todas las organizaciones tienen procesos, y muchas veces tienen la creencia de que sus procesos realmente se ejecutan como se espera. Pero al implementarlos en un BPMS, debido a los registros que un BPMS

provee de la ejecución de las actividades recientes y actuales y a la información agregada que almacena de actividades pasadas, suele quedar en evidencia una diferencia entre lo que se espera de la ejecución de un proceso y la realidad. Esto es lo que enriquece a las organizaciones orientadas a proceso, porque les permite evaluar sus procesos actuales y aplicar cambios que generen una evolución constante, o una **mejora continua**.

Por último, la **aplicación de reglas** es más precisa con la utilización de un BPMS. Es un beneficio de calidad, uno hace lo que uno promete. Por ejemplo, podemos considerar la separación de deberes que tiene lugar en el dominio de los servicios financieros. El registro y la inspección de una transacción financiera nunca deben ser realizadas por el mismo individuo. Este tipo de regla es fácilmente implementable en un BPMS. El sistema registra qué individuo realizó el registro de una operación, y habilita a cualquier otro usuario excepto a éste a realizar la tarea de inspección.

Desafíos del uso de un BPMS

A pesar de las varias ventajas de utilizar un BPMS, hay algunos obstáculos con respecto a la incorporación en una organización.

El primero consiste en la integración con sistemas antiguos, que puede ser un problema importante dado que muchos sistemas fueron desarrollados sin tener en cuenta en lo más mínimo la posibilidad de que fueran usados en coordinación con otros. Las aplicaciones mainframe aún pueden encontrarse por ejemplo en bancos y compañías de seguros.

A su vez, muchas aplicaciones carecen de la más mínima orientación a procesos. En un sistema conocedor de procesos, casos separados serían tratados por separado, mientras que en muchos sistemas tradicionales, el paradigma dominante es el procesamiento por lotes. Esto no se lleva bien con la filosofía de un BPMS.

La integración de un BPMS operacional a menudo tiene un impacto en varias partes de una organización. Se presentan desafíos a nivel organizacional, como el balanceo de los intereses de los diferentes actores, que tienen distintos objetivos y compiten por los mismos recursos. Por otro lado, llegar a comprender cómo se comportan los procesos implícitos de una organización puede ser un trabajo de meses. Intereses políticos pueden influir también, ya que no todos estarán de acuerdo con los cambios en cómo se llevan a cabo las tareas.

Un factor que agrega complejidad a la introducción de un BPMS es el dinamismo de las entidades. Es extremadamente común que durante la introducción de un BPMS, que puede llevar un par de meses, las reglas de la organización cambien, se eliminen o combinen departamentos, se asignen nuevas responsabilidades. Es por ello que la introducción gradual de este tipo de sistemas es más apropiada que un cambio abrupto.[8]

3.4. Conclusiones

Los BPMS presentan desafíos pero su incorporación puede brindar múltiples ventajas a ciertas organizaciones. El hecho de poder plasmar los procesos de la organización en un sistema que permita gestionarlos y que abstraiga al nivel operativo de la organización del conocimiento integral de todos ellos, es una gran virtud. Si bien las organizaciones pueden funcionar orientadas a procesos y utilizar BPM sin

implementar un BPMS, cuando la organización tiene un tamaño considerable y la gestión de procesos es compleja, la utilización de un BPMS resulta altamente efectiva.

Una vez desarrollados de forma general los temas que conforman el marco teórico de esta tesina, en el próximo capítulo se presentará la propuesta de integración entre estos conceptos.

Capítulo 4

Propuesta de integración de mecanismos de criptografía en BPM

En este capítulo se propone abordar los requerimientos de seguridad de una organización que funcione orientada a procesos. A su vez, se pretende elaborar soluciones informáticas integradas con un BPMS para proveer una mejora en la calidad de los servicios de seguridad.

4.1. Caso de estudio: Coordinación de asistencia médica

Durante el desarrollo de esta tesina tuve la oportunidad de interactuar con un equipo de personas que desarrollan su actividad laboral en una empresa de servicios de asistencia médica.

Esto hizo posible realizar una entrevista y conocer algunos de los procesos que ejecuta la organización para llevar a cabo su misión. Elegí este ejemplo como un caso de estudio para este informe, debido a que incorpora distintos niveles de requerimientos de seguridad, en distintas tareas de los procesos, dependiendo de los actores que se involucran.

La organización presta atención médica a viajeros. Hay varios tipos de actores o roles en juego. Los clientes se comunican por teléfono con los agentes de atención. La coordinación del caso es realizada por otro sector de empleados que se dedica exclusivamente a la gestión y coordinación, sin interactuar telefónicamente con el paciente. Los tipos de atención pueden ser una visita médica, atención derivada a un centro médico o internación. A su vez, cada una puede requerir distintas prácticas.

Los supervisores deben evaluar las decisiones tomadas por los agentes de coordinación y en ocasiones también deben tomar decisiones en cuanto a autorizar o no estudios o prácticas según la situación del cliente, el seguro contratado y el presupuesto.

Por último la organización dispone también de personal médico especialista, que en casos específicos evalúa ciertas solicitudes para determinar si se aprueban o no, y esto impacta en la decisión de los supervisores de autorizar o no el presupuesto y/o las prácticas.

Los principales procesos de la organización tienen que ver con atender y coordinar los casos de asistencia médica. Sin embargo, para abarcar todas las interacciones posibles entre los clientes y los actores de la empresa, no alcanza con un único proceso. El principal es el proceso de Gestión de caso de asistencia, el cual se inicia ante una llamada de un cliente, y es el que inicia la gestión y coordinación del caso.

Adicionalmente, hay otras interacciones específicas dependiendo del tipo de atención requerida, el tipo de incidente, la gravedad, la ubicación geográfica, los antecedentes médicos del cliente, que dan lugar a

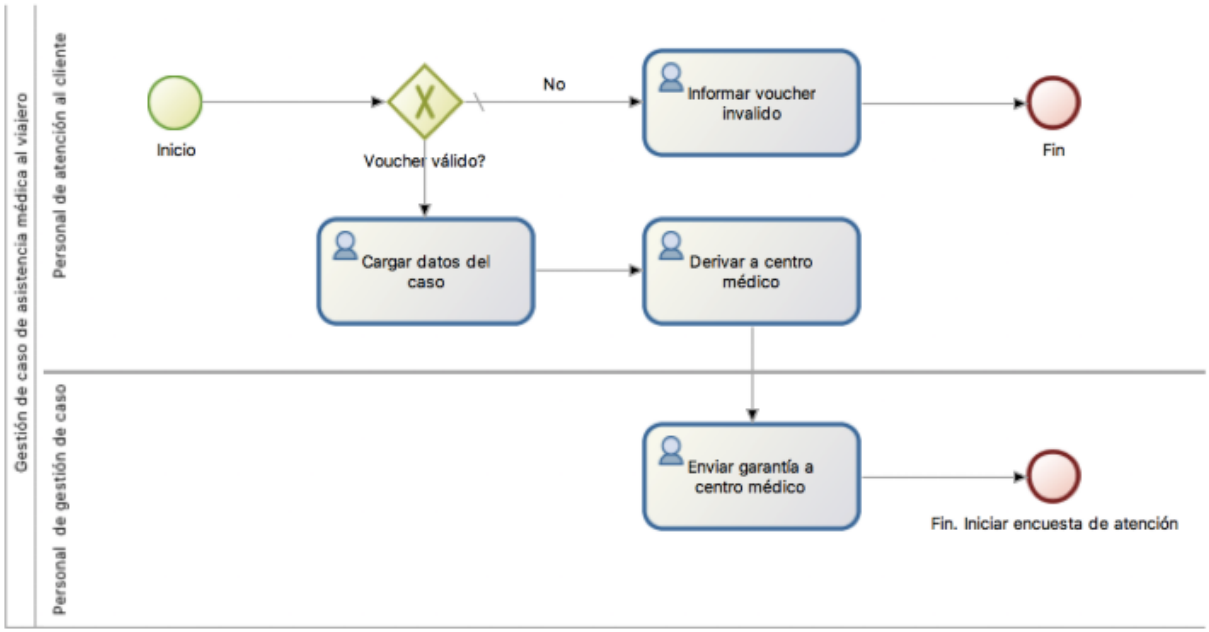

Figura 4.1: Proceso de Gestión de caso de asistencia médica

instanciar otro tipo de procesos, para desarrollar tareas más específicas.

Por ejemplo, coordinar una visita médica, coordinar una asistencia en un sanatorio, coordinar una internación, procesar una solicitud de prácticas, estudios o análisis.

No desarrollaremos todos estos procesos en esta tesina, pero profundizaremos en uno para que nos sea de utilidad práctica. Tomaremos como ejemplo el Proceso de solicitud de prácticas.

Es un proceso relevante como caso de estudio porque involucra a varios roles, contiene tareas en las que un actor de un determinado rol autoriza o rechaza solicitudes, y es un proceso cuyo tiempo de ejecución es crítico y debe mantenerse al mínimo.

Luego de iniciado un caso de asistencia y derivado el paciente a un centro médico cercano a su ubicación, es iniciado por un agente de coordinación bajo demanda del personal médico de la institución prestadora.

Tres de los roles mencionados anteriormente participan en este proceso: el personal de gestión de caso, el personal de supervisión y el personal médico especialista. El **supervisor** y el **especialista** tienen mayores responsabilidades y cada uno tiene en este proceso una tarea con un requerimiento de mayor grado en cuanto a los **servicios de autenticación** y **no repudio**.

El proceso comienza con la carga de la solicitud. Una solicitud tiene un tipo de práctica, y está asociada a un caso.

El agente debe aprobar o rechazar la solicitud. Los agentes están capacitados para evaluar el caso y determinar si se están solicitando prácticas por error. Todo esto estará sujeto a una auditoría, pero eso es otro proceso que no nos ocupa en este momento.

Una vez que el agente realiza un primer filtro detectando posibles errores, si la solicitud está contemplada dentro de un presupuesto que sea posible aprobar de forma automática, el sistema realiza la aprobación completa de la solicitud. El límite para esta regla está dado por el tipo de cobertura y plan que haya contratado el cliente en cuestión.

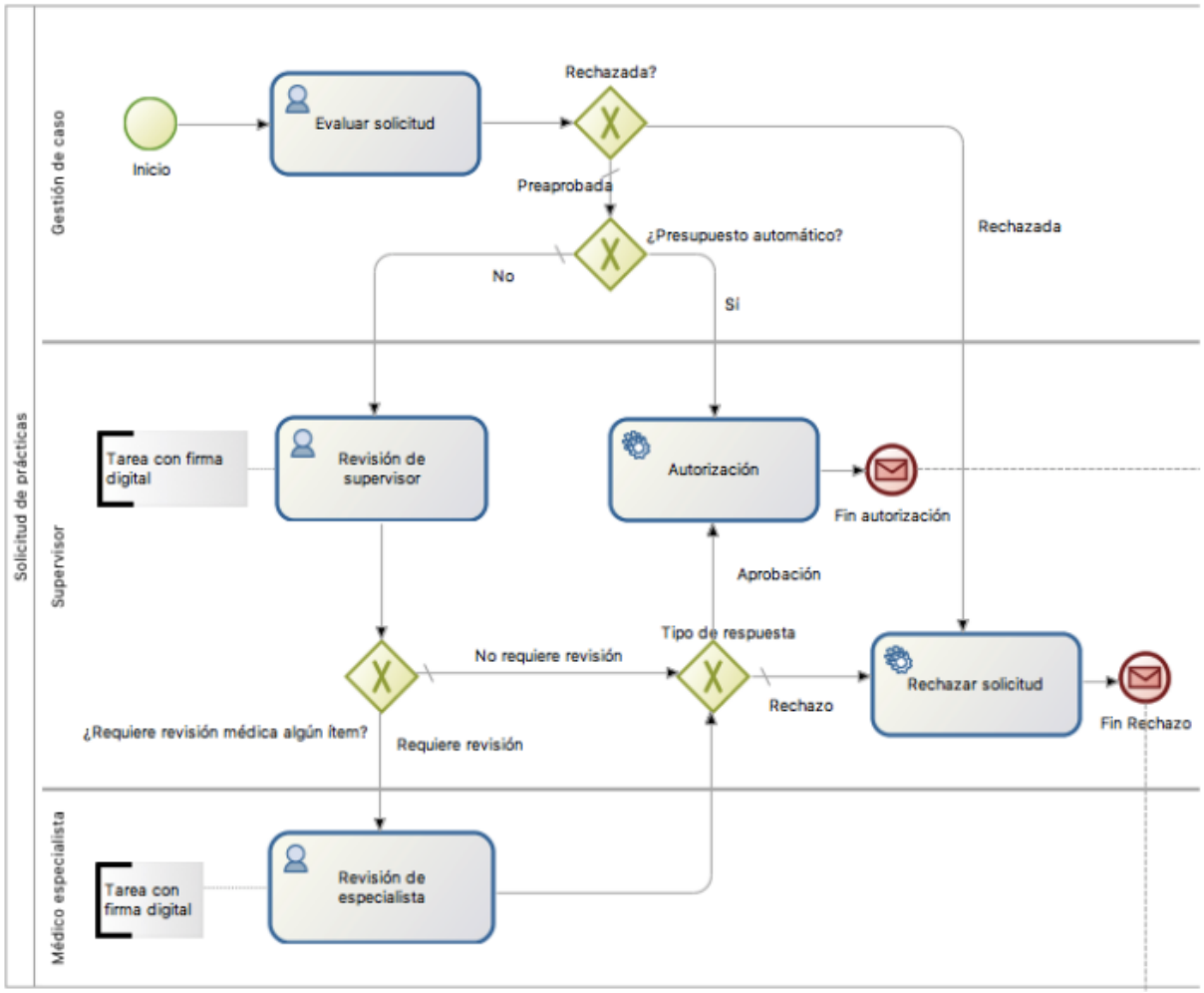

Figura 4.2: Proceso de solicitud de prácticas

Si la solicitud no es apta para aprobación automática de presupuesto, el caso debe ser revisado por un agente supervisor, cuya primera tarea será evaluar los ítems y determinar si alguno debe ser revisado por un médico especialista. Si bien los agentes y los supervisores están capacitados para tomar decisiones en muchos escenarios, frecuentemente el tipo de urgencia o los antecedentes clínicos del cliente ameritan la participación de un especialista para evaluar las solicitudes.

En caso de que sea necesaria la revisión del especialista, el mismo realizará una única tarea analizando la solicitud que le corresponde aprobar o rechazar y dando una respuesta. A continuación, si el especialista dio su aprobación, el supervisor aún podrá aprobar o rechazar la misma en la siguiente tarea. En caso de que el especialista rechace la solicitud, esto da por terminado el procesamiento de la misma. Las dos tareas que determinan la aprobación o rechazo de la solicitud, una del especialista y una del supervisor, tienen un nivel de requerimientos de seguridad más elevado, que es analizado en la siguiente sección.

El proceso no puede demorar. Estamos hablando de una urgencia. Es un proceso crítico que debe realizarse en el menor tiempo posible.

Este caso de estudio nos servirá como ejemplo para abordar los temas de la próxima sección y la implementación del capítulo 5.

4.2. Requerimientos y servicios de seguridad en BPM

Business Process Management es implementado por las organizaciones modernas a través de un BPMS. Los BPMS son, al fin y al cabo, sistemas que se integran con otros sistemas, ya sea internos a la organización, o externos, como servicios o BPMS de otras organizaciones, y que coordinan la participación de los usuarios, llevando a cabo la ejecución de los procesos a través de implementaciones de los modelos definidos en los modelos de proceso.

Un BPMS, como todo sistema, tiene sus servicios de seguridad, enunciados en el capítulo 2: autenticación, control de acceso, confidencialidad, integridad, no repudio, disponibilidad. Las aplicaciones y sistemas con los que él se integran también los tienen.

A su vez, una organización tiene sus requerimientos de seguridad que pueden ser más altos o más bajos dependiendo de varios factores. No necesariamente la criticidad será igual para todos los servicios de seguridad. Cada organización podría requerir más o menos servicios para proteger sus datos y a su vez cada servicio puede tener requerimientos de distinto nivel.

Por ejemplo, un instituto que publica el valor de sus cursos e información para la inscripción, tiene un requerimiento nulo de confidencialidad sobre esa información, debido a que precisamente, es pública. Por otro lado, el requerimiento de autenticidad e integridad es importante, y puede ser alto debido a que la organización puede perder mucho prestigio y dinero. Si un atacante modificase esa información pública e hiciera suplantación de identidad para hacer falsa publicidad o derivar a otros sitios a los usuarios, los servicios de integridad y autenticidad serían violados.

El caso de estudio enunciado en la sección anterior, puntualmente el proceso de Solicitud de prácticas es un excelente ejemplo para ilustrar los diferentes niveles de seguridad requeridos por diferentes tareas incluso dentro de un mismo proceso.

Este tipo de proceso tiene distintos niveles de seguridad en distintos servicios de seguridad para distintas tareas. Los agentes tienen requerimientos de seguridad en lo que respecta a los servicios de **autenticación** e **integridad**. Es importante que sólo verdaderos agentes tengan acceso al sistema, sobre todo si la interfaz es web.

Sin embargo, el nivel de estos requerimientos es mucho mayor cuando se trata de las tareas que realizan los supervisores, y podemos agregar el requerimiento del servicio de no repudio. Si un supervisor dice no haber aprobado una solicitud, puede alegar la posibilidad de un fraude producido por un atacante externo y eludir responsabilidades sobre sus acciones.

Igualmente importante es el nivel de seguridad que se requiere para las tareas realizadas por los médicos especialistas. Ellos son la última instancia y tienen la mayor autoridad a la hora de tomar decisiones sobre cada caso, y los requerimientos de servicios de autenticidad, integridad y no repudio son altos.

Proveer seguridad para todas las tareas puede ser costoso, por varios motivos.

- En primer lugar porque la implementación es costosa.
- En segundo lugar porque agrega complejidad a los sistemas, y probablemente a los procesos.
- En tercer lugar porque como vimos en el capítulo uno, la integración de mecanismos de seguridad tiende a hacer menos amigables los sistemas para la interacción humana.

Integrar mecanismos de seguridad en todos los niveles del proceso implicaría capacitar a todos los empleados, sin importar su rol, y multiplicar el costo de la generación de herramientas para la implementación de dichos mecanismos, como generación de claves privadas/públicas o certificados.

Por lo tanto, una organización con estas características, probablemente optaría por implementar mecanismos de seguridad para cada proceso según el rol o según la tarea. Los usuarios que tengan roles que requieran mayor seguridad o cuyos roles tengan participación en tareas que requieran mayor seguridad, deberían ser provistos de las herramientas y capacitados para la utilización de las mismas.

Un BPMS generalmente incluye sólo un mecanismo de autenticación a través del inicio de sesión web con nombre de usuario y contraseña. Para algunas organizaciones esto puede ser suficiente. Sin embargo, hemos visto ejemplos en los cuales es necesario incorporar mejores servicios de seguridad.

En la siguiente sección se introducirán dos propuestas para mejorar la seguridad de un BPMS para escenarios como el descrito. La primera es la utilización de HTTPS, dirigida a fortalecer la integridad y confidencialidad de los datos que viajan en la red, un cambio que es muy sencillo y a su vez realmente importante como base de otros mecanismos de seguridad. La segunda es la incorporación de firma digital, con el objetivo de brindar un buen servicio de autenticación y no repudio. [13]

4.3. BPMS sobre un esquema HTTPS

Los BPMS modernos se basan en una arquitectura 100 % web. El servidor ejecuta el motor de procesos, almacena los datos de usuarios y de modelos de procesos, y sirve la aplicación web que los usuarios utilizan para interactuar con el sistema. Los usuarios utilizan como cliente un navegador web moderno, y de esta forma se conectan con la interfaz web provista por el BPMS para iniciar sesión, ver sus tareas y realizarlas, o realizar labores de administración.

Sobre esta arquitectura web tradicional, incorporaremos un mecanismo de seguridad que es agnóstico del tipo de aplicación web que está sirviéndose, y que es una base fundamental y un pilar sobre el cual se pueden establecer otros mecanismos de seguridad más avanzados. Hablamos de HTTPS.

Cuando una aplicación web funciona sobre HTTPS, se encriptan el URL solicitado, el contenido, el contenido de los formularios web, las cookies y los encabezados HTTP.

Implementar HTTPS es relativamente sencillo en cualquier servidor web, y el valor que aporta en cuanto a seguridad es muy atractivo. Como vimos en el capítulo 2, **HTTPS** no es un protocolo separado, se refiere el uso del HTTP ordinario sobre una conexión con Seguridad de la Capa de Transporte (**TLS**). La identidad del servidor es autenticada a través de un certificado de confianza, por lo cual se garantiza el servicio de autenticación del servidor. La conexión TLS es privada ya que se usa criptografía simétrica para encriptar los datos, lo cual brinda el servicio de confidencialidad. Y dado que cada bloque de datos se concatena con un MAC, la conexión brinda también integridad.

Para la implementación se requiere:

- Acceso a la configuración del servidor
- Un certificado firmado por una Autoridad de Certificación reconocida

En el capítulo 5 se desarrolla el proceso de implementación y puesta en marcha. Es una ganancia de seguridad muy importante, ya estamos garantizando tres servicios a un costo de implementación relativamente bajo.

4.4. Autenticación y no repudio: Incorporación de firmas digitales

Si bien TLS soporta la autenticación del cliente a través de un certificado de cada usuario, no es una práctica común la utilización de este servicio. Algunas razones por las cuales no es una característica utilizada en la práctica son:

- **Ignorancia**. Casi ningún usuario en internet sabe qué son o para qué sirven los certificados.
- **Conveniencia**. Evita la necesidad de configurar los certificado en cada dispositivo que el usuario utilice.
- **Soporte**. Muchos usuarios necesitarían ayuda para configurar sus certificados o para resolver problemas como perderlos o no entender cómo utilizarlos. Adicionalmente varias versiones de Internet Explorer y otros navegadores no lo soportan y la configuración varía de navegador a navegador, incluyendo los navegadores móviles.
- **Complejidad**. Se requiere código adicional del lado del servidor para validar los certificados del cliente.
- **Tendencia**. Ninguno de los sitios más importantes de internet lo están utilizando hoy en día, por lo cual no es un requerimiento que se vuelva tendencia ni un punto por el cual competir a nivel de servicios de seguridad brindados.
- **Complacencia**. La mayoría de las compañías encuentran suficiente protección con el uso de contraseñas y/o segundo factor de autenticación.

Debido a esto, hay dos requerimientos de seguridad que quedan descubiertos con HTTPS, y son **autenticación del usuario** y el **no repudio**. La firma digital nos suple de un mecanismo que abarca los aspectos de seguridad relacionados con estos dos requerimientos.

En primer lugar, la **autenticación** queda **parcialmente descuidada**, por ello nos referimos a la **autenticación del usuario**. El usuario puede confiar en la identidad del servidor cuando la conexión es HTTPS y el certificado del servidor está firmado por una CA reconocida, pero el servidor no tiene el mismo nivel de confianza en la identidad del usuario.

En segundo lugar, el usuario podría alegar que alguien averiguó su contraseña y evadir responsabilidades por una acción realizada en una actividad determinada. Es decir que la calidad del servicio de seguridad de **no repudio** que se está brindando es pobre.

Esto se debe a que el usuario se autentica simplemente a través de un nombre de usuario y una contraseña. La **firma digital**, en cambio, provee un nivel de seguridad mucho mayor en cuanto a la **autenticación** y el **no repudio**.

4.5. Firmado de tareas

Las firmas digitales ya se utilizan hoy en clientes de correo, generadores de PDF y otros sistemas con los que podría interactuar un BPMS. Si bien el mismo sistema BPM podría incorporar la funcionalidad de firmar documentos o emails, esto no sería una verdadera integración de firma digital con BPM, sino una integración más con un servicio más dentro de un proceso de BPM común y corriente.

La parte central de una integración de firma digital para enriquecer la seguridad de una organización que implementa BPM es la **firma de tareas**. En un entorno BPM cobra sentido pensar en que el objeto a firmar sea una determinada tarea de un proceso.

Como se explica en la sección anterior, con el uso de una conexión segura a través de HTTPS, el usuario del sistema BPM puede confiar en la identidad del servidor y en la confidencialidad e integridad de los datos que el servidor le hace llegar.

En esta relación de confianza, imaginemos un **supervisor A** que está realizando una tarea y el sistema BPM informa que un determinado **usuario B** ha realizado otra tarea importante previamente. Él puede estar seguro de que esa información es exactamente la que el sistema BPM le envió, y no fue modificada en el camino entre el servidor y el cliente a través de la red, gracias al uso de HTTPS.

Ahora bien, retrocedamos en el tiempo, y formulemos la siguiente pregunta: ¿Puede el servidor estar seguro de que el usuario B realizó esa tarea?

Podemos solicitarles a los usuarios que recuerden múltiples contraseñas, con cuatro tipos de caracteres, entre ellos mayúsculas, minúsculas, números y símbolos. Podemos pedirles que cambien su contraseña cada 30 días.

Aún así, todo esa complejidad que agregamos al inicio de sesión del usuario y al manejo de sus contraseñas, termina siendo contraproducente. Por un lado, el uso de los sistemas se vuelve cada vez más engorroso. Por otro lado, puede suceder que por tener tantas contraseñas para recordar, él termina por anotarlas en un cuadernos que deja en la oficina.

La posibilidad de **firmar tareas** con una **clave privada** brinda un incremento significativo de la seguridad brindada en torno a los servicios de **autenticación** y **no repudio** del usuario, con un impacto reducido en la complejidad de la interacción del usuario.

Una vez que incorporamos un mecanismo de estas características, podemos decir que el servidor tendrá una certeza mucho mayor de la autenticidad del usuario. En ese caso el **supervisor A** puede confiar realmente, porque no sólo sabe que lo que le llega del servidor es certero y auténtico, también confía en que lo que previamente llegó de otro cliente -el usuario B- al servidor era certero y auténtico.

Para explicar el firmado de tareas, podemos recurrir a un subproceso que se asocie a cada tarea firmable.

El proceso anterior explica el subproceso abstracto realizado en cada tarea que requiera firma digital. Los actores participantes del subproceso de Firma de Tarea son el Usuario, es decir el humano que realiza las tareas, la aplicación cliente, es decir la porción de la aplicación del BPMS que se ejecuta en el navegador del lado cliente de la comunicación HTTPS, y el servidor o sistema BPM. Éste último verificará la firma basándose en la clave pública del usuario en cuestión.

Los requerimientos técnicos son:

- Disponer de una aplicación cliente con la capacidad de generar una firma digital y adjuntarla a los demás datos de la tarea en cuestión.
- Disponer de un servidor con la capacidad de verificar dicha firma, que conozca con certeza la clave pública del usuario que la está completando.

Este proceso es abstracto. Es un diagrama de procesos que nos permite representar el flujo de actividades del mecanismo de firma y verificación con la notación BPMN. En el capítulo 5 se describe un

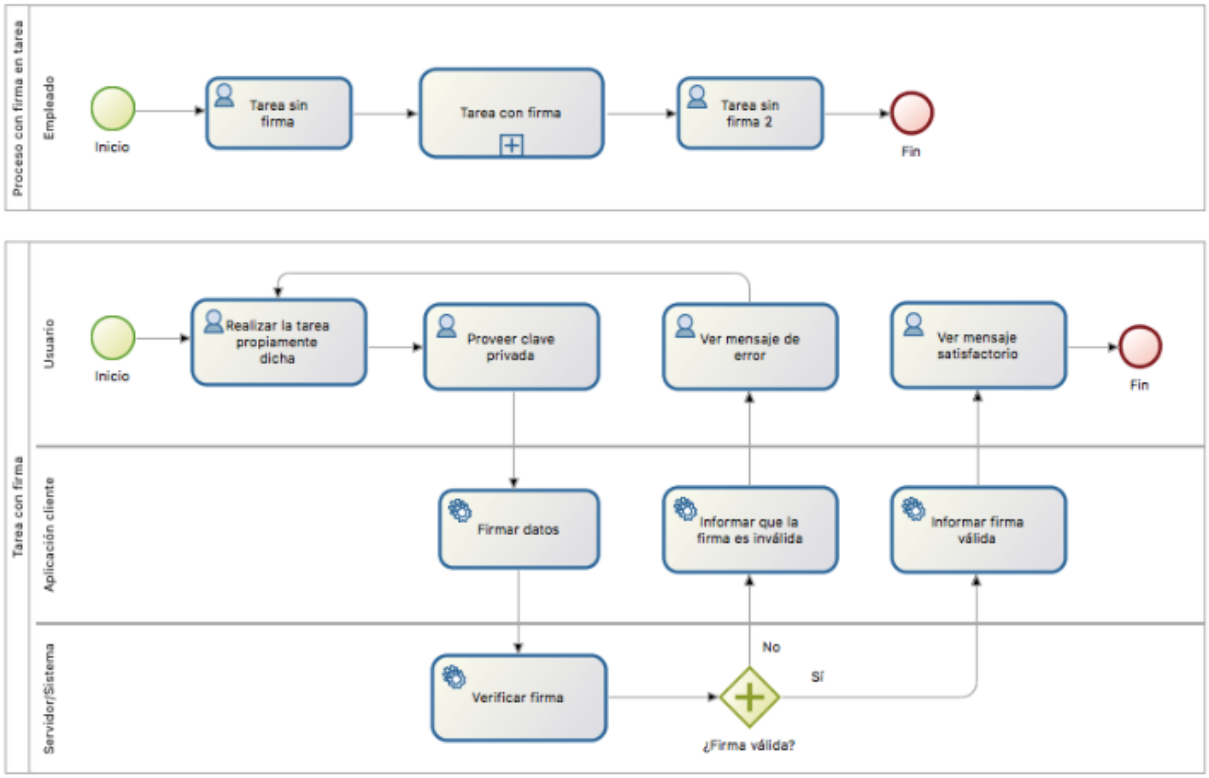

Figura 4.3: Explicación del proceso de firma de tarea y verificación

mecanismo concreto para firmar y verificar la firma de tareas, que en lugar de basarse en un subproceso, se basa en funcionalidades provistas por un BPMS concreto: Bonita Studio.

4.6. Desafíos de la implementación de un esquema de firma digital

Implementar firma digital no es tan sencillo como implementar HTTPS. Implica que cada usuario que deba firmar las tareas cada vez que sea requerido por el sistema, deberá disponer de una clave privada y una clave pública y conocer cómo proteger su clave privada. Se deben proveer mecanismos para:

1. la generación de claves privadas y públicas
2. la distribución de claves privadas, para que cada usuario pueda conservar la suya en algún sistema de archivos
3. utilizar la firma digital, es decir, permitir al usuario seleccionar su archivo de clave privada y firmar una determinada actividad de un determinado proceso con dicha clave
4. la protección de claves privadas, permitiendo al usuario denunciar una clave si ha sido comprometida, y poder obtener una nueva posteriormente

Adicionalmente, se presentan los siguientes **desafíos** para la incorporación de un **firmas digitales en un entorno BPM**.

1. **No es transparente para el usuario**
   Con HTTPS los usuarios deben ser conscientes de un concepto mínimo de seguridad referido a la forma en que su navegador web muestra si una página es segura. Más allá de eso, no requiere ningún

tipo de conocimiento adicional, no requiere más intervención del usuario que una simple mirada al ícono con un candado verde a la izquierda del URL. Se limita la necesidad de participación del usuario. En cambio en el caso del uso de firma digital, el usuario es partícipe en gran medida del mecanismo. Él debe seleccionar su clave privada de su sistema de archivos y debe entender la importancia de que la clave permanezca segura.

2. **Los usuarios necesitarán una capacitación de seguridad y firma digital**
   Debido a lo enunciado en el ítem anterior, será necesaria una capacitación para todos los usuarios que comiencen a utilizar firmas digitales. Serán aquellos cuyos roles involucren actividades con altos requerimientos de seguridad.
   Deberán entender los conceptos básicos del mecanismo de seguridad, como el concepto de clave privada y pública, la importancia de la seguridad de su clave privada y las implicancias del uso de firmas digitales. Debe conocer cómo proteger su clave privada y cómo actuar si la misma resulta comprometida.
3. **Los BPMS populares no brindan soporte específico para usar firma digital de forma nativa.** Si bien algunos BPMS brindan herramientas que podría resultar útiles para el uso de firmas digitales, ninguno de los más populares ofrece un soporte específico para ello.
4. **El uso de firmas digitales implica la distribución segura y eficiente de claves**
   Una de las partes más importantes de un esquema de firma digital es la distribución de claves. La distribución de claves puede darse de dos sentidos: del cliente al servidor o del servidor al cliente.
   Si las claves se generan en el cliente, el mismo debe hacer llegar su clave pública a un servidor para autenticarla (CA) y luego descargar su clave autenticada, siempre a través de una conexión segura. A continuación enviarla a un servidor que actúe como un directorio, que por supuesto podría ser el mismo que actúa como CA, y omitir este paso. Luego el directorio repartiría la clave entre los demás participantes del esquema que deban confiar en el autor.
   Las claves también pueden generarse en el servidor. Este puede ser el caso de clientes livianos, donde el servidor es la misma CA que genera la clave y la firma con su certificado de mayor jerarquía. Luego, a través de una conexión segura, deberá entregar la clave privada al usuario. En este caso el usuario debe confiar plenamente en el servidor de Certificación, pues es éste quien genera su clave. Debemos notar que en ambos escenarios es vital la disponibilidad de una conexión segura y confiable con la Autoridad de Certificación.
5. **El uso de firmas digitales implica lidiar con claves comprometidas**
   Si una clave resulta comprometida, es decir se cree que su seguridad podría haber sido violada o se constata un hecho de violación, se deben tomar medidas para invalidar la utilización de la clave e impedir la falsificación de la identidad del dueño.
   Este proceso implica:
   - Comunicar este tipo de situaciones a la entidad que actúa como directorio.
   - Dicha entidad debe invalidar la clave pública del dueño, para que los demás participantes, si los hay, no confíen en ella.
   - Generar una nueva clave, y autenticarla.
   - Distribuir la nueva clave.

Todos estos desafíos implican un cierto grado de complejidad que debe ser manejada.

4.7. Infraestructura de firma digital orientada a procesos

De los puntos 1, 4 y 5 enunciados anteriormente, se desprende que el uso de firmas digitales implica procesos adicionales, ajenos al negocio. Estos son como mínimo la **generación y distribución de claves**, la **revocación y renovación de claves comprometidas y vencidas** y la **baja de claves y usuarios**.

En un entorno BPM sería lógico incorporar dichos procesos a la batería de procesos de una organización. De esta forma, retomando el ejemplo del sistema BPM de una empresa aseguradora, podríamos pensar en incorporar a su lista de procesos, los dos mencionados en el párrafo anterior. En dicho ejemplo, antes de que un nuevo supervisor se incorpore a la compañía, debería llevarse a cabo el proceso de alta de un nuevo usuario, y la generación de un nuevo par de claves pública-privada para el mismo. Ese proceso, sería igual a cualquier otro proceso de la compañía, se ejecutaría como una instancia de un proceso del BPMS.

De esta forma BPM y firma digital se pueden integrar enriqueciéndose mutuamente, proponiendo que la organización administre su propia **infraestructura de firma digital**, también **orientada a procesos**.

4.8. Procesos de Contratación e Integración de Personal

Serán varios los procesos necesarios para administrar el mecanismo de firmas digitales. El primero y principal será la misma **alta de pares de claves pública-privada**. El objetivo es que la gestión de claves se realice a través de un proceso. En este caso, se presentan dos alternativas.

La primera consiste en implementar el alta de claves pública-privada de forma independiente al alta de usuarios, y mantener el mecanismo tradicional de alta de usuarios, a través del menú de administración del BPMS. Esta opción implica que deberá primero darse de alta el usuario nuevo a través de dicho menú, y a continuación, si es necesario un par de claves para dicho usuario, se procederá a iniciar una instancia del proceso de alta de claves de firma digital para el mismo.

La segunda alternativa es disponer de un proceso de **alta de usuarios**. En la mayoría de los BPMS, el alta de usuarios no se modela a través de un proceso, sino que se realiza manualmente como parte de las tareas de administración cotidianas del BPMS. Todos BPMSs populares proveen un menú de administración a través del cual el personal autorizado puede configurar usuarios, roles, permisos y jerarquías en la organización. No obstante, la acción de dar de alta un usuario en la organización puede modelarse a través de un proceso, lo cual no sólo implica implementar la acción de configurar el usuario, sus roles y permisos en el sistema de BPM, sino también modelar todas las implicancias de la incorporación de un nuevo miembro a la organización. Entre ellas la ejecución de un subproceso de generación de un par de claves para los usuarios que esto sea necesario.

El proceso de **Proceso de Contratación e Integración de Personal** describe las actividades realizadas por diferentes actores a la hora de incorporar a un nuevo miembro a la compañía. Los roles participantes son Recursos Humanos, Administración de tecnología y el nuevo empleado.

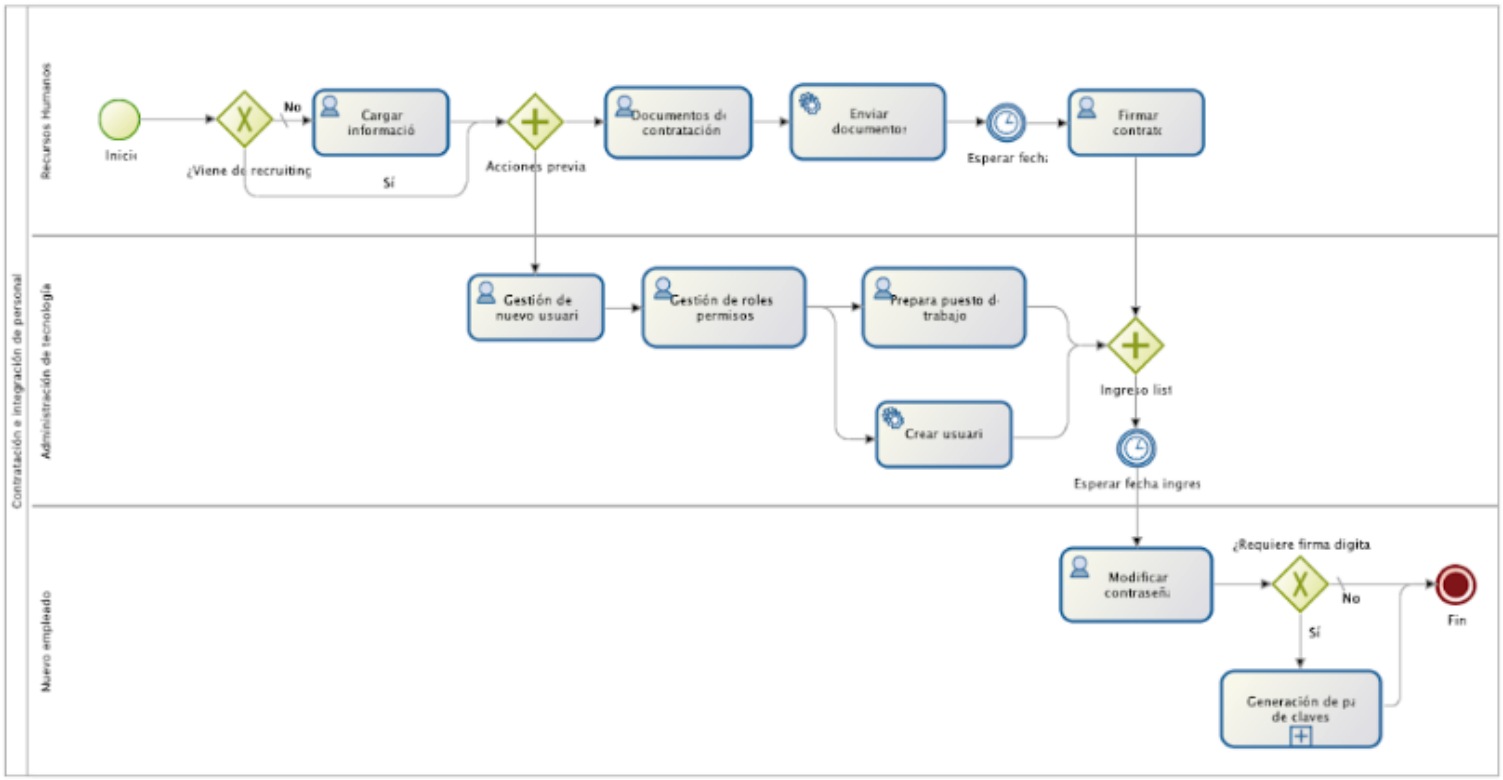

Figura 4.4: Proceso de Contratación e Integración de Personal

El proceso comienza ya sea como parte de un proceso previo de selección de personal -recruiting- o como un proceso independiente. En el caso de que el nuevo empleado provenga de un proceso de selección de personal, no es necesario cargar los datos del mismo. Caso contrario la primera actividad del proceso consiste en **cargar los datos** del nuevo empleado. Esto incluye datos personales, fecha estipulada de firma de contrato, fecha estipulada de ingreso a la compañía, entre otros.

A continuación comienzan dos flujos de actividades que se pueden ejecutar en paralelo.

El primero involucra al área de **Recursos Humanos**, que deberá adjuntar al proceso la **documentación de contratación**. A continuación el sistema enviará dicha documentación al nuevo empleado. Luego se **espera la fecha de firma** de contrato, y llegado el día estipulado se **realiza la firma** del mismo.

El segundo flujo involucra al área de **Administración de tecnología**. La primera actividad de este flujo es gestionar el alta de un nuevo usuario, ingresando un nombre de usuario y una contraseña provisoria. La segunda tarea será gestionar los roles, y por tanto los permisos que tendrá el nuevo miembro de la organización. Por último el sistema creará el usuario y asignará los roles, mientras que el área de Administración de tecnología deberá preparar el nuevo puesto de trabajo.

Una vez finalizadas todas las tareas de ambos flujos, estos se unen para continuar por un mismo camino en el avance de la ejecución del proceso. El siguiente paso es esperar a la fecha de ingreso a la compañía del nuevo miembro. Llegada la fecha, se habilita la primera tarea que puede él o ella puede completar: Modificar su contraseña provisoria y elegir una personal.

Por último, no obstante fundamental para nuestro caso de estudio, si el empleado tiene algún rol asignado que requiera un mayor nivel de seguridad, se ejecutará el subproceso de **Generación de par de**

claves, el cual es necesario para la utilización de firma digital. Caso contrario, el proceso de **Contratación e integración de personal finaliza**.

Con este proceso, no sólo logramos que la gestión de claves forme esté integrada en el proceso de alta de usuarios, sino que además modelamos de manera elegante el proceso completo de ingreso de un nuevo miembro de la compañía.

4.9. Proceso de desvinculación de personal

El proceso de contratación de personal, debe ir acompañado del proceso de desvinculación. Durante la desvinculación de un miembro de la compañía, en el esquema propuesto, las claves deberán darse de baja, al igual que el usuario para el acceso al sistema. A continuación se explican las actividades que forman parte del flujo de ejecución del proceso de desvinculación propuesto.

El proceso tiene dos posibles inicios: por renuncia de un empleado, o por un despido. En el caso de la renuncia, el empleado que renuncia es el iniciador del proceso. En el caso de un despido, el iniciador es un supervisor del mismo.

En el caso de un empleado que renuncia, la primera tarea es el **envío de carta de renuncia**, en la cual se genera la notificación y se envía la carta formal. A continuación se ejecutan dos tareas en paralelo. El supervisor tiene la tarea de notificarse de la desvinculación de uno de sus subordinados. Este flujo del proceso termina aquí. Por otro lado, el empleado debe completar sus tareas antes de la fecha de salida. Esta es una tarea abstracta.

En el caso de un despido, la primera tarea es **iniciar la desvinculación**, y la realiza el supervisor del empleado a desvincular. La siguiente tarea en el caso de este flujo de inicio es **Crear el plan de salida**, a cargo del área de Recursos humanos. Consiste en diseñar un plan de salida para el empleado e informarle personalmente sobre la desvinculación y los pasos a seguir.

En este punto el flujo del proceso toma un camino en común, independientemente del tipo de inicio que se haya dado, y el siguiente paso es esperar la fecha de salida y luego llevar a cabo varias tareas en paralelo.

Personal del área de recursos humanos deberá realizar estas tareas: **Recibir elementos de la compañía**, como computadora, monitor, y otros elementos utilizados por el mismo. Realizar la **entrevista de salida** con el empleado, donde se consulta al mismo acerca de su experiencia en la empresa y se lo asesora en cuanto a su futuro desarrollo profesional. Y realizar las **acciones legales de terminación de contrato**.

También en paralelo con las anteriores, se **dará de baja el acceso al sistema**, y en caso de que el empleado utilizará el mecanismo de firma digital, se darán de baja las claves utilizadas para el mismo. Esta es una parte fundamental de nuestro esquema integrado de gestión de claves de firma digital. Esta actividad es una llamada al subproceso de **Baja de claves**.

Una vez finalizadas todas las tareas anteriores, el área de contabilidad podrá proceder a **liquidar el sueldo restante** del empleado saliente, y sólo en caso de que sea un despido y el mismo sea sin justificación, se deberá liquidar a su vez la **indemnización** correspondiente.

Como tarea final del proceso, el sistema **actualiza automáticamente la nómina** de empleados de la compañía.

4.10. Procesos de gestión de pares de claves

Tanto el proceso de Contratación e Integración de Personal como el proceso de Desvinculación de personal utilizan subprocesos que tienen que ver con la gestión de pares de claves. Estos son **Generación de par de claves** y **Baja de claves**, respectivamente. A su vez definiremos un proceso llamado **Renovación de par de claves** que será llamado por el proceso de Generación de par de claves cuando este último finaliza. Por último, el proceso **Revocación de clave comprometida** será iniciado cuando un empleado o su supervisor sospechen o tengan certeza de que una clave de un usuario ha sido comprometida.

El proceso de **Generación de par de claves** tiene tres eventos de inicio posibles: Por **alta**, es decir se inicia el proceso como un subproceso de Contratación e Integración de Personal. Por **expiración**, cuando una clave generada previamente alcanza su fecha de vencimiento. Por **revocación**, cuando se detecta una clave comprometida y se requiere la renovación de la misma.

En todo el proceso el actor es el empleado. Independientemente del evento que le dé inicio, el proceso comienza con la tarea **Ingresar contraseña de clave privada**, en la que el empleado **ingresa la contraseña** con la que protegerá su archivo de clave privada. A continuación el sistema **genera el par de claves**, determina la fecha de vencimiento, y **almacena la clave privada** en un archivo protegido por la contraseña ingresada en la tarea anterior. El siguiente paso es **almacenar la clave pública** en un registro interno. Y por último el empleado debe descargar su archivo de clave privada y resguardarlo en cualquier almacenamiento seguro.

El flujo termina con un evento de fin, que dispara el inicio del proceso de **Renovación de par de claves**. Este proceso consiste en esperar a la fecha de vencimiento, y una vez ocurrido dicho evento de tipo temporizador, **validar que la clave en cuestión siga existiendo**. Si ya no existe, el flujo termina. Si aún existe, se procederá a llamar al subproceso de Baja de claves. El flujo en este caso termina con un evento de mensaje que da comienzo al proceso de Generación de par de claves nuevamente, invocando el evento de inicio Por **expiración** del mismo.

El proceso de **Revocación de clave comprometida** se utiliza cuando un empleado o su supervisor sospechen o tengan certeza de que una clave de un usuario ha sido comprometida. Hay un evento de inicio para cada uno de esos dos actores. La primera tarea difiere según sea uno u otro el que inicie el proceso, y ambas consisten en **informar de la situación** y explicarla.

A continuación el sistema almacena el motivo de la baja, y se procede a llamar al subproceso **Baja de claves**. Independientemente del actor iniciador, el supervisor del empleado en cuestión deberá analizar si corresponde la generación de un nuevo par de claves, y por último, el proceso termina en cualquier caso. Si se evalúa que corresponde generar un nuevo par, el evento de fin es un mensaje de llamado al proceso de Generación de par de claves.

El último proceso relacionado a la gestión de pares de claves, mencionado varias veces anteriormente, es el proceso de **Baja de claves**. Consiste en marcar la clave pública como obsoleta e informar al usuario asociado a través de un correo electrónico. Ambas tareas son automáticas, realizadas por el sistema. Este proceso es llamado en distintas tareas de otros tres procesos, como vimos anteriormente: el proceso de Revocación de clave comprometida, el proceso de Renovación de claves, y el proceso de Desvinculación de personal.

4.11. Conclusiones

El esquema propuesto se compone de un mecanismo de firma digital integrado que permite firmar una tarea al completarla, sumado a una batería de procesos que modelan la administración de personal, usuarios, roles y claves para la firma digital. En el próximo capítulo se llevará a cabo la implementación de los dos procesos del ejemplo de la sección 4.1 que ilustran estos mecanismos en procesos de negocio con inclusión de tareas que requieren firma digital. A su vez se detallará la implementación de los mecanismos de seguridad enunciados en este capítulo.

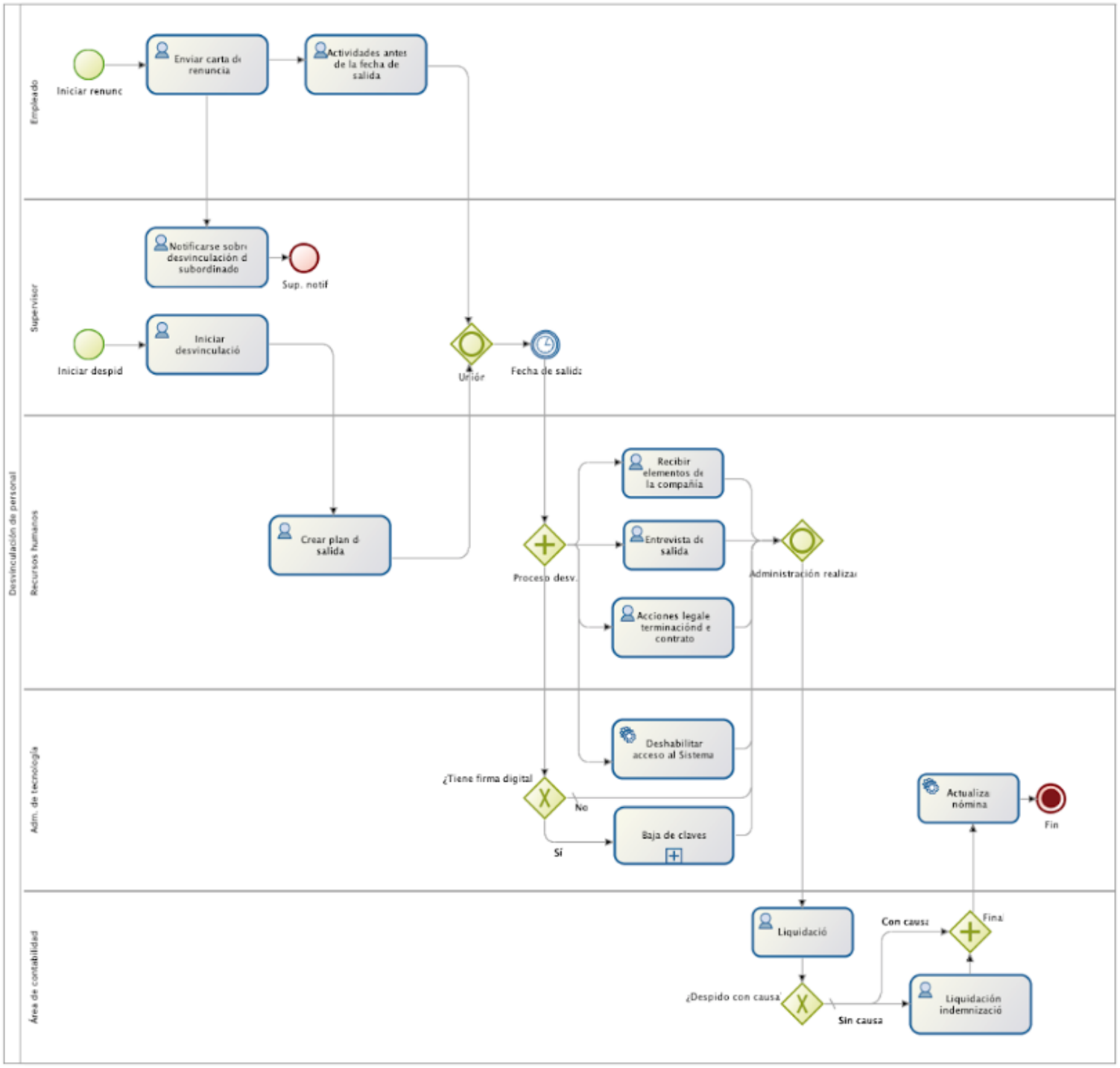

Figura 4.5: Proceso de Desvinculación de personal

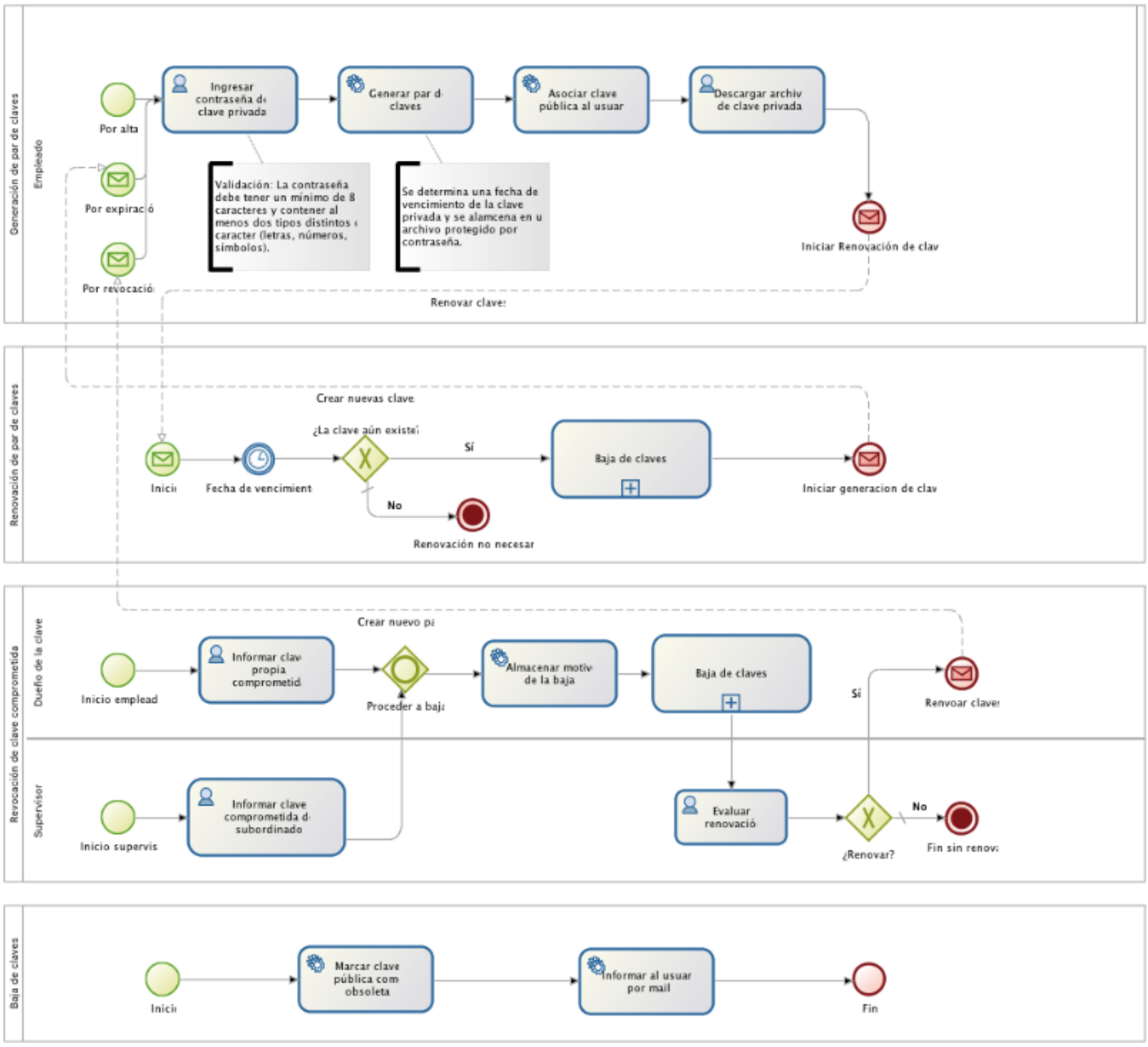

Figura 4.6: Procesos de Gestión de pares de claves

Capítulo 5

Implementación de procesos de prueba con el esquema propuesto

En este capítulo se explicará el trabajo realizado para la implementación de una prueba de concepto del esquema propuesto en el capítulo 4, en un BPMS concreto.

Como BPMS se evaluó la posibilidad de utilizar BizAgi, Process Maker y Bonita BPM.

Bizagi es una suite ofimática con dos productos complementarios, un Modelador de Procesos y una Suite de BPM. Bizagi Process Modeler es un Freeware utilizado para diagramar y simular procesos en notación BPMN. Bizagi BPM Suite es una solución para poner en marcha la implementación de un workflow de BPM completo. Bizagi es complejo y no ofrece todas sus funciones de forma gratuita, y tampoco es de código abierto.

Process Maker es una herramienta web para el modelado y ejecución de procesos de negocio muy simple, orientada a analistas de negocio para permitirles modelar sus flujos de trabajo basados en aprobaciones. No tiene las capacidades de Bizagi o Bonita ni tampoco ofrece buenos mecanismos de integración. Si bien dispone de un período de prueba de 30 días, no es gratuito.

La alternativa elegida es Bonita BPM, por ser una solución de código abierto, un proyecto con mucha actividad y constantes mejoras, y con una comunidad muy presente y activa. Tiene una muy buena relación entre complejidad y funcionalidad, y se ajusta muy bien a los estándares de BPM. Además ofrece varios mecanismos de integración con otros sistemas y herramientas, lo cual lo hace muy atractivo para el caso de estudio que se plantea.

5.1. Bonita BPM

Bonita BPM contiene una plataforma compuesta del BPM Studio, UI Designer, Bonita BPM Portal, Bonita BPM Engine, un servidor Tomcat, y una base de datos h2. Estas herramientas en conjunto permiten diseñar, configurar, ejecutar y probar procesos de negocio.

La herramienta web UI Designer permite crear formularios web para las tareas humanas, widgets personalizados para los mismos y aplicaciones web básicas para la interacción con los procesos. Los formularios se construyen con poca programación, gracias a una interfaz web visual para arrastrar y soltar componentes, agregar assets externos y configurar propiedades, clases css, validaciones y eventos. Los widgets personalizados se construyen desde una interfaz web similar donde ser requiere más programación, permitiendo implementar el código de un template y un controller de un componente de Angular para

crear un nuevo widget. La herramienta para aplicaciones web permite crear páginas web que tenga interacción con los procesos, por ejemplo para iniciarlos o para acceder a los datos de los mismos, ya que no toda la interacción con un proceso se realiza necesariamente desde el portal.

La aplicación Bonita BPM Portal se utiliza para ver y completar las tareas de todos los procesos. Es el portal web visible para los usuarios finales y desde allí ellos pueden ver sus tareas y realizar acciones gracias a una lista genérica de todas las tareas de los procesos en los que cada usuario está involucrado.

El motor de procesos -Bonita BPM Engine- ejecuta los procesos y mantiene el estado de todas las instancias en ejecución y finalizadas. Además provee soporte para las interacciones con otros sistemas y las lleva a cabo.

El servidor Tomcat es la plataforma sobre la cual se ejecutan las aplicaciones web mencionadas, y la base de datos H2 es por defecto el destino de los datos a almacenar.

Por último, BPM Studio es el entorno de desarrollo basado en Eclipse que permite diagramar procesos, implementarlos, configurarlos y probarlos. Además permite la administración de la estructura de la organización vinculada a los procesos y la edición de un modelo de datos que se verá reflejado en la base de datos. [5]

5.2. Configuración del entorno

El primer paso para la implementación de un proceso en Bonita es la configuración del entorno de desarrollo. El objetivo es crear un proceso lo suficientemente completo como para llevar las ideas planteadas en el capítulo anterior a un ejemplo concreto.

Para esto es necesario especificar algunos datos de prueba, como usuarios y roles, y crear un modelo de datos. Éstos no son un reflejo exacto de la realidad, sino que son una versión simplificada adecuada a nuestro ejemplo.

En las siguientes secciones explicaremos la configuración del modelo de la organización y el modelo de datos de negocio.

5.2.1. Organización

Con fines prácticos realizaremos la creación de algunos usuarios y grupos. Los grupos son sectores de la organización y tienen una relación estrecha con los actores que se definirán posteriormente en los procesos, en notación BPMN.

Todo actor de un “lane” o “senda” en un diagrama de Bonita, debe estar asociado a un filtro de actor, cuya implementación más común es la selección de un grupo. Esto quiere decir que si un actor de un proceso llamado “Personal de Finanzas” está vinculado al grupo “Finanzas” de nuestra organización de ejemplo, sólo los usuarios que pertenezcan a ese grupo podrán ver las tareas de ese actor disponibles para tomar en su lista de tareas en Bonita BPM Portal.

Para crear grupos y usuarios en Bonita Studio, debemos acceder al Menú Organización → Administrar. En la siguiente figura podemos ver los grupos que hemos definido:

Cada grupo se puede editar desde un diálogo de administración de la organización, y podemos ver en forma de árbol la relación jerárquica entre ellos. La interfaz de la siguiente figura nos permite añadir grupos y subgrupos, eliminarlos, editar su nombre, título, ruta y descripción.

Tabla 5.1: Grupos de nuestro conjunto de datos de prueba

/globecare	GlobeCare	Este grupo representa el departamento de GlobeCare de la organización GlobeCare
/globecare/hr	Recursos Humanos	Este grupo representa el departamento de recursos humanos de la organización.
/globecare/finance	Finanzas	Este grupo representa el departamento de contabilidad de la organización.
/globecare/it	Infraestructura	Este grupo representa el departamento de infraestructura de la organización.
/globecare/marketing	Marketing	Este grupo representa el departamento de marketing de la organización.
/globecare/production	Producción	Este grupo representa el área de producción de la organización.
/globecare/research	Investigación y desarrollo	Este grupo se encarga de la investigación y desarrollo para la mejora continua de la organización.
/globecare/services	Servicio	Este grupo provee servicio a nuestros clientes, directa o indirectamente, para llevar a cabo la misión de la organización.
/globecare/services/agents	Agentes	Este grupo atiende los llamados de los clientes.
/globecare/services/coordinators	Coordinadores de asistencia	Este grupo se encarga de coordinar las asistencias médicas.
/globecare/services/supervisors	Supervisores	Este grupo realiza la supervisión de los casos de asistencia coordinados, evalúa casos que requieran toma de decisiones y realiza monitoreos y auditorías.
/globecare/services/specialists	Personal especialista	Este grupo aporta la visión de especialistas para casos complejos que requieran conocimiento específico para poder ser evaluados.
/globecare/services/auditors	Auditores	Este grupo se dedica a realizar auditorías de ciertas actividades de la organización.

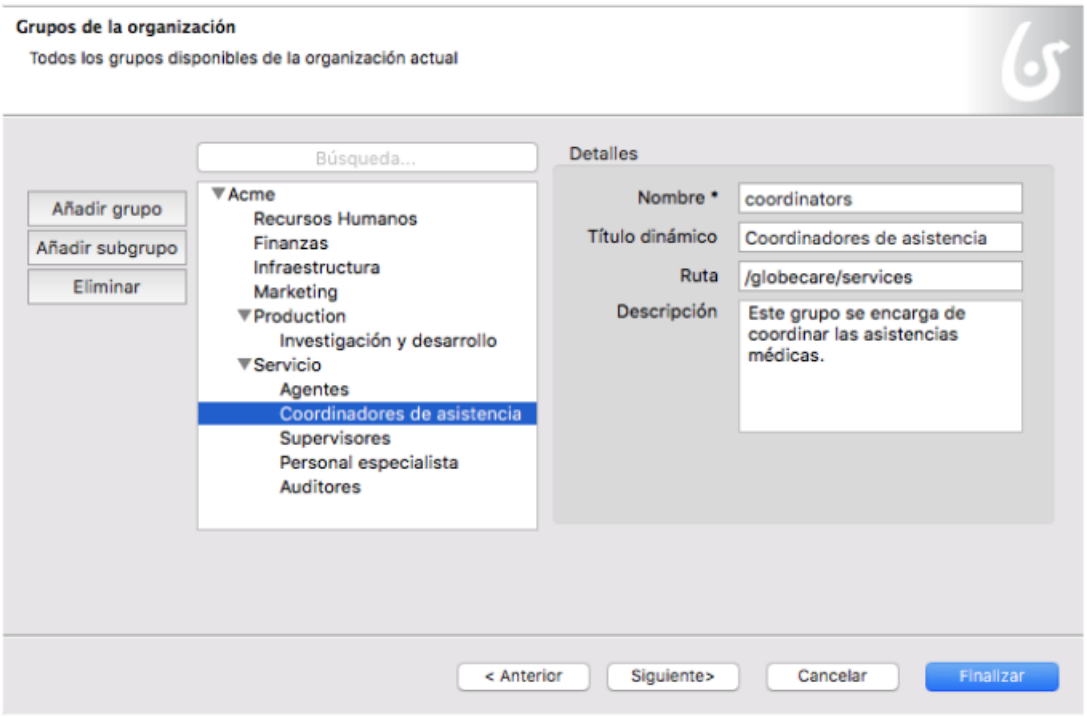

Figura 5.1: Edición de grupos de la organización

La edición de la organización continúa como un wizard, donde el siguiente paso es la creación o edición de roles. Para nuestro ejemplo no será necesario ningún otro rol adicional al que está disponible por defecto: "Member". Con la jerarquía de grupos es más que suficiente para agrupar a nuestros usuarios y distinguir sus responsabilidades en los distintos procesos.

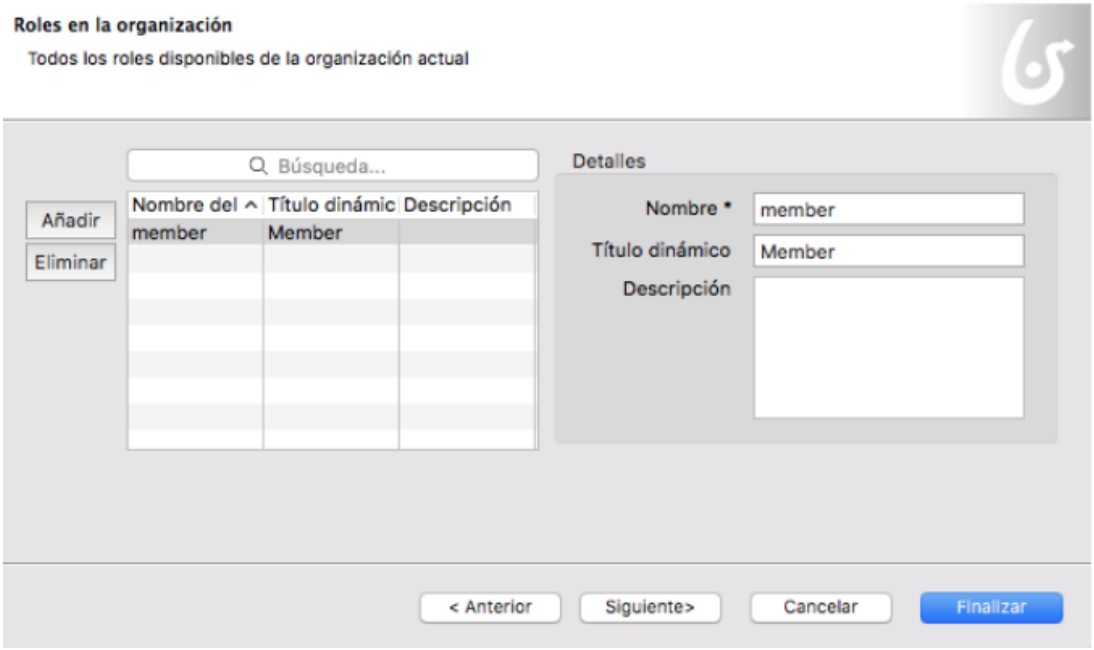

Figura 5.2: Edición de roles de la organización

El siguiente paso es la creación y edición de usuarios. Bonita Studio provee usuarios precargados que podemos utilizar como base para crear nuestra organización, para tener datos de prueba que nos eviten tener que hacer una carga inicial de estos datos antes de poder empezar a ejecutar procesos.

Sin embargo, con el objetivo de obtener una implementación de nuestro ejemplo que sea relativamente fiel a la realidad, y facilitar la realización de pruebas, crearemos algunos usuarios de nuestra organización prestadora de asistencia médica y los asignaremos a algunos de los grupos creados anteriormente.

La siguiente tabla muestra los usuarios creados, con nombres que tienen siempre un sufijo para no confundir sus roles durante nuestras pruebas.

Tabla 5.2: Usuarios de ejemplo de la organización y grupos asignados

Usuario	Nombre	Apellido	Miembro de grupo
juan.a	Juan	Agente	/globecare/services/agents
pablo.c	Pablo	Coordinador	/globecare/services/coordinators
sabrina.s	Sabrina	Supervisora	/globecare/services/supervisors
florencia.sp	Florencia	Especialista	/globecare/services/specialists

Para cada usuario, la interfaz nos permite configurar un nombre de usuario, contraseña, responsable (superior), nombre, apellido, grupo y rol, entre otros atributos. Los mencionados son los que tienen relevancia para nuestra implementación, y se pueden ver en las figuras siguientes.

Una vez configurados los grupos, roles y usuarios, podemos finalizar el asistente y nuestra organización se publicará, es decir, se crearán los datos necesarios en las tablas de la base de datos H2 utilizada por

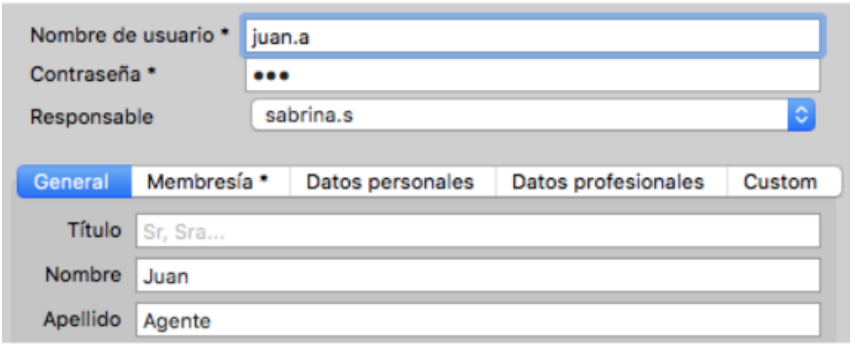

Figura 5.3: Edición de atributos generales de los usuarios

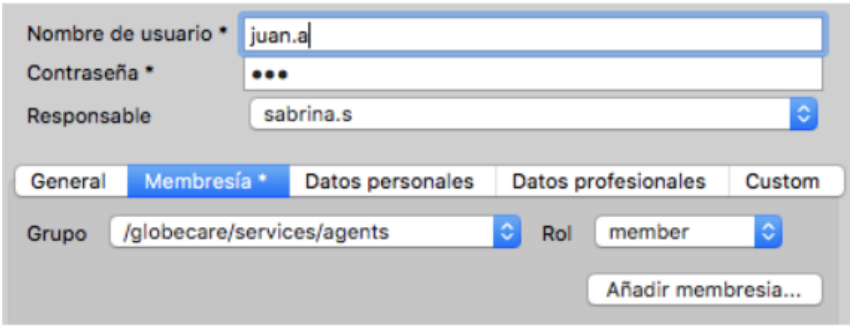

Figura 5.4: Edición de membresía de grupo y rol

Bonita BPM Platform.

Estos datos de prueba nos permitirán configurar nuestros procesos con actores asociados a cada tarea y a cada grupo de la organización, y los usuarios de prueba nos permitirán realizar pruebas desde Bonita BPM Portal, llevando a cabo cada una de las tareas de los procesos con un usuario que pertenezca al grupo correspondiente.

5.2.2. Modelo de datos de negocio

La configuración del modelo de datos de negocio es lo que nos permitirá definir los tipos de datos necesarios para nuestros procesos y las relaciones entre ellos. Esta configuración se plasma en clases Java a nivel implementación, y también en un mapeo en la base de datos H2 que provee Bonita, pero a la hora de configurarlo el ambiente provee un wizard que nos abstrae de tener que escribir manualmente las clases Java o utilizar el lenguaje SQL.

Accediendo al menú "Desarrollo" → "Modelo de datos de negocio" → "Administrar", podremos configurar una lista de "Objetos de Negocio", cada uno de los cuales tendrá una descripción, atributos, restricciones de valores únicos, consultas e índices. Los atributos podrán ser de un tipo básico o pueden ser una instancia de otro objeto de negocio. Cada atributo puede ser múltiple y/o requerido. Las consultas son métodos de los Objetos de negocio que permiten acceder de forma simple a consultas a la base de datos. Los índices nos permiten determinar qué atributos serán mapeados a la base de datos como un índice de la tabla, además de en una columna de la misma.

Bonita provee los siguientes tipos básicos: boolean, date, double, float, integer, long, string, text. Y las siguientes consultas predeterminadas: *find()*, *countForFind()*, *findByPersistenceId(id)*, *findBy[atributo](valor)*,

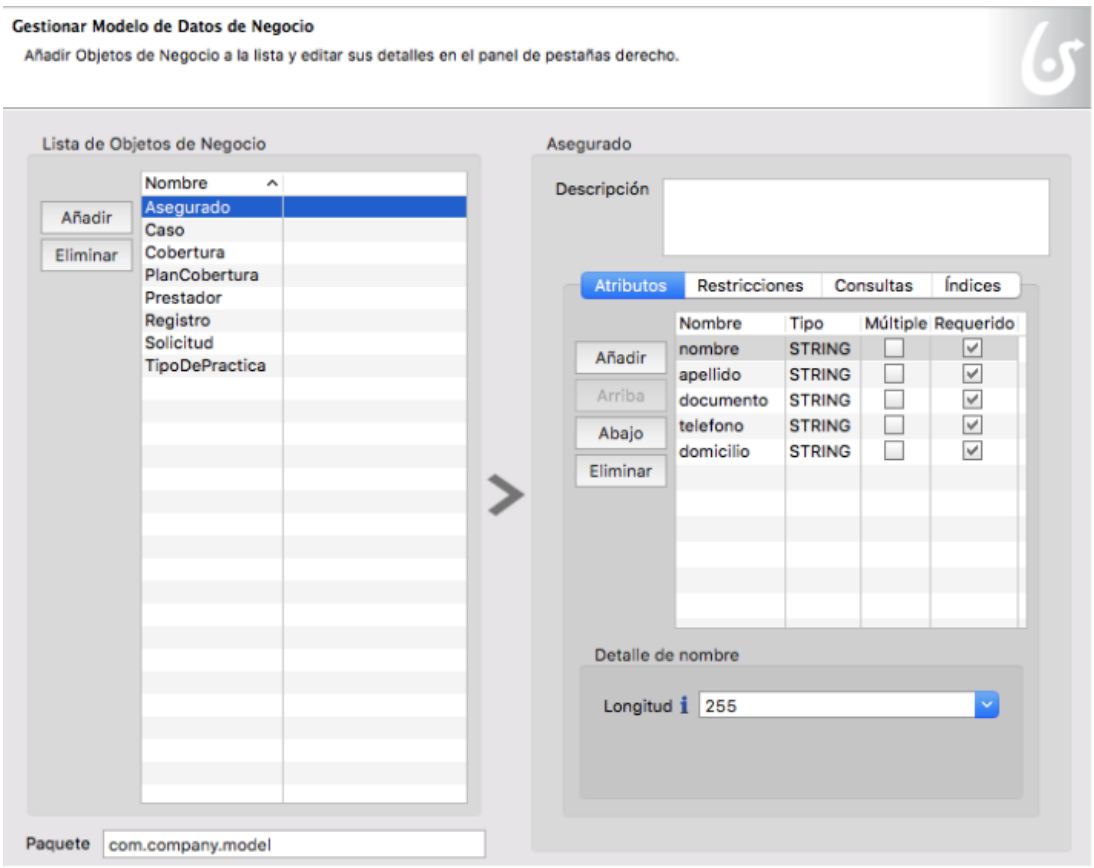

Figura 5.5: Wizard de Bonita Studio para la edición del modelo de datos de negocio

countForFind[atributo](valor). Existe la posibilidad de agregar consultas personalizadas en lugar de utilizar las predeterminadas provistas por Bonita.

Para nuestro caso de estudio, no será necesario configurar restricciones, consultas o índices. Bastará con crear los Objetos con sus atributos y relacionar algunos objetos entre ellos. Algunos objetos de negocio son simples, es decir que se componen en su totalidad de atributos de tipos de datos simples. En nuestro ejemplo son TipoDePractica, Prestador, PlanCobertura y Asegurado.

Otros objetos de negocio por otro lado serán compuestos, ya que a su vez se relacionan con otros objetos de negocio ya sea por composición o por agregación. La relación de agregación implica que el objeto relacionado tiene su propio ciclo de vida y no puede modificarse cuando el objeto padre se guarda en la base de datos. En cambio en la relación de composición, el objeto relacionado es parte del objeto padre. Hay una opción que nos permite determinar qué política se utilizará para cargar objetos relacionados al acceder a la base de datos: “Siempre cargar objetos relacionados” o “Sólo cuando sea necesario”. La última opción se recomienda para objetos que rara vez se muestren o se modifiquen desde el padre.

En nuestro ejemplo, los objetos de negocio compuestos son:

- Registro, que se relaciona con Prestador en su atributo “prestador”.
- Cobertura, que se relaciona con PlanCobertura y Asegurado en sus atributos requeridos “planCobertura” y “asegurado”.
- Caso, que se relaciona con Cobertura y Registro, en su atributo requerido “cobertura” y us atributo múltiple “registros”.

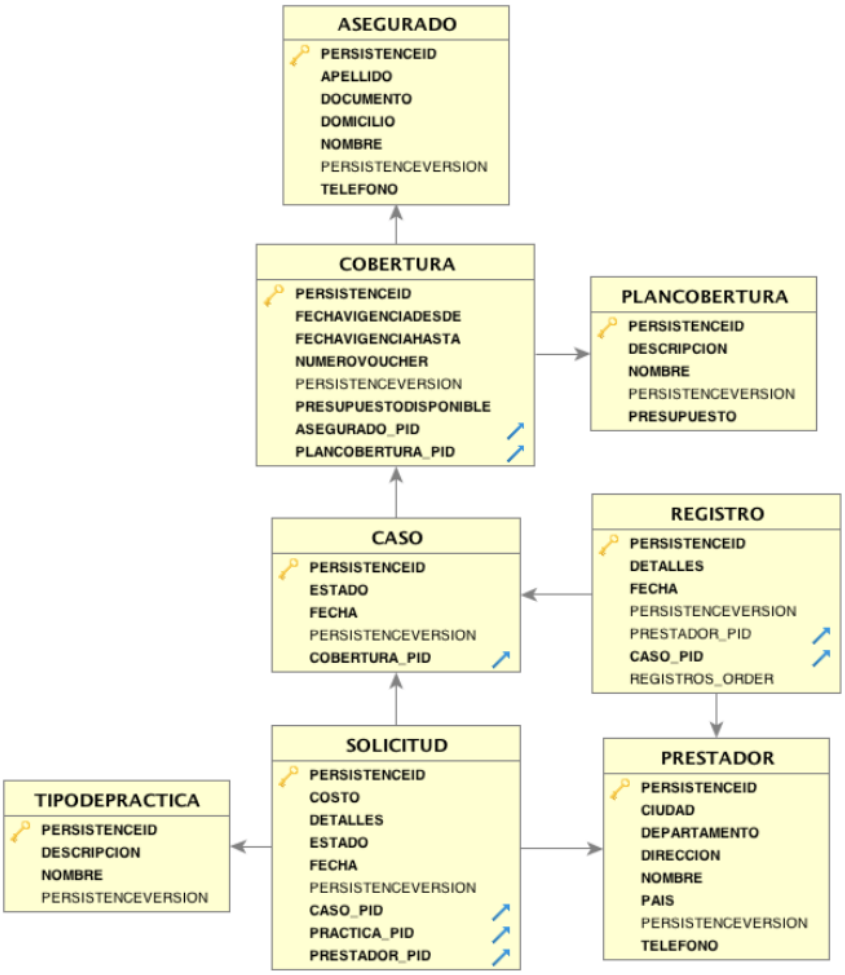

Figura 5.6: Modelo de datos de negocio configurado

- Solicitud, que se relaciona con TipoDePractica, Prestador y Caso, en sus atributos requeridos “practica”, “prestador” y “caso”.

El estado de una solicitud puede ser “Pendiente”, “Preaprobada”, “Aprobada” o “Rechazada”. El estado de un caso puede ser “Abierto” o “Cerrado”.

5.2.3. Datos de prueba

Con fines prácticos, con el objetivo de facilitar la ejecución de los procesos implementados, se creó un proceso llamado “Alta Datos de Prueba”, que al ser ejecutado carga varios datos en la base de datos y luego muestra un formulario que permite al usuario corroborar la correcta generación de los datos.

El proceso genera datos de prueba para que el modelo de datos y la base de datos de Bonita no estén vacíos a la hora de ejecutar los procesos centrales de este proyecto. Crea y almacena datos de los siguientes tipos: TipoDePractica, PlanCobertura, Prestador, Asegurado y Cobertura.

Los datos se instancian como variables de negocio de Bonita, de forma que al ejecutar el proceso, sean almacenados en la base de datos. A continuación se muestran tres ejemplos de instanciación de datos.

```
def oro = new com.company.model.PlanCobertura()
oro.nombre = "Oro"
oro.descripcion = "Plan Oro, cubre hasta 40mil dolares de gastos medicos."
oro.presupuesto = 40000
return neorowItem
```

Tabla 5.3: Datos de prueba generados en "Alta Datos de Prueba"

Variable	Tipo	Descripción
rayosX	TipoDePractica	Tipo de práctica "Rayos X"
ecografia	TipoDePractica	Tipo de práctica "Ecografía"
cirugiaApendicitis	TipoDePractica	Tipo de práctica "Cirugía Apendicitis"
bronce	PlanCobertura	Plan de Cobertura "Bronce"
oro	PlanCobertura	Plan de Cobertura "Oro"
platino	PlanCobertura	Plan de Cobertura "Platino"
cruzRoja	Prestador	Prestador "Hospital Cruz Roja", Madrid
universitario	Prestador	Prestador "Hospital Universitario", Madrid
juanPeralta	Asegurado	Asegurado "Juan Peralta"
coberturaEjemplo	Cobertura	Cobertura de Juan Peralta
coberturaVieja	Cobertura	Cobertura anterior ya vencida de Juan Peralta

```
def juanPeralta = new com.company.model.Asegurado()
juanPeralta.nombre = "Juan"
juanPeralta.apellido = "Peralta"
juanPeralta.documento = "35888999"
juanPeralta.domicilio = "Calle 15 N 37 entre 32 y 33, 1900"
juanPeralta.telefono = "+542215876534"
return juanPeralta
```

```
def desde = Calendar.getInstance(); desde.set(2018, Calendar.MAY, 20)
def hasta = Calendar.getInstance(); hasta.set(2018, Calendar.AUGUST, 31)
def numeroVoucher = "2222"
def plan = planCoberturaDAO.findByNombre("Oro", 0, 1).get(0)
def asegurado = aseguradoDAO.findByApellido("Peralta", 0, 1).get(0)
def coberturaEjemplo = new com.company.model.Cobertura()
coberturaEjemplo.asegurado = asegurado
coberturaEjemplo.fechaVigenciaDesde = desde.getTime()
coberturaEjemplo.fechaVigenciaHasta = hasta.getTime()
coberturaEjemplo.numeroVoucher = numeroVoucher
coberturaEjemplo.planCobertura = plan
coberturaEjemplo.presupuestoDisponible = plan.presupuesto
return coberturaEjemplo
```

El proceso de Alta Datos de Prueba finaliza con una tarea con interacción humana que muestra un formulario que permite visualizar los datos recién generados.

Figura 5.7: Formulario de comprobación de datos de prueba

Estos datos permiten la ejecución de los procesos "Gestión de caso de asistencia médica" y "Solicitud de práctica" sin necesidad de implementar y ejecutar los demás procesos descritos en el capítulo 4.

Cada vez que se realizaba un cambio en el modelo de datos o se necesitaba probar todo de cero durante el desarrollo de este proyecto, se procedía a borrar el modelo de datos, volverlo a generar y ejecutar este proceso para cargar los datos de prueba sobre la base de datos recién generada.

5.3. Proceso de "Gestión de caso de asistencia médica"

Este es el primero de los dos procesos implementados para el ejemplo en cuestión: "Gestión de caso de asistencia médica". A continuación describiremos el trabajo realizado, desde el diseño del proceso y su configuración, hasta algunos detalles de implementación. El diseño BPMN del proceso puede apreciarse en la Figura 4.1. Los roles involucrados y los flujos del proceso fueron descritos en la sección 4.1. Debemos describir la configuración de actores y lanes o sendas, la definición de variables de negocio y su inicialización, el contrato del proceso y su formulario de instanciación y la configuración de tareas y compuertas.

5.3.1. Actores

En Bonita BPM, los actores se definen a varios niveles. Primero, a nivel proceso, se definen todos los actores involucrados, y cuál de ellos será el "Iniciador". Los actores son simplemente nombres que describen un rol participante.

En el caso de este proceso, tendremos 2 actores: "Personal de atención al cliente" y "Personal de gestión del caso". Dichos actores serán asociados a un "lane" o senda, en este caso, las sendas homónimas.

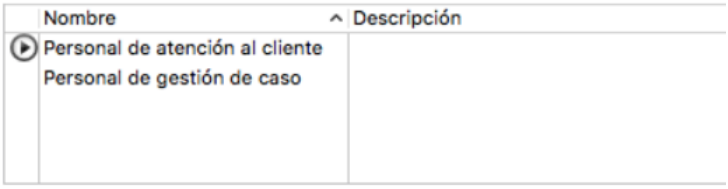

Figura 5.8: Definición de actores en Bonita BPM

Los actores deben ser asociados a un grupo, rol o usuario. Esta configuración se efectúa en Bonita BPM a través del menú "Servidor" → "Configurar" → "Mapeo de actores". En este cuadro de diálogo podemos vincular cada actor definido en el proceso a un determinado grupo. En el caso de nuestro proceso, el mapeo se define de la siguiente manera:

Tabla 5.4: Actores de Gestión de caso de asistencia médica

Actor	**Grupo**
Personal de atención al cliente	/globecare/services/agents
Personal de gestión de caso	/globecare/services/coordinators

Con esto queda definido el mapeo de actores para el proceso y por lo tanto las tareas de la senda "Personal de atención al cliente", vinculadas al actor homónimo, podrán ser tomadas para su realización sólo por usuarios pertenecientes al grupo `/globecare/services/agents`. Y las tareas del lane o senda

"Personal de gestión de caso", vinculadas al actor homónimo, podrán ser tomadas sólo por usuarios pertenecientes al grupo `/globecare/services/coordinators`.

5.3.2. Contrato y formulario de instanciación

La noción de contrato en Bonita BPM está disponible en dos niveles: a nivel instanciación de proceso, y a nivel ejecución de tareas humanas. Un contrato está compuesto de entradas y restricciones. Las entradas son piezas de información que deben ser provistas al proceso o a la tarea humana. Las restricciones son aplicadas a las entradas para corroborar que los valores de las entradas son válidos.

Un contrato especifica las piezas de información que el proceso requiere para iniciar. Sin esta información el proceso no puede ejecutar su lógica de negocio, y carece de sentido intentar iniciar una instancia del mismo. Es decir, si se intenta iniciar una instancia de un proceso sin proveer las entradas esperadas, el sistema se negará a iniciar el proceso. Lo mismo sucede con las tareas humanas: si se intenta realizar la tarea sin proveer las entradas esperadas, el sistema se negará a ejecutar la tarea. Un contrato protege el proceso de interacciones externas incorrectas. Luego de la validación, las entradas son utilizadas para inicializar los datos de negocio y la información es utilizada en la ejecución del proceso.

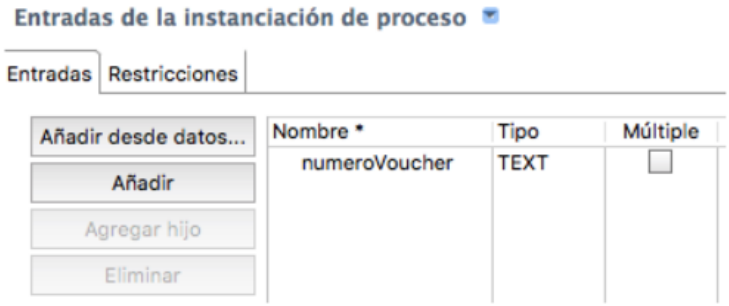

Figura 5.9: Definición de contrato de proceso en Bonita BPM

El contrato de este proceso tendrá como único ítem `numeroVoucher`, que es el número identificador del voucher asociado a la cobertura del asegurado.

Formulario de instanciación

En Bonita BPM, un formulario es una página que pertenece a un proceso. Puede ser un formulario de instanciación, un formulario de una tarea humana, o un formulario de revisión. El formulario de instanciación se utiliza para iniciar manualmente un proceso. Los datos necesarios para el contrato del proceso se proveen a través de este formulario.

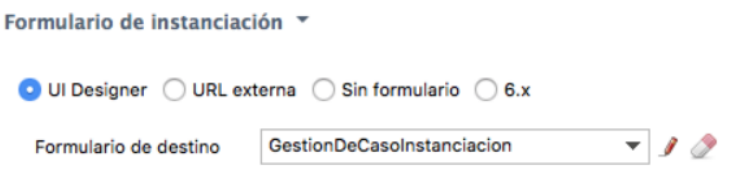

Figura 5.10: Configuración de formulario de instanciación en Bonita BPM

En este proceso se solicita al usuario el número de voucher asociado a la cobertura del asegurado. Para evitar que el proceso se pueda iniciar con un número de voucher no válido, se utiliza un widget de tipo autocompletar.

Figura 5.11: Formulario de instanciación del proceso "Gestión de caso de asistencia"

El widget de tipo autocompletar permite ingresar caracteres y sugiere una lista de valores de entre las opciones disponibles, que coincidan total o parcialmente con los caracteres ingresados. Asimismo, el valor del widget no se considera válido hasta no haber seleccionado un elemento de las sugerencias del widget de autocompletado.

Las opciones disponibles para el valor de la entrada `numeroVoucher` provienen de las coberturas precargadas, descritas en la sección 5.2.3. Para acceder a ellas desde el formulario, se crea una variable `varCoberturas`, de tipo `External API`, y se accede al modelo de datos de negocio para obtener todas las coberturas almacenadas en el sistema.

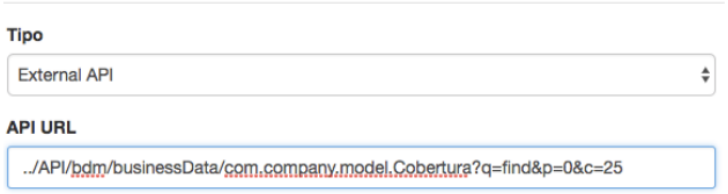

Figura 5.12: Configuración de variable `varCoberturas` de tipo `External API`

En Bonita BPM, los formularios puede utilizar requests AJAX a la API para obtener datos del modelo de datos. Por ejemplo en este caso:

`../API/bdm/businessData/com.company.model.Cobertura?q=find&p=0&c=25`

Luego, dada la variable `varCoberturas` del formulario, procedemos a la configuración del widget de tipo Autocompletar. Este widget requiere algunos atributos, entre ellos la Etiqueta y su posición, los valores disponibles, la clave mostrada y el valor.

Figura 5.13: Configuración del Widget Autocomplete "Número de voucher"

Los valores disponibles serán asociados a la variable `varCoberturas`, es decir la lista completa de coberturas almacenadas en el sistema conformará la lista de valores disponibles para ingresar en el Autocomplete. Y la "Clave mostrada" es el atributo de cada ítem de la lista de Variables disponibles que será visualizado, en este caso el atributo `numeroVoucher`. El valor indica a qué variable se vincula el valor del widget, en otras palabras: una vez elegido un valor a través del widget, a qué variable será asignado dicho valor.

Por último, la variable `formOutput`, que debe corresponderse con el formato del contrato esperado por el proceso, se define de la siguiente forma:

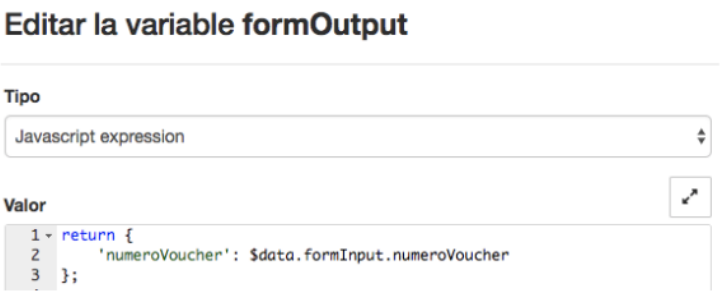

Figura 5.14: Variable formOutput de tipo `Javascript expression`

El valor del atributo `numeroVoucher` del input del formulario, se asigna al atributo `numeroVoucher` de la salida del formulario de instanciación, que será la entrada del contrato del proceso.

5.3.3. Variables de negocio

El siguiente paso es la configuración de variables de negocio. Las mismas se utilizan para almacenar información con una o más de las siguientes características:

- Es utilizada principalmente por Bonita BPM
- Tiene un significado luego de que la instancia de un proceso es archivada
- Es usada varias en varios lugares del proceso
- Es leída por un proceso para orientar su flujo

La configuración de las mismas es a nivel proceso, y consiste en definir un nombre, un tipo y una inicialización para cada una.

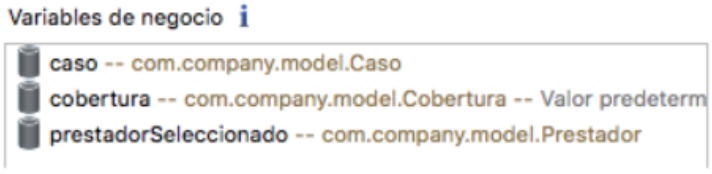

Figura 5.15: Configuración de variables de negocio

En el caso de este proceso, se definen tres variables de negocio:

Algunas variables deben ser inicializadas al instanciar el proceso. Para ello se puede recurrir a un DAO que provee Bonita para todos los modelos definidos en el Modelo de Datos de Negocio, cuya explicación fue desarrollada en la sección 5.2.2. La variable cobertura será inicializada con un script que haga uso de `coberturaDAO` y del ítem `numeroVoucher` del contrato, de la siguiente forma:

Tabla 5.5: Variables del proceso Gestión de caso de asistencia médica

Variable de negocio	Tipo	Contenido
caso	model.Caso	El caso a crear
cobertura	model.Cobertura	La cobertura del asegurado
prestadorSeleccionado	model.Prestador	Prestador al que se deriva el caso

```
def coberturasResult = coberturaDAO.findByNumeroVoucher(numeroVoucher, 0, 0)

if (coberturasResult.isEmpty()){
  return null
} else {
  return coberturasResult.get(0)
}
```

Las variables `caso` y `prestadorSeleccionado`, tomarán un valor luego de la instanciación del proceso, conforme avance el flujo del mismo y se vayan completando las tareas.

5.3.4. Configuración de tareas y compuertas

En esta sección explicaremos la implementación del proceso restante: cada compuerta y tarea del diseño debe configurarse, las compuertas con una lógica de decisión y las tareas con un tipo de tarea. Las tareas humanas tienen formularios, un contrato y pueden tener operaciones post-terea. Las tareas de servicio ejecutan alguna operación o utilizan algún conector para interacciones con elementos externos al proceso. En nuestro caso el proceso está compuesto únicamente por tareas humanas y compuertas.

Compuerta "Voucher válido?"

Esta compuerta determina si el voucher es válido comparando la fecha de finalización de la vigencia de la cobertura con la fecha actual. Estas condiciones se definen como scripts o comparaciones simples de datos, en las flechas que conectan la compuerta con cada una de los siguientes elementos posibles en el flujo. Cuando hay sólo dos elementos posibles, uno de ellos define una condición, y el otro se marca como flujo por defecto, es decir sólo se lo tomará cuando las demás condiciones evalúen todas a falso.

```
return cobertura != null &&
   !Calendar.getInstance().getTime()
        .after(cobertura.fechaVigenciaHasta)
```

La implementación es muy sencilla. Dado que la variable cobertura fue inicializada a través del contrato y el formulario de instanciación del proceso, está disponible en este punto. Se accede al atributo `fechaVigenciaHasta` de la misma y se lo compara con la fecha actual

Tarea "Informar voucher invalido"

Es una tarea humana sin contrato que muestra un formulario de sólo lectura solicitando a un actor "Personal de atención al cliente" que informe al cliente que el voucher es inválido. Es una tarea humana sin contrato y sin operaciones, y el formulario es extremadamente sencillo: contiene el título de la tarea un breve texto explicando el estado del proceso.

Tarea "Cargar datos del caso"

Es una tarea humana que permite a un actor "Personal de atención al cliente" ingresar los detalles del caso puntual que se está registrando.

El contrato está compuesto de dos elementos: un campo de tipo TEXT llamado `detalles` y un campo de tipo DATE un input para la fecha y un input de texto libre para ingresar los detalles del caso. El contrato tiene una única validación: la fecha no puede ser posterior al día en que se está creando el caso.

Hay una operación post-tarea, que consiste en inicializar la variable de negocio `caso`. Esto se lleva a cabo a través de un script que retorna el valor a asignar a la variable.

```
def registro = new Registro()
def calendar = Calendar.getInstance()
registro.fecha = calendar.getTime()
registro.detalles = detalles

def newItem = new Caso();
newItem.estado = "Abierto"
newItem.fecha = fecha
newItem.cobertura = cobertura
newItem.addToRegistros(registro);

return newItem;
```

El script crea un nuevo registro, le asigna la fecha actual y los detalles recibidos en el contrato. Luego crea un objeto de tipo Caso y establece su estado como "Abierto", su fecha como la fecha recibida en el contrato, la cobertura como la cobertura del proceso actual y los registros como una lista cuyo único ítem es el registro creado anteriormente. Por último retorna el caso recién creado.

Tarea "Derivar a centro médico"

Tarea humana que permite a un actor "Personal de atención al cliente" seleccionar un prestador al cual derivar el caso.

El contrato de la tarea tiene un único atributo de tipo `INTEGER` llamado `id_prestador`. El formulario muestra los datos del caso y los registros del mismo, y un input de tipo lista desplegable que permite elegir un prestador. El id del prestador es enviado como salida del formulario. Como operaciones post-tarea, se inicializa la variable `prestadorSeleccionado` Inicializar la variable de negocio `prestadorSeleccionado` utilizando el DAO del Objeto de Negocio Prestador para recuperar el prestador según el id recibido en el contrato.

```
return prestadorDAO.findByPersistenceId(id_prestador);
```

Como segunda operación post-tarea, se agrega a los registros del caso un nuevo ítem informando a qué prestador fue derivado.

```
def newItem = new Registro()
newItem.detalles = "Derivado a prestador "+prestadorSeleccionado.nombre + ", "+
prestadorSeleccionado.ciudad + ", " + prestadorSeleccionado.pais + ". "
newItem.fecha = Calendar.getInstance().getTime()
newItem.prestador = prestadorDAO.findByPersistenceId(id_prestador)
return newItem
```

Este script retorna como valor el nuevo Registro para agregarlo a la lista de registros de la variable de negocio `caso`. Los detalles de este registro se construyen automáticamente en base a un texto predefinido

y al nombre del prestador, la ciudad y el país del mismo. La fecha del registro se establece con la fecha actual en ese momento. El prestador se obtiene nuevamente utilizando el DAO del Objeto de Negocio Prestador, a partir del `id_prestador` recibido como entrada en el contrato.

Tarea "Enviar garantía a centro médico"

Tarea que solicita a un actor "Personal de gestión de caso" que envíe la garantía al centro médico que figura asignado al caso. Esta tarea no tiene contrato y no tiene operaciones. Es una tarea operativa manual cuyo formulario es una mera interacción con el usuario para que visualice la tarea y pueda marcarla como completada cuando haya finalizado la misma.

5.4. Proceso de "Solicitud de prácticas"

Este es el segundo de los dos procesos implementados para el ejemplo en cuestión: **Solicitud de prácticas**. A continuación describiremos el trabajo realizado, desde el diseño del proceso y su configuración, hasta algunos detalles de implementación. Este proceso a su vez contiene elementos claves de la integración con Firma Digital, en dos de sus tareas. Dicha integración será explicada superficialmente en esta sección, y el tema será abordado en mayor profundidad en la secciones posteriores.

El diseño BPMN del proceso puede apreciarse en la Figura 4.2. Los roles involucrados y los flujos del proceso fueron descritos en la sección 4.1. Al igual que en la sección anterior, procederemos a describir la configuración de actores y lanes o sendas, la definición de variables de negocio y su inicialización, el contrato del proceso y su formulario de instanciación y la configuración de tareas y compuertas.

5.4.1. Actores

Se configuran los roles definidos en el capítulo 4, que estarán asociados a cada uno de los lanes o sendas del proceso.

Tabla 5.6: Roles de Solicitud de prácticas

Actor	**Grupo**
Personal de gestión de caso	/globecare/services/coordinators
Personal de supervisión	/globecare/services/supervisors
Personal médico especialista	/globecare/services/specialists

Con esto queda definido el mapeo de actores para el proceso y por lo tanto las tareas de la senda "Gestión de caso", vinculadas al actor "Personal de gestión de caso", podrán ser tomadas para su realización sólo por usuarios pertenecientes al grupo `/globecare/services/coordinators`. Las tareas del lane o senda "Supervisor", vinculadas al actor "Personal de supervisión", podrán ser tomadas sólo por usuarios pertenecientes al grupo `/globecare/services/supervisors`. Y las tareas del lane o senda "Médico especialista", vinculadas al actor "Personal médico especialista", podrán ser tomadas sólo por usuarios pertenecientes al grupo `/globecare/services/specialists`

5.4.2. Contrato y formulario de instanciación

El contrato de este proceso tendrá como único ítem `solicitudInput`, que será de un tipo de dato `COMPLEX`, y que a su vez deberá tener los siguientes atributos:

Tabla 5.7: Atributos de `solicitudInput`

Atributo	**Descripción**
`id_practica`	Identificador de la práctica que se solicita
`detalles`	Detalles respecto a la solicitud
`fecha`	Fecha de la solicitud
`id_prestador`	Identificador del prestador al que se realiza la solicitud
`id_caso`	Identificador del caso (generado a través del proceso de la sección 5.3)
costo	Costo en dólares estadounidenses de la práctica solicitada

Este contrato no tendrá restricciones, ya que el formulario aplica varias validaciones que garantizan la validez de los datos recibidos como entrada. La forma de definir el contrato es ligeramente diferente a la del proceso anterior, dado que en este caso tenemos un único ítem que a su vez tiene sub-ítems. El formulario deberá devolver un objeto JSON en este punto en su variable de salida, cuyos atributos deberán corresponderse con cada uno de estos sub-ítems mencionados.

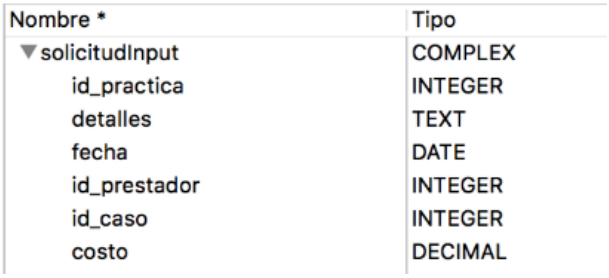

Figura 5.16: Definición de tipo de dato complejo en un contrato

Formulario de instanciación

El formulario de instanciación de este proceso solicita al usuario iniciador un el conjunto de datos definidos en el contrato de la siguiente manera: un campo `DatePicker` para seleccionar la fecha, tres campos de tipo `Select` para elegir Caso, Prestador y Práctica, un campo de tipo `Input` restringido a números para ingresar el costo, y un campo de tipo `TextArea` para ingresar los detalles de la solicitud.

Las variables necesarias para el funcionamiento de este formulario son cinco. Las primeras tres son de tipo `External API`, es decir acceden a un recurso del modelo de negocio a través de la API REST.

Tabla 5.8: Variables de tipo `External API` en la instanciación de "Solicitud de práctica"

Atributo	**Descripción**
`varCasos`	`../API/bdm/businessData/com.company.model.Caso?q=find&p=0&c=25`
`varPracticas`	`../API/bdm/businessData/com.company.model.TipoDePractica?q=find&p=0&c=25`
`varPrestadores`	`../API/bdm/businessData/com.company.model.Prestador?q=find&p=0&c=25`

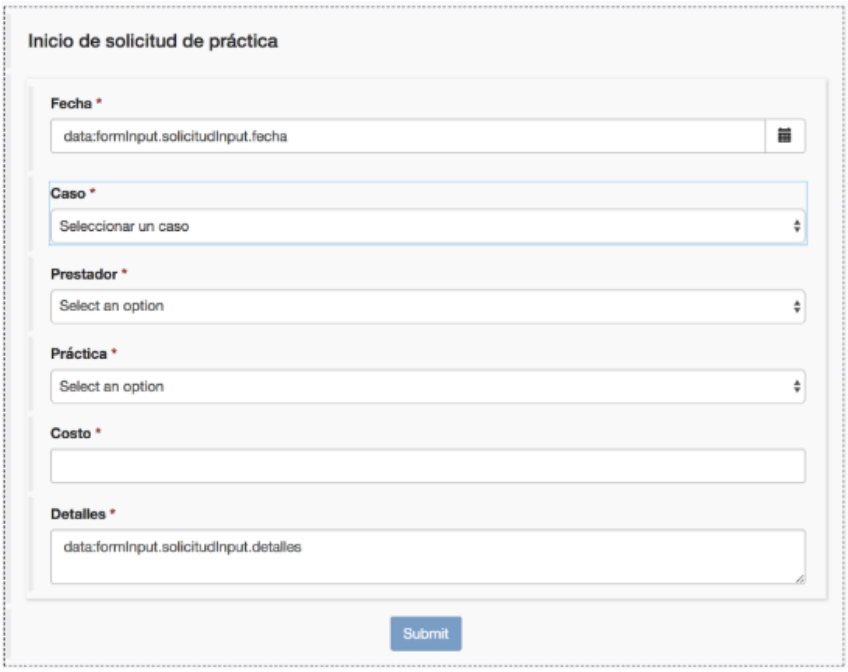

Figura 5.17: Formulario de instanciación de "Solicitud de Prácticas"

La cuarta variable es `formInput`, de tipo `Javascript Expression`, que se inicializa con valores default.

```
return {
   "solicitudInput" : {
       "id_practica" : 0, "id_caso": 0, "detalles" : "",
       "costo": 0, "fecha" : new Date(), "id_prestador" : 0
   }
};
```

La última variable necesaria es `formOutput`, también de tipo `Javascript Expression`, que se inicializa de la siguiente forma:

```
return {
   'solicitudInput': \$data.formInput.solicitudInput
};
```

El input de tipo `select` que se utiliza para elegir el tipo de práctica está configurado para mostrar una lista de las prácticas disponibles a partir de la variable `varPracticas`. Sabemos que `varPracticas` se inicializa con el llamado a la API REST y que tendrá como valor un arreglo con los objetos que representan cada tipo de prática. Cada elemento es un pequeño objeto JSON con un `persistenceId`, un `nombre` y `una descripción`.

En este widget, la clave mostrada es una composición del nombre y la descripción de cada `TipoDePractica`, mientras que el valor de retorno es el `persistenceId`, y la variable vinculada al valor seleccionado en el widget es el atributo `id_practica` del elemenot `formInput.solicitudInput`.

De forma muy similar se configuran los Select de casos y prestadores. El botón de Submit está configurado para no habilitarse hasta que todos los campos del formulario hayan sido completados correctamente. Esto evita que se puedan llegar a enviar datos inválidos como entradas del contrato.

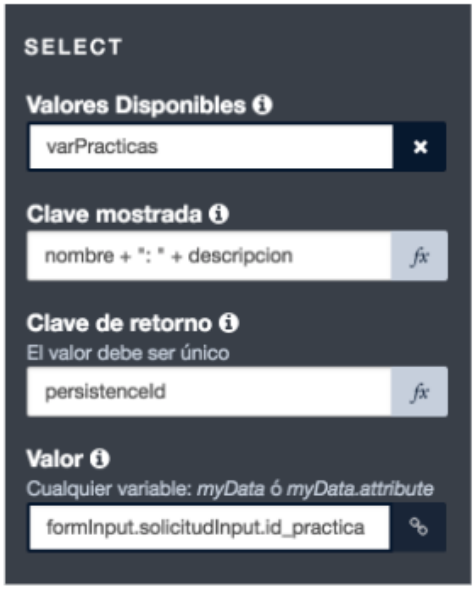

Figura 5.18: Configuración del widget SELECT

5.4.3. Variables de negocio

En el caso de este proceso, se definen tres variables de negocio:

Tabla 5.9: Variables del proceso Solicitud de prácticas

Variable de negocio	Tipo	Contenido
`caso`	`model.Caso`	El caso asociado
`cobertura`	`model.Cobertura`	La cobertura del asegurado
`solicitud`	`model.Solicitud`	La solicitud que se está creando

La variable caso se inicializa utilizando el DAO de `caso` para realizar una búsqueda por `persistenceId`.

```
casoDAO.findByPersistenceId(solicitudInput.id_caso.toLong());
```

La variable cobertura se inicializa de forma muy similar, pero en este caso se obtiene la cobertura del caso en cuestión.

```
def caso = casoDAO.findByPersistenceId(solicitudInput.id_caso.toLong())
return caso.cobertura
```

La variable `solicitud` se inicializa como una nueva instancia de la clase `Solicitud`. Luego se establecen los valores de cada uno de sus atributos. Para ello, es necesario recurrir a algunos accesos a DAOs, concretamente el de `TipoDePractica`, `Prestador` y `Caso`, realizando búsquedas por id utilizando los ids recibidos en `solicitudInput` a través del contrato. El estado se inicializa como "Pendiente" y la fecha, los detalles y el costo también se inicializan a partir de `solicitudInput`, del contrato, con los valores de los atributos correspondientes.

```
def solicitudVar = new Solicitud()
solicitudVar.practica =
    tipoDePracticaDAO.findByPersistenceId(solicitudInput.id_practica.toLong())
solicitudVar.detalles = solicitudInput.detalles
solicitudVar.fecha = solicitudInput.fecha
solicitudVar.prestador = prestadorDAO
      .findByPersistenceId(solicitudInput.id_prestador.toLong())
```

```
solicitudVar.caso = casoDAO.findByPersistenceId(solicitudInput.id_caso.toLong())
solicitudVar.estado = "Pendiente"
solicitudVar.costo = solicitudInput.costo
return solicitudVar
```

El `estado` de la solicitud y los `registros` del caso asociado se irán modificando a medida que el proceso avance. Veremos cómo se manipulan los datos en la implementación de cada tarea del mismo.

5.4.4. Configuración de tareas y compuertas

Al igual que en la sección 5.3.4, en esta sección explicaremos la implementación restante del proceso. Es decir compuertas, tareas, formularios, operaciones. Este proceso tiene tres tareas humanas y dos tareas de servicio. Recordemos que las tareas de servicio en Bonita son tareas sin interacción humana que representan simplemente acciones automáticas como cambios en los datos en el proceso o accesos a servicios o bases de datos.

Tarea "Evaluar solicitud"

Esta tarea es una Tarea Humana que permite a un usuario coordinador de asistencia evaluar la solicitud y determinar si se puede aprobar o no. Si se encuentran ítems que no corresponden a la cobertura o al caso, corresponde rechazarla. La tarea actúa como un primer filtro para que los supervisores tengan menos trabajo, ya que si la solicitud no corresponde nunca llega a manos de un supervisor para su correspondiente análisis.

El Contrato de esta tarea tiene un único ítem: `estado`, de tipo `TEXT`. En el formulario se muestran todos los datos del caso: voucher, fecha inicio cobertura, fecha fin cobertura, plan, asegurado y fecha del caso actual. A su vez, se muestran los registros del caso, por cada uno su fecha y detalles. Y por último se muestra los datos de la solicitud actual: fecha, prestador, práctica, detalles y costo. Todos estos datos permiten al coordinador realizar un análisis adecuado de la solicitud. Se permite elegir entre dos opciones para cambiar el estado de la solicitud: "Rechazada" o "Preaprobada". Es obligatorio seleccionar un nuevo estado antes de completar la tarea.

Cuando la interacción humana finaliza, se realiza una única operación post-tarea a partir del dato de entrada `estado` recibido en el contrato de la misma. Se establece el estado de la solicitud en base al nuevo estado seleccionado.

Las siguientes dos compuertas determinan la próxima tarea en el flujo, dependiendo del estado seleccionado y de la evaluación del costo en relación al presupuesto total y disponible de la cobertura.

Compuerta "¿Rechazada?"

Compuerta en la que según el nuevo estado de la solicitud, elegido en el paso previo, se determina la tarea que continúa. Si la solicitud fue Rechazada, se pasa directamente a la tarea **Rechazar solicitud**. Si la tarea pasó a estado "Preaprobada", se pasa a la compuerta **¿Presupuesto automático?**.

Tarea "Rechazar solicitud"

Tarea de servicio que, sin interacción humana, actualiza el estado de la solicitud al valor "Rechazada" a través de una operación post-tarea. Luego de esta tarea, la instancia del proceso termina con el evento de fin **Fin Rechazo**.

Compuerta "¿Presupuesto automático?"

En esta compuerta se evalúa una regla de negocio que determina si la solicitud puede ser aprobada de forma automática en base al costo y al presupuesto total y disponible de la cobertura en cuestión. Para ello, el costo de la solicitud no debe ser mayor al monto disponible de la cobertura en cuestión y tampoco debe superar el 5 % del presupuesto total del plan de la cobertura asociada al caso actual.

```
def costo = solicitud.costo
def disponible = cobertura.presupuestoDisponible
def total = cobertura.planCobertura.presupuesto
return (costo <= disponible) && (costo <= (total * 0.05))
```

Si la compuerta evalúa a verdadero, se pasa directamente a la tarea **Autorización**. Si la compuerta evalúa a falso, el flujo por defecto es continuar con la tarea "**Revisión de supervisor**".

Tarea "Autorización"

Tarea de servicio que, sin interacción humana, realiza tres operaciones. Actualiza el estado de la solicitud al valor "Aprobada", agrega un nuevo registro al caso indicando la aprobación de la solicitud, y por último sustrae al presupuesto disponible de la cobertura el costo de la solicitud aprobada.

El siguiente es el código de la operación que agrega un nuevo registro al caso asociado a la solicitud:

```
def nuevoRegistro = new Registro()
nuevoRegistro.fecha = Calendar.getInstance().getTime()
nuevoRegistro.detalles = "Autorizada practica "+solicitud.practica.nombre+" al prestador"+
    solicitud.prestador.nombre+", "+solicitud.prestador.ciudad+", "+solicitud.prestador.pais+
    ". "+solicitud.detalles
nuevoRegistro.prestador = solicitud.prestador
return nuevoRegistro
```

Luego de esta tarea, la instancia del proceso termina con el evento de fin **Fin autorización**.

Tarea "Revisión de supervisor"

Tarea humana que consiste en revisar la pre-aprobación y determinar si es necesaria o no la evaluación de un médico especialista. De no ser necesaria, el supervisor deberá aprobar o rechazar la solicitud directamente.

Esta tarea es una de las dos que están integradas con firma digital. Es por ello que el contrato cambia ligeramente. El contrato de esta tarea está compuesto por tres entradas:

Tabla 5.10: Contrato de la tarea con firma digital "Revisión de supervisor"

Atributo	Tipo	Contenido
`datos`	`COMPLEX`	Los datos de la tarea a firmar, que en este caso tiene un atributo que representa el estado de la solicitud.
`firma`	`TEXT`	La firma generada desde el formulario al completar la tarea.
`username`	`TEXT`	El nombre de usuario de quien completó el formulario de la tarea y generó la firma.

Se aplicará una restricción a este contrato, que verificará la firma respecto al atributo `datos`, utilizando la clave pública del usuario que completó la tarea.

```
return Verifier.verify(publicKey, signature, toVerify);
```

El formulario muestra los datos del caso: voucher, fecha inicio de cobertura, fecha fin de cobertura, plan asegurado y fecha del caso actual. También se muestran los registros del caso, y los datos de la solicitud actual: fecha, prestador, práctica, detalles y costo. En base a estos datos, se le solicita al supervisor que elija una acción a tomar: "Solicitar revisión de especialista", "Aprobar solicitud" o "Rechazar solicitud".

Por último se le solicita firmar la tarea, para lo cual se dispone de un selector de archivo para elegir el certificado. A continuación el formulario permite firmar los datos con la clave privada del usuario. Se debe ingresar la contraseña en un cuadro de diálogo para poder acceder al mismo. Par ellos se utiliza un widget personalizado de Bonita, creado con el fin de integrar la firma digital. Una vez que se selecciona una respuesta y se firma la tarea, se habilita el botón de submit para poder enviar el formulario.

Esta estructura de tres elementos llamados `datos`, `firma` y `username`, la restricción de contrato descrita y la utilización de este widget personalizado para firmar, se mantendrán para ambas tareas que integran firma digital. Los detalles se desarrollarán en una sección posterior.

La única operación post-tarea consiste en establecer el nuevo estado de la solicitud según lo elegido en el formulario, puede ser "En Revisión", "Aprobada" o "Rechazada".

Compuerta "¿Requiere revisión médica?"

Esta compuerta evalúa el estado de la solicitud recientemente actualizado y si es igual a "En Revisión", la siguiente tarea en el flujo será **Revisión de especialista**. Caso contrario, el flujo continuará con la compuerta **Tipo de respuesta**.

Tarea "Revisión de especialista"

Esta tarea es una Tarea Humana que consiste en que un especialista evalúe una solicitud para determinar si es pertinente o no. Es la segunda y última tarea con firma digital.

El contrato mantiene la misma estructura y tipos de datos que en la otra tarea con firma digital, pero cambia la semántica de los datos y su estructura interna.

Tabla 5.11: Contrato de la tarea con firma digital "Revisión de especialista"

Atributo	Tipo	Contenido
`datos`	`COMPLEX`	Los datos de la tarea a firmar, que en este caso tiene dos atributos: estado y comentarios, que contienen el estado de la solicitud y los comentarios del especialista.
`firma`	`TEXT`	La firma generada desde el formulario al completar la tarea.
`username`	`TEXT`	El nombre de usuario de quien completó el formulario de la tarea y generó la firma.

A nivel restricciones de contrato, aplica la misma restricción que para **Revisión de Supervisor** con la misma implementación. Se verifica la firma respecto a la entrada `datos`, utilizando la clave pública del usuario que completó la tarea.

El formulario muestra los datos del caso, y se le solicita al usuario que elija una acción a tomar: "Aprobar solicitud" o "Rechazar solicitud". Adicionalmente se provee un campo para completar comentarios. Por último, igual que en la tarea **Revisión de supervisor**, se le solicita al usuario firmar la tarea.

Se ejecuta una única operación post-tarea, que consiste en establecer el nuevo estado de la solicitud, que según lo elegido en el formulario puede ser "Aprobada" o "Rechazada".

Compuerta "Tipo de respuesta"

Esta compuerta evalúa el estado de la solicitud nuevamente: si es igual a "Aprobada" el flujo continúa con la tarea **Autorización** descrita anteriormente, y si es igual a "Rechazara" el flujo continúa con la tarea **Rechazar solicitud** también descrita anteriormente.

5.5. Implementación de mecanismos de firma digital

Para la implementación de la generación de claves, firmado de tareas y verificación de firmas, se realizaron diversas pruebas en base a la tecnología disponible y siempre pensando en la integración con un BPMS.

Se dividió el problema en dos partes. Dado que un **BPMS** es un sistema **distribuido**, parte de la solución se ejecuta del lado servidor y parte del lado cliente. La mayoría de los BPMS en el lado cliente se ejecutan en el contexto de un navegador, por lo cual para esta parte, la creación de firmas, se utilizó **Javascript**. Para la parte que se ejecutaría en el servidor, es decir la generación de par de claves y la verificación de firmas, se utilizó **Java**.

En el caso de la parte implementada en javascript, se hizo uso de una librería llamada **Forge** [3], que provee algunas implementaciones de mecanismos criptográficos en Javascript, y resultó ser la más utilizada y aceptada por la comunidad frontend. En el caso de la parte implementada en Java, se hizo uso de una librería llamada **BouncyCastle** [6] que en el lenguaje Java facilita la realización de operaciones criptográficas diversas.

5.5.1. Mecanismo de generación de par de claves en Java

El mecanismo de generación de par de claves propuesto consiste en la creación de un proyecto Java que permita encapsular gran parte del comportamiento necesario para generar pares de claves, almacenarlas en archivos, recuperarlas, generar firmas, y verificar firmas.

Comenzando por la generación de claves, se definen de dos nuevas clases Java: `EncodedKeyPair` y `KeysGenerator`.

La primera permite instancias objetos de su tipo y cuyas instancias contendrán 2 atributos: la clave privada codificada en formato **PKCS8 protegido**, y la clave pública codificada en un certificado **x509**.

```
public class EncodedKeyPair {
   private PemObject protectedPKCS8prvtKey;
   private PemObject x509EncodedCertificate;
   ...
}
```

La segunda, es una clase con dos métodos estáticos: uno para genera un par de claves que dado un nombre y una contraseña para proteger la clave privada genera un par de claves y devuelve un objeto de tipo `EncodedKeyPair`, y otro para guardar en un archivo cada uno de los objetos codificados.

```
public class KeysGenerator {
   private static String KEY_ALGORITHM = "RSA";
   public static EncodedKeyPair generateKeyPair(String name, String passwordForPrivateKey) {
       ... }
   public static void saveToFile(PemObject pemObject, String fileName, String path) throws
       IOException { ... }
   ...
}
```

Para la implementación del método `generateKeyPair` se utiliza `KeyPairGenerator` del paquete `java.security`, con el **algoritmo RSA** del proveedor **BC** (BouncyCastle). Luego, la clave privada es codificada en formato **PKCS8**, típicamente, con cifrado **PBE SHA1 3DES** protegiéndola con la contraseña recibida como parámetro. La clave pública es codificada como un **certificado X509**, formato estándar para certificados de claves públicas. El certificado se carga con información básica, una fecha de expiración por defecto y el nombre recibido como parámetro. Finalmente, se retorna un objeto de tipo `EncodedKeyPair`, que contiene cada uno de los objetos PEM generados previamente. [12]

```
public static EncodedKeyPair generateKeyPair(String name, String passwordForPrivateKey) {
   try {
       //get KeyGenerator and random instance and initialize the generator with the secure
           random
       KeyPairGenerator keyGen = KeyPairGenerator.getInstance(KEY_ALGORITHM, "BC");
       SecureRandom random = SecureRandom.getInstanceStrong();
       keyGen.initialize(1024, random);
       //generate the key pair
       KeyPair pair = keyGen.generateKeyPair();
       PrivateKey priv = pair.getPrivate();
       //store keys in files
       PemObject protectedPKCS8prvtKey = protectPrivateKey(priv, passwordForPrivateKey);
       PemObject x509Cert = createCertificate(pair, caPrivateKey, name);
       return new EncodedKeyPair(protectedPKCS8prvtKey, x509Cert);
   } catch (NoSuchAlgorithmException | NoSuchProviderException | OperatorCreationException
           | IOException e) {
       // ... ...
   }
}
```

Estas clases se pueden exportar en un archivo JAR construyendo el proyecto Java al que pertenecen. Esto permite su utilización en diversos entornos. En nuestro caso de estudio, en Bonita BPM a través de la importación de archivos JAR.

5.5.2. Mecanismo de firma de datos en Javascript

El mecanismo de firma de datos en Javascript se implementó también en un pequeño proyecto. Se crearon algunos archivos Javascript con un conjunto de funciones que se pueden integrar en un componente. El componente puede ser una directiva angular, un web component o un componente JQuery. Para fines prácticos **primero** se realizaron pruebas con un componente **JQuery**, y **luego** para la **integración con Bonita** se construyó una directiva o **componente de Angular**.

Se utilizó una librería llamada **forge**: una implementación nativa de TLS en Javascript que brinda herramientas para escribir aplicaciones web basadas en criptografía. Principalmente se hizo uso de sus implementaciones de estándares PKI y Message Digests. [3]

Las principales funciones de nuestro componente son la **lectura de la clave privada** a partir de un **archivo PEM** protegido por **contraseña**, y la **generación de la firma** a partir de la **clave privada** y una determinada cadena de caracteres a firmar. A continuación se describen las funciones `doLoadPrivateKeyFromPEM` y `doSign`.

```
function doLoadPrivateKeyFromPEM (pemFile, password) {
   var deferred = $q.defer();
   var reader = new FileReader();
   reader.onload = (function (){
       return function(evt) {
           var content = evt.target.result;
```

```
            try {
                var privateKey = forge.pki.decryptRsaPrivateKey(content, password);
                deferred.resolve(privateKey);
            } catch (e){
                deferred.reject(e);
            }
        };
    })();
    reader.readAsBinaryString(pemFile);
    return deferred.promise;
}
```

La función `doLoadPrivateKeyFromPEM` recibe dos parámetros, el **archivo** a leer, y la **contraseña** que lo protege. Lee el archivo y debe decodificar su contenido. Para ello, utiliza la contraseña recibida como parámetro y la función `forge.pki.decryptRsaPrivateKey` provista por la librería Forge. **Retorna una promesa** que se resuelve favorablemente cuando se pudo decodificar la clave y es rechazada cuando hubo algún error.

```
function doSign (privateKey, string) {
    //Create message digest with SHA512 (SHA-2)
    var md = forge.md.sha512.create();
    md.update(string);
    //sign with RSA algorithm (default)
    return privateKey.sign(md);
}
```

La función `doSign` recibe dos parámetros, la **clave privada** y una **cadena de caracteres a firmar**. Lleva a cabo la creación de la firma creando un **resumen de mensaje** (message digest) con **SHA512** y firmando el mismo con el **algoritmo** predeterminado **RSA**.

A fines prácticos, estas funciones se integraron para las pruebas en un componente JQuery llamado `DigitalSignatureUIC`, que además implementa la presentación y la interacción con el usuario para seleccionar un archivo, solicitar la contraseña y permitir firmar los datos.

En una sección posterior se explicará el mecanismo de integración de estas funciones con **Bonita BPM** a través de la creación de una **directiva Angular** con el editor de widgets personalizados provisto por Bonita en su componente de Editor de formularios.

5.5.3. Mecanismo de verificación de firma en Java

El mecanismo de verificación de firma propuesto en Java, forma parte del mismo proyecto que contiene las clases para la generación de par de claves.

En esta sección se describe la clase `Verifier`, que tiene dos métodos estáticos: `loadPublicKey` y `verify`.

El método `loadPublicKey` recibe una **cadena** con la ruta a un archivo que contenga un **certificado X509** y **retorna un objeto** de la clase `java.security.PublicKey`.

El método `verify` recibe un **objeto** de la clase `PublicKey`, un **arreglo de bytes** que representa la firma a verificar y una cadena que representa el texto firmado. **Retorna un valor booleano** que representa si la firma ha sido verificada correctamente o no.

Para verificar la firma, se utiliza el mismo **algoritmo** que para la generación de firmas: **SHA512withRSA**. Se crea una instancia de `java.security.Signature` para este algoritmo, se la inicializa con la **clave**

pública recibida, se utiliza un buffer para leer los bytes de los datos a verificar e ir actualizando la firma hasta leerlos todos, y por último, se verifica la firma comparándola con la firma recibida.

```
public class Verifier {
   static String ALGORITHM = "SHA512withRSA";
   public static PublicKey loadPublicKey (String fileName) throws IOException {
       File initialFile = new File(fileName);
       InputStream res = new FileInputStream(initialFile);
       Reader fRd = new BufferedReader(new InputStreamReader(res));
       PEMParser pemParser = new PEMParser(fRd);
       X509CertificateHolder certificateHolder =
           (X509CertificateHolder)pemParser.readObject();
       SubjectPublicKeyInfo publicKeyInfo = certificateHolder.getSubjectPublicKeyInfo();
       JcaPEMKeyConverter converter = new JcaPEMKeyConverter().setProvider("BC");
       return converter.getPublicKey(publicKeyInfo);
   }
   public static boolean verify(PublicKey publicKey, byte[] sigToVerify, String
       textToVerify) {
       try {
           Signature signature = Signature.getInstance(ALGORITHM);
           signature.initVerify(publicKey);
           BufferedInputStream bufin = new BufferedInputStream(new
               ByteArrayInputStream(textToVerify.getBytes()));
           byte[] buffer = new byte[1024];
           int len;
           while (bufin.available() != 0) {
               len = bufin.read(buffer);
               signature.update(buffer, 0, len);
           }
           bufin.close();
           return signature.verify(sigToVerify);
       } catch (NoSuchAlgorithmException e) { ... }
   }
}
```

El código debe manejar varias excepciones posibles, como un algoritmo inválido, una clave pública inválida, una firma inválida o un error de entrada/salida al leer o escribir el buffer.

Al ser parte del mismo proyecto, puede exportarse junto con las clases mencionadas en la sección 5.5.1 y agregarse como una dependencia en el entorno de Bonita BPM.

5.6. Integración de firma digital en Bonita BPM

En base a los mecanismos criptográficos descritos en la sección 5.5, en esta sección se explicará la implementación de la integración con el entorno de Bonita BPM. Bonita permite incorporar JARs Java al proyecto para poder utilizar código externo, y permite importar assets Javascript, HTML y CSS para desarrollar los formularios y widgets personalizados para la parte visual de las tareas y la interacción de los usuarios finales.

Para agregar dependencias Java externas, se debe acceder a la opción “Gestione jars...” del menú “Desarrollo” de Bonita BPM. Agregaremos como dependencia los siguientes JARs:

- gson-2.8.2: lo utilizaremos para facilitar la transformación de los datos entre JSON String y Objetos Java.
- Backend-BPM-Crypto-integration: un JAR que contiene las clases descritas en las secciones 5.5.1 y 5.5.3: `EncodedKeyPair`, `KeysGenerator` y `Verifier`.

A continuación veremos de qué manera se pueden utilizar estas dependencias externas dentro de la implementación de un proceso de negocio con Bonita BPM para llevar a cabo la integración explicada en el capítulo 4, en las secciones 4.5 y 4.10.

5.6.1. Propuesta de integración del mecanismo de generación de claves

La generación de claves propuesta forma parte del proceso de Generación de Par de Claves (Fig. 4.6), que a su vez es utilizado dentro del proceso de Contratación e Incorporación de personal (Fig. 4.4). Dentro del alcance de esta tesina no se realizó la implementación de estos procesos.

La primera tarea, **Ingresar contraseña de clave privada**, debe mostrar un formulario al usuario para ingresar una contraseña que proteja su clave privada en el **formato PKCS8** protegido, como se explica en la sección 5.5.1.

En la tarea **Generar par de claves**, que es una tarea automática de tipo servicio, se utilizan las clases Java mencionadas en dicha sección y se genera un par de claves para el usuario. La contraseña usada para proteger la clave privada es la ingresada en la tarea anterior, y el nombre usado para configurar los datos del certificado de clave pública es el nombre de usuario del usuario que está generando sus claves.

Una de las opciones disponibles en **Bonita** para la **utilización de código externo** importado a través de la gestión de **JARs** es un conector de tipo **Script Groovy**, que consiste en un script que se ejecuta en un punto dado de la ejecución de un proceso que puede ser el **inicio** -conector de entrada- o el **final** -conector de salida- de una **tarea de servicio**. En un script groovy se puede acceder a las variables de la tarea y del proceso, y se pueden utilizar las clases java disponibles con solo importarlas.

En la tarea **Asociar clave pública al usuario**, que también es una **tarea de servicio**, se vincula el archivo de clave pública generado al usuario de Bonita Studio y se lo almacena de forma segura en el servidor. Por último, en la tarea **Descargar archivo de clave privada**, el usuario descarga su archivo de clave privada. Esta descarga, al igual que toda la interacción con la aplicación web de Bonita Studio en cuestión, es realizada a través de una **conexión segura por HTTPS**.

5.6.2. Integración de mecanismo de firma

Javascript es un lenguaje que se **ejecuta en el navegador**, y cuyo código **viaja por la red**. A su vez es **muy fácil sobreescribir funciones** de objetos javascript y no tiene el nivel de seguridad de un lenguaje compilado como Java. Por lo tanto, la **única manera** de que el mecanismo de firma sea **lo más seguro posible** es que todo el sitio web se sirva **a través de HTTPS**. De esta forma, todo el código Javascript y **los datos viajan por la red autenticados** por el servidor y **encriptados**. Esto impide que otro código malicioso proveniente de un tercero pueda descargarse de la red e interferir en la implementación de la firma. El único peligro restante reside en el propio navegador, es decir si el dispositivo que el usuario utiliza para acceder a la aplicación web, su sistema operativo o el navegador no son seguros.

Con esta premisa, se debe integrar de la forma más simple posible, el mecanismo de firma elaborado en la sección 5.5.2. Se propone crear una **solución reutilizable**, que nos permita agregar la funcionalidad para generar firmas a varios formularios de varios procesos, sin reescribir demasiado código.

Bonita tiene un **editor de formularios** para generar de forma sencilla los formularios de interacción humana para las tareas de este tipo. Y además de proveer un conjunto de widgets predeterminados, provee la **posibilidad de crear widgets personalizados** y reutilizarlos en tantos formularios como se desee.

Los widgets predeterminados disponibles en formularios de Bonita Studio: `Image`, `Table`, `Text`, `Title`, `Link`, `Button`, `Date Picker`, `Upload`, `Radio Buttons`, `Checklist`, `Checkbox`, `Select`, `Autocomplete`, `Text Area`, `Input`, `Form Container`, `Tab Container` y `Container`.

Tanto para la creación de widgets personalizados como para la creación de formularios, **se pueden incorporar assets externos**, que pueden ser archivos **Javascript**, **CSS** o **Imágenes**.

El **Editor de Widgets** personalizados, consiste en una interfaz **web** simplificada para crear un componente angular reutilizable, que tenga propiedades como interfaz para su utilización, un controlador y una plantilla para su presentación.

Comenzando por las propiedades del componente de firma digital, para cada una debemos definir el nombre, tipo y la manera en que se enlazarán los datos. Serán configuradas las propiedades de la tabla 5.12.

Tabla 5.12: Propiedades del widget personalizado de firma digital

Propiedad	**Etiqueta**	**Tratar como**
`value`	Value	Enlace bidireccional
`formOutput`	Variable salida del formulario	Enlace bidireccional

La propiedad `value` se utiliza para almacenar el la cadena que representa el valor de la firma. La propiedad `formOutput` se utiliza para obtener el valor de salida del formulario y con este construir el texto a firmar.

```
<div class="digital-signature-uic">
   <span ng-if="environment" ...>
   <p><label for="file">{{texts['label.text']}}</label></p>
   <input type="file" class="file-loader" name="file" >
   <button type="button" class="sign-button" ng-click="sign()" ng-hide="!file">
       {{texts['signButton.text']}}
   </button>
</div>
```

Para la presentación del componente, se construyó una plantilla simple, que muestra algunos elementos HTML básicos: Un label para el archivo, un input de tipo `file` para seleccionar el archivo con la clave privada, un botón para realizar la firma, que permanece oculto hasta que se haya seleccionado el archivo. Este botón, al ser clickeado, ejecuta la función `sign()` del controlador, que se describe a continuación.

El controlador será llamado `digitalSignatureController` y contiene la inicialización y lógica necesaria para el funcionamiento del componente.

Algunas funciones y variables son privadas, y otras son expuestas al template. Concretamente, el `$scope` del componente, que expone datos al template, tiene los atributos `texts` que contiene los textos para el idioma actual, `file` que contendrá el archivo seleccionado, `sign()` que será la función para firmar y `properties` que contiene las propiedades descritas en la tabla 5.12.

```
$scope.sign = function (){
   promptForPassword().then(function (password){
       doLoadPrivateKeyFromPEM($scope.file, password).then(function (privateKey){
```

```
            if(privateKey){
                var jsonData = JSON.stringify($scope.properties.formOutput);
                var textToSign = btoa(unescape(encodeURIComponent(jsonData)));
                var signatureString = btoa(doSign(privateKey, textToSign));
                $scope.properties.value = signatureString;
            } else {
                alert("Invalid password");
            }
        });
    }, function (err){
        console.log(err);
    });
}
```

La **forma elegida para realizar la firma** es tomar el valor de `formOutput`, transformarlo en un **JSON** en formato cadena de caracteres, codificarlo en **Base64** y **realizar la firma** a partir de estos datos.

Las **variables privadas** del controlador son `LANGUAGE` que contiene el lenguaje actual, `TEXTS` que representa un mapa con los textos en cada idioma, `promptForPassword()` que es una función para solicitar la contraseña de la clave privada al usuario, `doSign()` que implementa la firma de los datos y `doLoadPrivateKeyFromPEM()` que usa la clave para desencriptar el archivo.

Las funciones `doSign()` y `doLoadPrivateKeyFromPEM()`, fueron explicadas en la sección 5.5.2.

Con esto **terminamos** de definir la **implementación** del componente de **Firma Digital**, un widget personalizado de Bonita que podremos colocar en múltiples formularios. En el editor de formularios de Bonita aparece una sección con los widgets personalizados creados, y las propiedades que se especificaron en su definición se pueden completar en la interfaz gráfica web del mismo.

De esta forma, podremos utilizar el nuevo widget para los formularios de interacción humana de las tareas que requieren firma digital de los procesos mencionados en el capítulo 4, la tarea **Revisión de supervisor**, y **Revisión de especialista**.

En cada uno de estos formularios, deberemos agregar el widget de firma digital y pasarle como parámetro las variables `varFirma` y `formInput` mencionadas en las secciones 5.3.4 y 5.4.4 en las propiedades **Valor** y **Variable de salida del formulario**. Si bien el widget espera recibir la **variable de salida del formulario**, en este caso `formInput` representa todos los datos ingresados por el usuario en el formulario excepto por la firma digital y el nombre de usuario. Por tanto, esto es lo que consideraremos **semánticamente**, la salida del formulario que le interesa a nuestro componente de firma.

Para el correcto funcionamiento del formulario y el mecanismo de firmado, la variable `formOutput` del mismo deberá ser inicializada con una expresión Javascript de la siguiente forma:

```
return {
   datos: $data.formInput,
   firma: $data.varFirma,
   username: $data.userName
};
```

Esto coincidirá con el contrato de las tareas que requieran firma digital.

La variable `username` en el formulario se inicializa de la siguiente manera:

```
return $data.user.user_name;
```

Por último, el botón de tipo submit del formulario, es configurado con una condición que lo deshabilita o habilita, en la propiedad **Desactivado** del widget Button:

```
$form.$invalid || !varFirma
```

Con todo esto, el usuario deberá completar los datos del formulario, luego firmar los mismos, y finalmente se habilitará el botón **Enviar** para poder completar la tarea. En la próxima sección se explica el mecanismo con el cual se integra la verificación de la firma en Bonita.

5.6.3. Integración de mecanismo de verificación de firma

El contrato para las tareas con firma digital en nuestra implementación tendrá siempre la misma estructura base, tal y como se explica en la sección 5.4.4, en las tablas 5.10 y 5.11. La siguiente es la estructura general propuesta para los contratos de todas las tareas que tengan firma digital para nuestra integración de firmado en procesos BPM.

Tabla 5.13: Contrato de una tarea con firma digital

Atributo	Tipo	Contenido
`datos`	`COMPLEX`	Todos los datos de la tarea, excepto la firma y el nombre de usuario. El contenido de este objeto `COMPLEX` dependerá de los datos concretos de cada tarea.
`firma`	`TEXT`	La firma generada desde el formulario al completar la tarea.
`username`	`TEXT`	El nombre de usuario de quien completó el formulario de la tarea y generó la firma.

En `datos` estarán agrupados todos los datos ingresados en el formulario en formato JSON, y adicionalmente se dispondrá de `firma` y `username` como datos adicionales necesarios para la verificación de la firma y por tanto la **validez**, **integridad** y **autenticidad** de los datos, y de la tarea propiamente dicha.

Bonita BPM provee una manera de **verificar** que el **contrato** de una tarea sea **válido**. Se pueden **definir restricciones**, validaciones en forma de scripts que realizan chequeos y retornan un valor booleano, como se explica en las secciones anteriores. Estas restricciones **se aplican** sobre los **atributos del contrato** definido para una tarea. El contrato se valida luego de que se completa el formulario y se envía al servidor, y en base a la respuesta, se define si la tarea se puede completar. En caso de que existan errores en la validación, el formulario puede manejarlos y mostrar mensajes de error correspondientes.

La validación del contrato en el caso de las tareas con firma digital de los procesos mencionados, se realiza con la restricción `comprobarFirma`, cuyo mensaje de error es “La firma no es válida.” y su implementación es la siguiente:

```
...
Security.addProvider(new BouncyCastleProvider());
String file = "./generatedKeys/" + username + ".pem";
try {
   PublicKey publicKey = Verifier.loadPublicKey(file);
   byte[] signature = Base64.decode(firma);
   Gson gson = new Gson();
   String jsonString = gson.toJson(datos);
   String toVerify = jsonString.getBytes("UTF-8").encodeBase64().toString();
   boolean result = Verifier.verify(publicKey, signature, toVerify);
   return result;
```

```
} catch (FileNotFoundException e) {
   ...
   return false;
} catch (IOException e) {
   ...
   return false;
}
```

Se hace uso de la clase `Verifier` importada a través del **JAR** `Backend-BPM-Crypto-integration` agregado a las dependencias del ambiente, cuya implementación es explicada en la sección 5.5.3. Dado que cada usuario tiene asignado un **archivo de clave pública**, se debe acceder al mismo **según el nombre de usuario**, atributo `username` del contrato. El archivo de clave pública asociado al usuario nos permite **verificar su firma**.

Para poder verificar los datos, primero deben ser traducidos a un JSON String en el mismo formato utilizado por la integración de generación de firmas para crear los datos a firmar. En este caso se utiliza la **librería Gson**, también importada como dependencia, para transformar los datos del objeto `COMPLEX datos` en un `JSON-String`. Es importante que el formato y la codificación coincidan exactamente con la implementación de generación de la firma, por lo cual este String se codifica en **UTF-8** y luego en **Base64** de la misma manera que lo hace la directiva angular.

Una vez preparados los datos a verificar, se procede a llamar al método `verify` de la clase `Verifier` importada, con la clave pública, la firma y los datos a verificar como argumentos.

5.6.4. Propuesta de integración de procesos de PKI

En esta sección se elaboran las consideraciones generales sobre la implementación del resto de los procesos descritos en el capítulo 4: “Contratación e Integración de Personal”, “Desvinculación de Personal”, “Generación de par de claves”, “Renovación de par de claves”, “Revocación de clave comprometida” y “Baja de claves”.

Si bien debido al alcance del desarrollo propuesto en esta tesina estas implementaciones no fueron llevadas a cabo, a continuación se enuncian algunas consideraciones sobre su implementación.

Tal y como fue descrito en la sección 5.6.1, el proceso de **Generación de claves** de la figura 4.6 utiliza las clases `KeysGenerator` y `EncodedKeyPair` importadas para generar un par de claves para un usuario dado, permite la descarga de la clave privada y almacena el certificado de clave pública.

El proceso de **Baja de claves**, de la figura 4.6, realiza la desvinculación de un archivo de clave pública dado del usuario en cuestión. Dado que en nuestro caso de estudio la PKI es interna a la organización, este proceso simplemente puede eliminar o renombrar el certificado de clave pública asociado al usuario en cuestión.

Los procesos de “**Contratación e Integración de Personal**” y de “**Desvinculación de personal**” -figuras 4.4 y 4.5 respectivamente- integran los procesos de gestión de claves mencionados en los párrafos precedentes de forma que la administración de claves se ejecute como parte del flujo de los procesos de la organización, integrada en la gestión por procesos, y no como parte de una administración manual de un sector técnico de la organización disociado del esquema BPM.

La implementación de los procesos “**Renovación de par de claves**” y “**Revocación de clave comprometida**” no requiere de una explicación más detallada para el alcance de esta tesina, ya que

basta con decir que utilizan a los procesos de generación y baja de claves que son los procesos centrales de la administración de claves de nuestro esquema.

Pese a no haber implementado los procesos anteriormente mencionados en el desarrollo de esta tesina, la ejecución de los procesos de "**Gestión de caso de asistencia médica al viajero**" y "**Solicitud de prácticas**" fue posible gracias a los usuarios y roles de prueba generados, los datos de prueba mencionados en la sección 5.2.3 y a un procedimiento que permite simular la ejecución del proceso de generación de firmas con fines prácticos.

Los pasos de este procedimiento se enuncian a continuación.

1. Se ejecutó una aplicación Java minimalista que haciendo uso del mecanismo de generación de claves implementado, permitió la generación de claves necesarias para las pruebas.
2. Se renombraron los archivos de claves generados, tanto claves privadas como certificados de clave pública, para que coincidan con cada uno de los nombres de usuarios de nuestros datos de prueba.
3. Los certificados de clave pública se colocaron en un directorio configurado, cuya ruta se asignó a la configuración de la integración del mecanismo de verificación de firma.

Este procedimiento se llevó a cabo para cada uno de los usuarios de prueba de los roles Supervisor y Especialista, es decir los usuarios "sabrina.s" y "florencia.sp". De esta forma, las instancias del proceso de "Solicitud de prácticas", en sus tareas con firma digital, pudieron acceder a los archivos de clave pública correspondientes a cada usuario de prueba y utilizar esos datos para la validar la firma en cada caso.

En conclusión, fue posible validar el correcto funcionamiento de la integración para la generación de firmas y la validación de las mismas. Para cada una de las tareas con firma digital, al utilizar un archivo de clave privada correcto, correspondiente al usuario en cuestión, la validación fue satisfactoria y la tarea pudo completarse exitosamente. En cambio, al utilizar un archivo de clave privada incorrecto, el proceso de validación falló, y la tarea no pudo ser completada.

Capítulo 6

Conclusiones y trabajos futuros

Las organizaciones y empresas utilizan tecnologías de la información para soportar su flujo de trabajo y muchas funcionan orientadas a procesos. A su vez, un gran número utiliza BPM -Business Process Management o Gestión de Procesos de negocio) para implementar sus procesos de negocio.

Las herramientas de BPM disponibles hoy en día tiene cada vez más capacidades, pero la mayoría ofrece pocos mecanismos de seguridad para la autenticación de los usuarios, y la protección de la integridad, confidencialidad y no repudio de la información.

Estos requerimientos de seguridad son más altos en general para todas las organizaciones con la creciente importancia de los datos que circulan por la web. Y lo son aún más elevados para ciertas organizaciones específicas cuyo negocio se basa en datos sensibles.

En este sentido, las herramientas criptográficas en general, y en particular la firma digital, son propuestas en esta tesina como mecanismos para fortalecer los servicios de seguridad mencionados. Sin embargo, en el contexto de BPM las desventajas de los mecanismos de seguridad se hacen más evidentes. La mayoría de los mecanismos de seguridad agregan complejidad para el usuario final, y esto pareciera oponerse a varios de los principios y fundamentos de las implementaciones de BPM para la gestión por procesos. En especial la gestión claves y certificados y la utilización de firmas digitales, conlleva un alto impacto en la simplicidad de uso de un sistema informático.

Es por ello que este trabajo ha elaborado un concepto de integración entre los mecanismos de criptografía como TLS y Firma digital de forma tal que sean lo más transparentes posibles para el usuario, persiguiendo un balance adecuado entre complejidad de uso y seguridad. Con tal fin, se ha propuesto no sólo la integración de la firma digital para tareas críticas dentro de los procesos, sino también la descripción e implementación de procesos de gestión de claves y de contratación y desvinculación de personal que integran la gestión de claves a los procesos más fundamentales de una organización.

Para cumplir los objetivos de este trabajo, se llevaron a cabo las siguientes tareas:

1. Se realizó un análisis de las posibles formas de realizar la integración de la firma digital en un entorno de BPM y la comparación de las alternativas planteadas.
2. Se elaboró el concepto de integración de los mecanismos criptográficos
3. Una vez elegida la mejor alternativa, la integración de la firma digital propiamente dicha ha sido implementada en una prueba de concepto en un proceso puntual, utilizando como ejemplo un caso real de una empresa de asistencias médicas para viajeros. El flujo de trabajo de la misma

fue plasmado en los diseños de varios procesos, de los cuales dos han sido implementados en su totalidad.

4. Para el diseño e implementación de estos procesos se utilizó Bonita Studio y se integró a la herramienta código específico creado para la solución de criptografía de esta tesina.
5. Se redactó este informe, incluyendo un marco teórico de criptografía, firma digital y BPM, una explicación del esquema planteado y el detalle de los procesos y mecanismos criptográficos implementados y las integraciones realizadas.

Los procesos diseñados y las implementaciones realizadas de dos de ellos han permitido plasmar de forma práctica la propuesta de integración elaborada en esta tesina. A medida que se iba desarrollando la implementación, se iban haciendo mejoras y ajustes al esquema planteado y a sus definiciones.

Se han obtenido como conclusión cuatro aspectos de esta integración que cabe mencionar. En primer lugar, se ha logrado un adecuado nivel de transparencia para el usuario final de la herramienta de BPM a pesar de haber integrado un mecanismos de seguridad como la firma digital. En segundo lugar, la integración de la gestión de claves en los procesos de la organización es una integración mucho más simbiótica que la mera introducción de un mecanismo de firma para completar ciertas tareas. Es decir, BPM se ve enriquecido por los mecanismo criptográficos, y a su vez la gestión de claves para la utilización de dichos mecanismos se implementa orientada a procesos, y como parte de los procesos de la organización. En tercer lugar, esta solución no es una solución genérica: los conceptos planteados deberán ser adaptados a cada organización según su estructura y procesos específicos. En cuarto lugar, tareas que normalmente serían realizadas por fuera de la gestión por procesos -probablemente por administradores de sistemas-, como el alta y baja de usuarios y claves, en este esquema son también implementados como procesos de negocio.

Como trabajo futuro se propone el estudio de alternativas a la firma digital, que permitan lograr una mejora en los atributos de seguridad de los procesos de negocio. La web evoluciona constantemente y a un ritmo creciente, de la misma forma que lo hacen los dispositivos móviles y sus aplicaciones. En este marco, tecnologías emergentes, distribuidas y basadas en Cloud pueden ser objeto de estudio para tal fin. Por ejemplo, podría analizarse una integración con una aplicación de generación de códigos para autenticación en dos pasos. Estas aplicaciones, como por ejemplo Google Authenticator, permiten a los sistemas integrarse de forma tal que los usuarios pueden utilizarlas para obtener desde su dispositivo móvil un código que cambia cada 60 segundos y sirve como segundo factor de autenticación. Un atacante que quiera realizar acciones en nombre de un usuario, deberá tener acceso a su dispositivo móvil además de averiguar su contraseña.

Ésta y/u otras alternativas podrían ser estudiadas en un trabajo futuro que tenga como objetivo mejorar la usabilidad del sistema de BPM sin prescindir de las ventajas de un mayor nivel calidad en los servicios de seguridad.

Anexo A

BPMN, orquestación y coreografía

Este anexo introduce a la notación BPMN, y explica la relación de sus elementos con la orquestación y la coreografía de procesos. Basado en [16].

Este anexo introduce a la notación BPMN, y explica la relación de sus elementos con la orquestación y la coreografía de procesos.

La **orquestación** es la forma en que las restricciones entre actividades de un proceso de negocio generan dependencias entre unas tareas y otras, e influyen sobre el orden y la paralelización de la ejecución de tareas. El flujo de la ejecución del proceso está dado por el modelo de orquestación plasmado en el diseño del mismo.

En **BPMN**, los fijos se describen de manera gráfica como parte del diagrama de un proceso. Los objetos de flujo son los bloques constitutivos de un proceso de negocio. Incluyen eventos, actividades y compuertas. La ocurrencia de cambios de estado en el mundo real que son relevantes para el proceso de negocio y más generalmente cualquier cosa relevante que suceda, puede ser representada por eventos. Las actividades representan unidades de trabajo. Las compuertas son usadas para representar la división y unión del flujo de control entre actividades, eventos y otras compuertas.

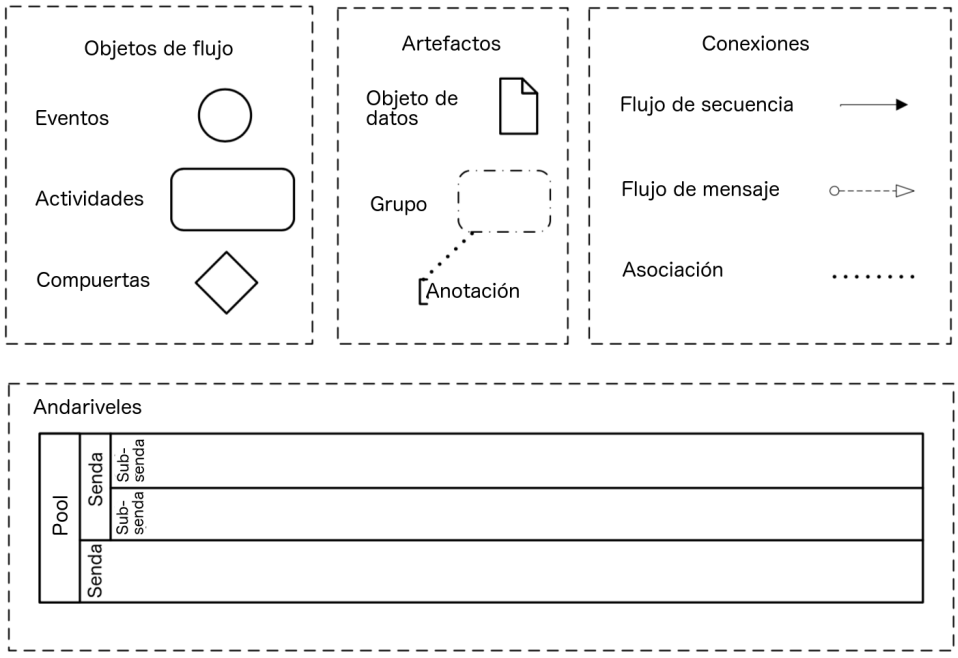

Figura A.1: Categorías de elementos en BPMN

Se utilizan artefactos para mostrar información adicional acerca de los procesos de negocio que no es directamente relevante para la secuencia de ejecución del proceso. Los objetos de datos se representan simplemente con un nombre, la estructura interna de los objetos no puede ser representada en BPMN. Las anotaciones documentan aspectos específicos de los procesos en forma de texto.

Las conexiones entre los objetos, sendas o artefactos se representan con líneas y flechas. Las flechas de secuencia se usan para especificar el orden de flujo entre objetos, mientras que las flechas de mensajes se usan para describir el flujo de mensajes entre actores del negocio representados por pools.

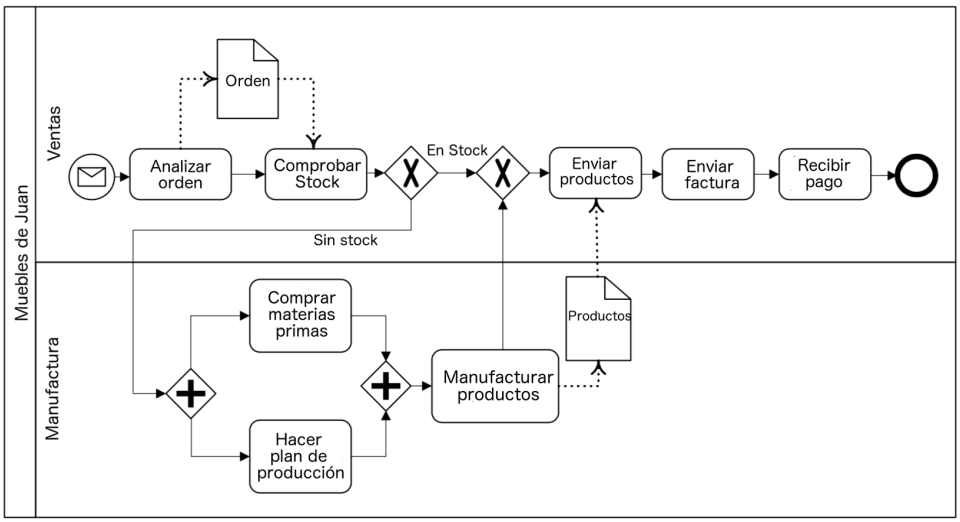

Figura A.2: Diagrama de proceso expresado en BPMN

La figura muestra un diagrama de proceso de negocio que introduce los elementos principales del lenguaje BPMN.

Los roles involucrados en un proceso se representan en el diagrama del proceso como **sendas o lanes** dentro del **pool o contenedor** del proceso. Ventas y manufactura son dos ejemplos de roles dentro del proceso en la figura .A.2

Las actividades son unidades de trabajo. Cada caja con un texto en su interior representa una actividad en el diagrama. Pueden a su vez tener una estructura interna, en cual caso son llamadas subprocesos, que pueden ser expandidos para ver su diseño.

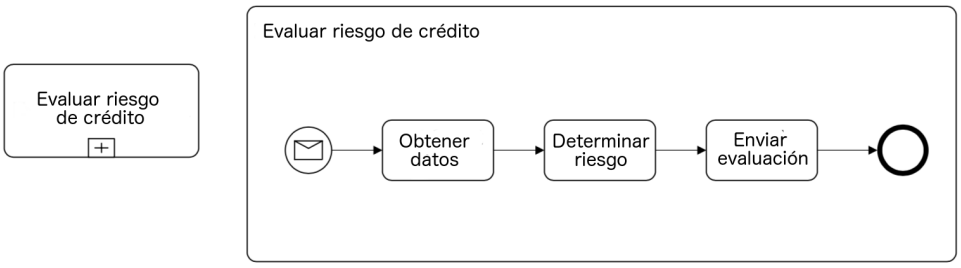

Figura A.3: Ejemplo de subproceso

La estructura de una actividad también puede estar dada por otro proceso global, en cual caso es una **actividad de llamada** y se está reutilizando el proceso llamado.

Las tareas o actividades pueden a su vez ser decoradas con tipos, que permiten identificar fácilmente al lector humano qué tipo de tarea se realizará. Existen tareas de usuario, tareas manuales, tareas de

servicio, tareas de script, entre otras. Esto permite distinguir, por ejemplo, una tarea con intervención humana, una tarea 100 % manual, o una tarea que involucra solamente al sistema o a un servicio web.

Otro elemento fundamental, y uno de los más utilizados en los diagramas de proceso, es el **evento**. Un evento representa una situación en el mundo real a la cual los procesos deben reaccionar, o que los procesos deben generar.

Hay tres categorías de eventos. Los **eventos de inicio** se utilizan para iniciar procesos, los **eventos intermedios** pueden demorar un proceso en cierto punto o pueden ser disparados durante la ejecución del mismo, y los **eventos de fin** causan la finalización de su ejecución.

Hay dos roles posibles de eventos y todo evento juega uno de esos dos roles: **captura** o **lanzamiento**. Un evento juega un rol de **captura** cuando el proceso escucha y espera que un evento suceda. Todos los eventos de inicio son de tipo captura. Por otro lado, un evento juega un rol de **lanzamiento** cuando el mismo proceso activamente genera un evento. Todos los eventos de fin son de tipo lanzamiento. Los eventos intermedios pueden ser tanto de tipo captura como de tipo lanzamiento. Un buen ejemplo es un evento de mensaje intermedio. Puede simbolizar un mensaje que debe llegar al proceso, en cual caso será un evento de captura, y el proceso se quedará esperando a que llegue dicho mensaje para continuar. O puede simbolizar el envío de un mensaje, en cual caso será un evento de tipo lanzamiento.

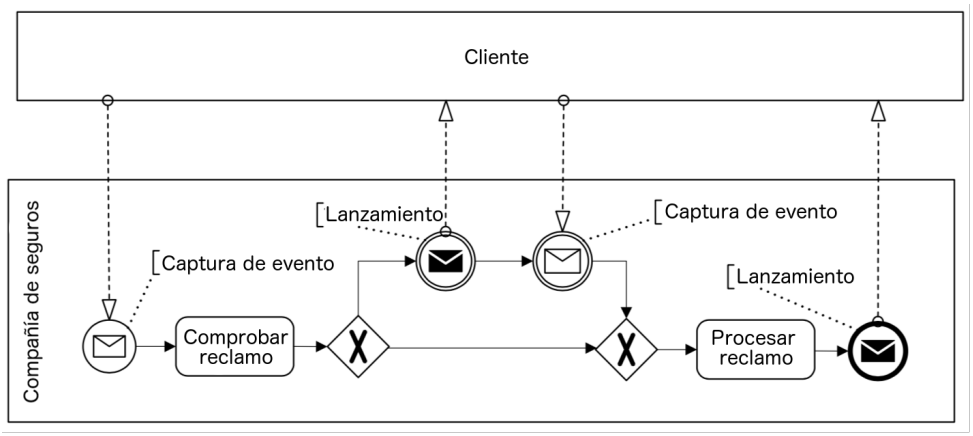

Figura A.4: Ejemplo de eventos de captura y lanzamiento

Otro tipo de evento utilizado muy frecuentemente en diagramas de procesos es el evento de tipo **temporizador**. Pueden representar intervalos de tiempo, puntos en el tiempo, y temporizadores, como los relojes de cuenta regresiva.

Un ejemplo de temporizador se puede ver en la siguiente figura. El proceso inicia con un evento de tipo temporizador. Por la anotación nos damos cuenta de que el proceso se instancia cada primer lunes de Octubre. Cuando este evento es capturado, se anuncia una reunión de estrategia. El siguiente evento temporizador es un reloj de cuenta regresiva. Luego, durante la ejecución del subproceso central, vemos dos temporizadores adicionales, uno de 12 horas que interrumpirá al subproceso si éste no finaliza en menos de ese tiempo, y uno que se ejecutará cada 5 horas sin interrumpir al subproceso.

En BPMN, el **flujo de control** se llama flujo de secuencia. Éste es representado por flechas sólidas entre objetos de flujo, es decir, actividades, eventos y compuertas. BPMN soporta varios tipos de flujos de secuencia, incluyendo flujo normal, flujo condicional, flujo predeterminado y flujo de excepción.

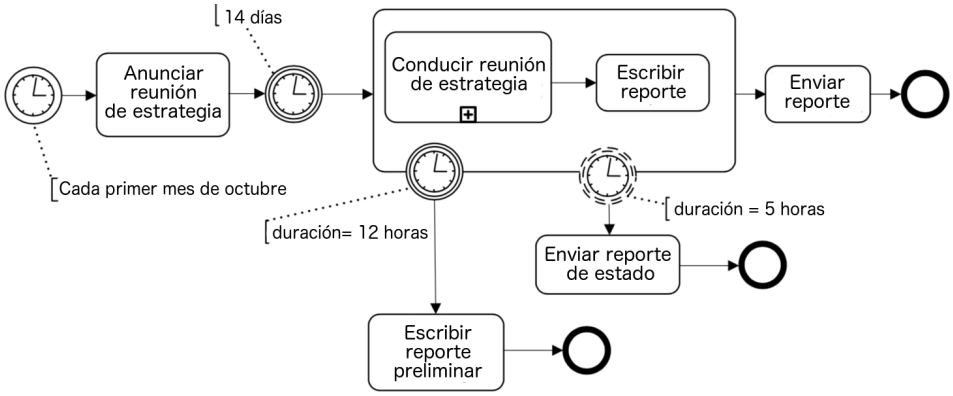

Figura A.5: Diagrama de proceso con eventos temporizadores de interrupción y de no interrupción

El flujo normal de un proceso de negocio representa el comportamiento esperado y deseado del proceso. Empieza con el evento de inicio de un diagrama de proceso y continúa a través de un conjunto de objetos de flujo hasta que alcanza un evento de fin.

En BPMN, cada compuerta actúa como un nodo de unión o un nodo de división. Los nodos de unión tienen al menos dos entradas y una salida. Los nodos de división tienen exactamente una entrada y al menos dos salidas.

Las **compuertas paralelas** representan el típico comportamiento de división y unión. Cada actividad conectada a una compuerta inclusiva es ejecutada en el flujo del proceso.

Las **compuertas exclusivas** eligen una rama basándose en condiciones. El estándar define que las condiciones se evalúan en orden. La primera condición en evaluarse verdadera es la elegida.

La **compuerta inclusiva** expone un comportamiento muy flexible ya que suscribe y extiende el comportamiento tanto de las compuertas exclusivas como las compuertas paralelas. Las compuertas inclusivas se utilizan cuando un número arbitrario de ramas salientes debe ser seleccionado.

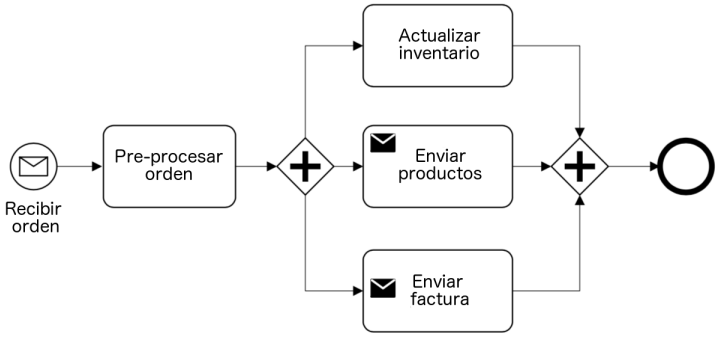

Figura A.6: Compuerta paralela

Para cada construcción del lenguaje hay una semántica. Por ejemplo una flujo de secuencia entre dos actividades restringe su orden de ejecución. Otro ejemplo es que luego de una puerta exclusiva exactamente una opción será elegida.

Por último, un proceso es instanciado cuando un evento de inicio ocurre. Puede haber múltiples eventos de inicio para un proceso dado.

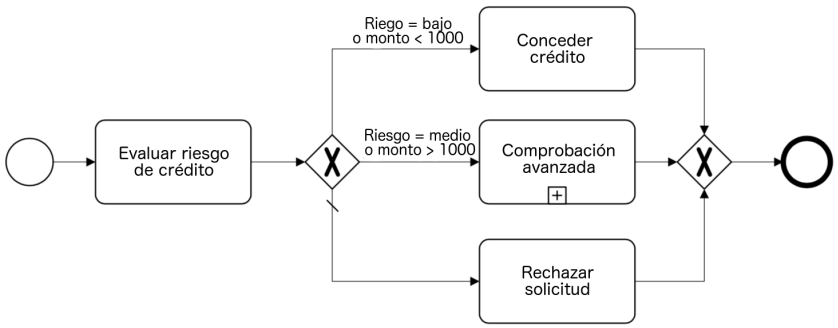

Figura A.7: Compuerta exclusiva

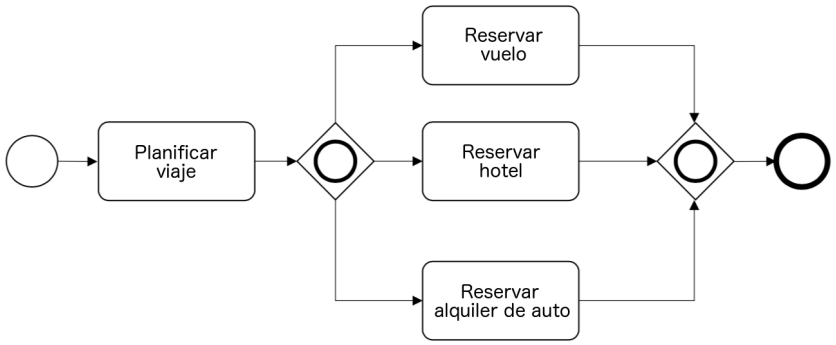

Figura A.8: Compuerta inclusiva

Coreografía de procesos

BPMN está preparado para expresar múltiples organizaciones cuyos procesos de negocio colaboran entre sí. Como fue mencionado previamente, los pools representan participantes de un proceso. Las sendas representan entidades organizacionales dentro de los participantes. Los flujos de secuencia sólo están permitidos dentro de los procesos, es decir, entre nodos que residan dentro de un mismo pool. Por lo tanto, los flujos de secuencia pueden cruzar los límites de una senda, pero nunca pueden cruzar los de un pool. La comunicación entre procesos puede ocurrir sólo con flujo de mensajes.

Esto tiene un significado fundamental para la representación de procesos. Cada organización define la secuencia y el flujo de ejecución para las actividades de sus procesos, pero no puede influir sobre la estructura de los procesos de las demás organizaciones o entidades. Sólamente se puede enviar un mensaje para influenciar en procesos ajenos. Por lo tanto, la comunicación negocio a negocio se da siempre a través de mensajes. Esto nos permite introducir el segundo tema central de este anexo: **Coreografía de procesos**.

Así como la orquestación se ocupa de la definición de las reglas y restricciones para la ejecución de actividades dentro de un proceso de negocio, la coreografía se ocupa de la interacción y colaboración entre distintas orquestaciones de procesos, o en otras palabras, entre distintos procesos de negocio.

En el escenario actual de los negocios, las compañías frecuentemente unen fuerzas para combinar sus servicios y productos de forma tal que sean capaces de proveer valor agregado al mercado. Los productos y servicios son fruto de procesos de negocio, que en muchos casos toman ventaja de infraestructuras de software preexistentes de las compañías que participan en el dominio.

Si los requerimientos de coreografía son simples, probablemente sea suficiente con una interacción

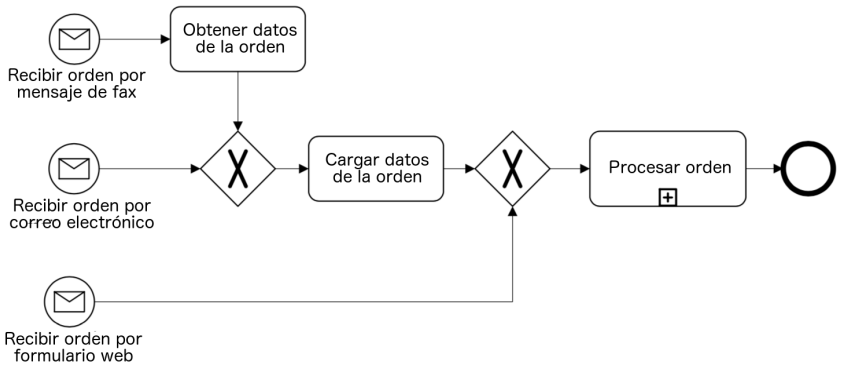

Figura A.9: Diagrama de proceso con múltiples eventos de inicio alternativos

humana entre organizaciones intercambiando Faxes, mails, o llamados telefónicos. Por otro lado, si se quisiera alcanzar un alto nivel de automatización, es necesario especificar modelos específicos que detallen la naturaleza de las relaciones de colaboración entre negocios pares.

La coreografía de procesos se diseña a través de un proceso complejo que involucra varias etapas de análisis del dominio, de escenarios, de participantes y de mensajes, una etapa de definición de la coreografía en sí, y la implementación de los mecanismos de colaboración.

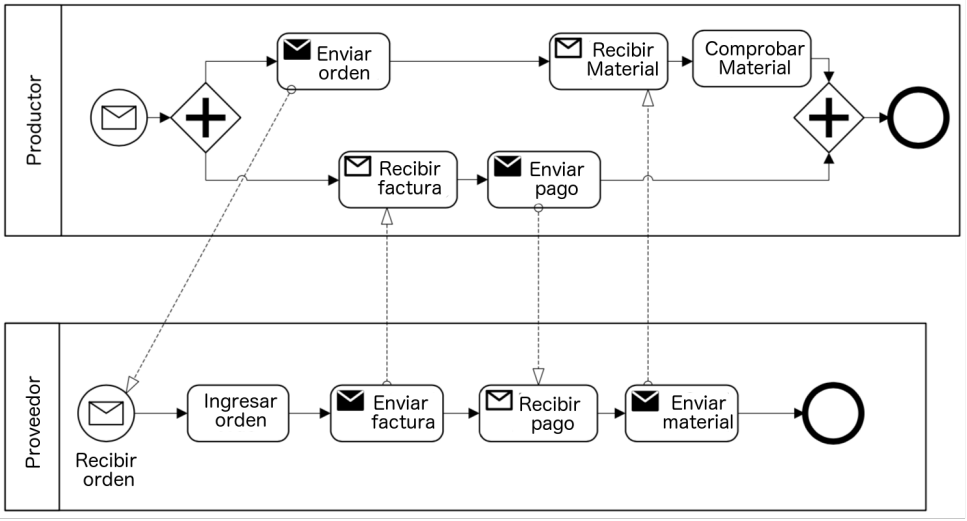

Figura A.10: Coreografía de procesos donde dos entidades colaboran entre sí

BPMN permite representar gráficamente la interacción de servicios entre distintos procesos. Podemos identificar varios patrones que se repiten en la coreografía de procesos.

El **envío** es un patrón que representa una interacción de una sola dirección entre dos participantes, vista desde la perspectiva del remitente. El receptor del mensaje puede conocerse desde el diseño del proceso, o determinarse dinámicamente durante la ejecución.

El **patrón de recepción** también describe una interacción de una sola dirección entre dos participantes, pero está visto desde la perspectiva del receptor. Se pueden distinguir dos casos respecto al comportamiento de la recepción de mensajes del receptor. Los mensajes que no son esperados pueden ser descartados o almacenados para ser procesados más tarde.

En el patrón de **envío/recepción**, un participante envía un pedido a otro, el cual retorna un mensaje de respuesta. Ambos mensajes pertenecen a la misma conversación.

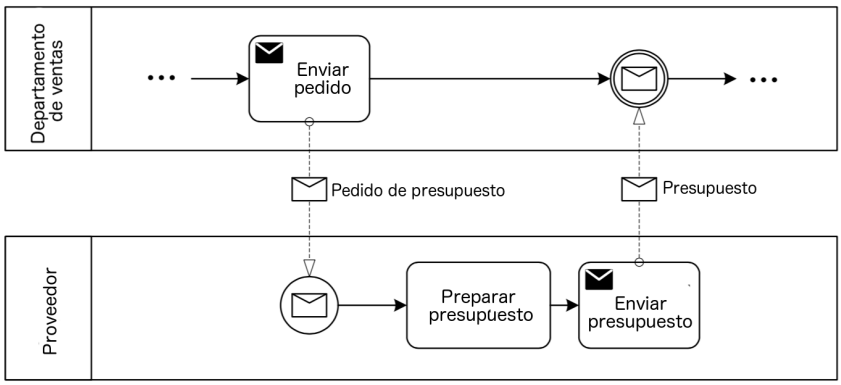

Figura A.11: Patrón de envío/recepción

El **envío de uno a varios** es un patrón en el cual se envían múltiples mensajes en paralelo a varios participantes. La lista de destinatarios puede conocerse desde el diseño de los procesos y la coreografía o puede ser una selección dinámica durante la ejecución. Cuando un mensaje se envía a múltiples instancias de otro proceso, el símbolo de paralelismo de BPMN se utiliza tanto en la tarea que envía el mensaje como en el pool del proceso receptor.

El patrón de **recepción de varios a uno** representa la recepción de mensajes de múltiples remitentes. Un único participante espera la recepción de mensajes de otros participantes, y cada uno de éstos puede enviarle exactamente un mensaje. Típicamente, el receptor no conoce el número de mensajes que va a recibir.

Por último, el patrón de **envío/recepción de uno a varios** representa el envío de varios mensajes a distintos participantes y la espera de sus respuestas. Normalmente no es necesario esperar por todas las respuestas.

Referencias

[1] M. Andrews and J. A. Whittaker. Computer security. *IEEE Security Privacy*, 2(5):68–71, Sept 2004.

[2] Elaine B. Barker, William C. Barker, William E. Burr, W. Timothy Polk, and Miles E. Smid. Sp 800-57. recommendation for key management, part 1: General (revised). Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2007.

[3] Digital Bazaar. Forge Website, Aug 2018. `https://github.com/digitalbazaar/forge` [Online; accedido 16 de Septiembre de 2018].

[4] E. Bertino, J. Crampton, and F. Paci. Access control and authorization constraints for ws-bpel. In *2006 IEEE International Conference on Web Services (ICWS'06)*, pages 275–284, Sept 2006.

[5] Bonitasoft. Bonita BPM Documentation. `https://documentation.bonitasoft.com/bonita/7.4/` [Online; accedido 16 de Septiembre de 2018].

[6] BouncyCastle.org. Bouncy Castle Website. `https://www.bouncycastle.org/java.html` [Online; accedido 16 de Septiembre de 2018].

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008. `http://www.rfc-editor.org/rfc/rfc5280.txt` [Online; accedido 16 de Septiembre de 2018].

[8] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer Publishing Company, Incorporated, 2013.

[9] M. Menzel, I. Thomas, and C. Meinel. Security requirements specification in service-oriented business process management. In *2009 International Conference on Availability, Reliability and Security*, pages 41–48, March 2009.

[10] J. Muller and K. Bohm. The architecture of a secure business-process-management system in service-oriented environments. In *2011 IEEE Ninth European Conference on Web Services*, pages 49–56, Sept 2011.

[11] J. Mülle, S. von Stackelberg, and K. Böhm. Modelling and transforming security constraints in privacy-aware business processes. In *2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 1–4, Dec 2011.

[12] Oracle. Generate Public and Private keys. `https://docs.oracle.com/javase/tutorial/security/apisign/step2.html` [Online; accedido 16 de Septiembre de 2018].

[13] IBM Redbooks. *IBM Business Process Manager Security: Concepts and Guidance*. Vervante, 2012.

[14] Eric Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley Professional, 2000.

[15] William Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2010.

[16] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, Berlin, Heidelberg, 2007.

[17] J.K. Wood, J. Engelke, and IBM Redbooks. *IBM Business Process Manager Security: Concepts and Guidance*. IBM redbooks. IBM Redbooks, 2012.