# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## FACULTAD DE INGENIERÍA

# DISEÑO DE UNA ARQUITECTURA SOA APLICADA EN UN SISTEMA DE CONSULTORIO MÉDICO VIRTUAL

TESIS

QUE PARA OBTENER EL TÍTULO DE

**INGENIERO EN COMPUTACIÓN**


PRESENTA:

**EDUARDO GRANADOS CHAVARRÍA**


DIRECTOR DE TESIS:
M. C. ALEJANDRO VELÁZQUEZ MENA

Ciudad Universitaria, México, D.F., Junio 2014.

Para mis padres Ángel José Granados y Ma. Rosa Chavarría.

Hermanos Iván y Emmanuel.

A mis abuelitos:

José Granados QDEP – Amalia Palomo.

Prisciliano Chavarría – Juana Castañeda.

A mi Alma mater la Universidad Nacional Autónoma de México, Facultad de Ingeniería.

Un agradecimiento muy especial a Roció Arzate y Elizabeth Machuca.

Un saludo muy caluroso a mis amigos:

- Jaqueline Torres
- Gisela Pineda
- Ireri Raya
- Lucely Mata
- Haydeé Gómez
- César Vélez
- Yesica Hernández
- Fabiola Zarate
- Andrés Hernández

Colegas y amigos de Think and code:

- Eduardo Villegas
- Luis Nava
- Alejandro Hernández

# INDICE

# INDICE DE TABLAS

# INDICE DE ILUSTRACIONES

# INTRODUCCIÓN

En más de una ocasión me he encontrado con líderes de proyecto o arquitectos de software que cometen el error de decir que una arquitectura SOA forzosamente se trata de desarrollar servicios web y/o utilizar ciertas herramientas disponibles en el mercado, sin comprender su naturaleza de servicio. Es por eso que decidí realizar el presente trabajo de tesis, que tiene como objetivo ilustrar fácilmente qué y cuáles son las características de este tipo de arquitectura, sin requerir el uso de herramientas sofisticadas. Adicionalmente, se identifican las ventajas de ésta y su ámbito de aplicación, dejando de lado los argumentos sobre el uso forzoso de ciertas tecnologías o productos. Posteriormente se implementa la metodología para realizar el análisis, diseño y construcción de un sistema de administración de consultorios médicos, llamado “Sistema de Consultorio Médico Virtual” (SCMV).

Este trabajo se divide en cuatro grandes partes:

Parte 1: Consta del Capítulo 1 donde se explica qué es arquitectura de software y se identifican buenas prácticas aplicables en su formulación. Se define el concepto de Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés), explicándose los principios en los que se sustentan los servicios asociados.

Parte 2: Abarca los Capítulos 2, 3 y 4. En el Capítulo 2 se explica una metodología genérica para SOA llamada MSOAM, Mainstream SOA Methodology, que puede ser adaptada a las necesidades específicas de un proyecto dado. En el Capítulo 3 se abarca un paso de la metodología llamado “Análisis Orientado a Servicios”. En el Capítulo 4 se habla del paso “Diseño Orientado a Servicios”.

Parte 3: Comprende de los capítulos 5, 6 y 7. En el Capítulo 5 se menciona el alcance y requerimientos que debe tener el SCMV. Tanto el Análisis Orientado a Servicios, como el Diseño Orientado a Servicios se ponen en práctica en el ciclo de desarrollo del SCMV en los capítulos 6 y 7 respectivamente.

Parte 4: En el Capítulo 8 se explica cómo fue construido el SCMV siguiendo la metodología MSOAM. Se detallan las tecnologías utilizadas en el desarrollo, seleccionadas siguiendo criterios de robustez y simplicidad.

# CAPÍTULO 1. MARCO TEÓRICO

Por Arquitectura de software, se conoce al diseño de más alto nivel de una aplicación, la palabra “arquitectura” viene del griego αρχ que significa “jefe(a), quien tiene el mando”, y de τεκτων que significa constructor o carpintero, dicho de otra manera, el arquitecto de software es quien dirige el desarrollo del software.

En los siguientes subcapítulos se desarrollará el concepto, analizando los factores que intervienen en esta actividad y apuntando a las buenas prácticas que la acompañan.

## 1.1 Arquitectura de Software

Existen varias definiciones del término Arquitectura de Software, aunque se enfocan en distintos aspectos de la actividad, puede considerarse que se orientan en la misma dirección, y por lo tanto, son complementarias.

“La arquitectura de software de un programa o sistema de cómputo es una representación del mismo que ayuda a la comprensión de su comportamiento.

La arquitectura de software sirve como modelo para el sistema y el desarrollo del proyecto, define las asignaciones de trabajo que deben ser llevadas a cabo por los equipos de diseño e implementación. La arquitectura es el portador primario de las cualidades del sistema, tales como rendimiento, mantenimiento, y seguridad, ninguna de las cuales se puede lograr sin una visión arquitectónica unificadora. La arquitectura como artefacto facilita el análisis temprano, buscando que un paradigma de solución se concrete en un sistema que cumpla con los requerimientos establecidos. Cuando definida correctamente, la arquitectura permite identificar riesgos de diseño y por lo tanto tomar acciones para mitigarlos en etapas tempranas del proceso de desarrollo” [5].

La arquitectura comienza por comprender las especificaciones del sistema con base en el QUÉ es lo que se desea desarrollar. Posteriormente da una solución al CÓMO lograrlo, dictando las tareas que realizarán los equipos de trabajo, así como las estructuras físicas y lógicas del sistema. Además establece las especificaciones de diseño y las de construcción del sistema. Por lo que la arquitectura es determinante para lograr un sistema funcional.

La arquitectura establece también la especificación de los elementos internos, su comportamiento y forma de comunicación. Asimismo, en caso de interactuar con entidades externas, define la manera en que estas interacciones ocurrirán.

## 1.2 Factores que guían el desarrollo de la arquitectura

Durante el desarrollo arquitectónico intervienen muchos factores, algunos de ellos son: los grupos de interés, las necesidades de negocio, las limitaciones técnicas, la experiencia del equipo de trabajo (en especial del arquitecto), etc. Estos factores establecen restricciones a la solución arquitectónica, al mismo tiempo que definen cualidades requeridas en el sistema, tales como: modificabilidad, eficiencia, mantenibilidad, interoperabilidad, confiabilidad, reusabilidad, facilidad de ejecución de pruebas, etc.

| Factores de intereses | |
|---|---|
| **Grupo de interés** | **Interés** |
| Cliente | Bajo costo, entrega oportuna |
| Usuario final | Rendimiento, seguridad, usabilidad, redituable |
| Mantenimiento | Modificable |
| Líderes de proyecto | Entregas oportunas |
| Hardware | Capacidad de procesamiento, interfaces |
| Sistemas externos/legados | Interfaces establecidas |

**Tabla 1-1 Factores de intereses.**

La arquitectura puede recibir retroalimentación de los mismos grupos de interés, además de ayudar a reforzar la experiencia del arquitecto, es posible que se reconsideren decisiones previas en busca de posibles debilidades y, por ende, agregar nueva infraestructura, también pueden identificarse elementos reutilizables.

En algunos casos los usuarios del sistema pueden realizar recomendaciones para que éste sea más fácil y eficiente sin descuidar los requerimientos ya establecidos.

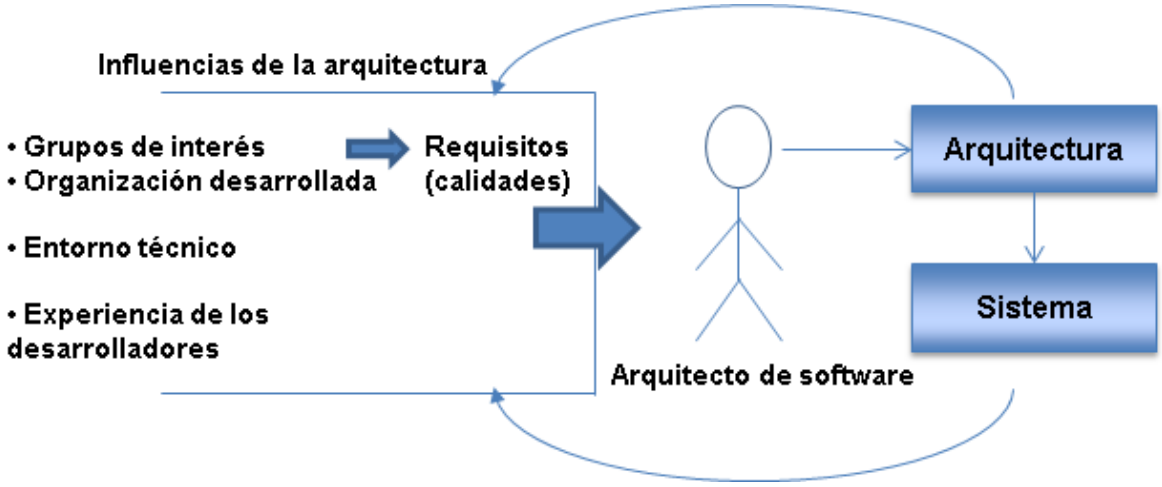


Ilustración 1-1 Influencias de la arquitectura.

### 1.2.1 Atributos de calidad en la arquitectura

Para poder decir que el diseño de la arquitectura es adecuado para el sistema, cubriendo en la medida de lo posible todas las necesidades, se debe realizar una evaluación de sus atributos de calidad, que son requerimientos adicionales a los funcionales. Estos atributos están fuertemente entrelazados y trabajan en conjunto, generalmente no es posible cumplir con todos al 100% o beneficiar a uno sin afectar a otro. La arquitectura debe alcanzar un punto en el que se cumplan todos los atributos de manera aceptable.

Los atributos de calidad son clasificados en:

- Requisitos funcionales
- Requisitos no funcionales

#### 1.2.1.1 Requisitos funcionales

Estos atributos se ven reflejados en tiempo de ejecución del sistema

- Rendimiento
  Es la capacidad de respuesta para ejecutar una acción en un determinado tiempo, por ejemplo, la cantidad de información o transacciones que pueden ser procesadas. Aquí se ven involucrados los algoritmos implementados, la capacidad de la red para sistemas distribuidos.
- Seguridad interna
  Se refiere a la capacidad del sistema para evitar el uso indebido o alteración de los servicios por usuarios no autorizados.

- Confidencialidad

  Se encarga de que los datos o servicios estén protegidos contra el acceso no permitido.

- Integridad

  Procura que los datos o servicios sean entregados bajo las condiciones acordadas.

- Garantía

  En una transacción asegura que la identificación de las partes involucradas sea verificada.

- Auditoría

  Es la propiedad que capacita al sistema para llevar un seguimiento de las actividades realizadas, esto permite la reconstrucción de lo sucedido.

- Seguridad externa

  Este atributo se encarga de disminuir daños a la información y al sistema que pudieran ocasionar eventos externos no tecnológicos, por ejemplo: interrupciones eléctricas, incendios, derrumbes, etc.

- Disponibilidad y fiabilidad

  La disponibilidad es la propiedad de que el sistema esté en correcto funcionamiento, fiabilidad es el tiempo medio entre fallos, la disponibilidad puede evaluarse a partir del siguiente modelo:

$$Disponibilidad = \frac{tiempo\ \ medio\ \ entre\ \ fallos}{tiempo\ \ medio\ \ entre\ \ fallos\ +\ tiempo\ \ medio\ \ de\ \ reparación}$$

- Funcionalidad

  Es la habilidad para realizar de forma correcta las tareas asignadas, aquí se involucra tanto la asignación de cargas de trabajo como la correcta operación de los datos.

- Usabilidad

  Es la facilidad y comodidad que el sistema ofrece a los usuarios para su uso, además de los apoyos que ofrezca.

  - Facilidad de aprender las características del sistema e interfaz de usuario
  - Uso eficiente del sistema aprovechando sus capacidades
  - Disminución de los errores que puedan ocurrir y ayuda para corregirlos
  - Adaptación del sistema a las necesidades del usuario
  - Incremento de la confianza y satisfacción

#### 1.2.1.2 Requisitos no funcionales

Atributos no apreciables en tiempo de ejecución.

- Modificabilidad y mantenibilidad

  Es la habilidad del sistema para realizar cambios en sí mismo, módulos, componentes, relaciones sin que estos afecten a otras secciones y al mismo tiempo no sean muy costosos o requieran una gran inversión de tiempo.

- Portabilidad

  Es la capacidad de un sistema para ser ejecutado en distintos ambientes, software o hardware. Por ejemplo, un servidor de aplicación de otra compañía, una actualización del servidor o del sistema operativo.

- Reusabilidad

  Esta capacidad procura que los distintos componentes del sistema puedan ser utilizados por otros componentes del mismo, incluso nuevos sistemas, con el fin de ahorrar tiempos de construcción y de inversión.

- Integridad

  Esta capacidad se refiere a la correcta funcionalidad y coordinación de los componentes, donde cada uno tiene asignada cierta responsabilidad para comunicarse a través de interfaces conocidas por los mismos.

- Facilidad de ejecución de pruebas

  Es la capacidad del sistema sea probado por secciones y, en caso de que ocurra un fallo, éste pueda ser localizado.

### 1.2.2 Recomendaciones para el desarrollo de la arquitectura

A continuación se listan algunas recomendaciones para el desarrollo de una arquitectura, no son reglas obligatorias o absolutas pero pueden considerarse como una buena práctica, se dividen en dos grupos.

Recomendaciones de proceso.

- La arquitectura debe ser desarrollada por un arquitecto o un grupo pequeño de arquitectos (designando un líder).
- Listar los requerimientos funcionales y priorizar los atributos de calidad deseados.

- Documentar la arquitectura, al menos una visión estática y una dinámica, que pueda ser interpretada por los involucrados en el sistema.
- La arquitectura debe revisarse por las partes interesadas en el sistema.
- La arquitectura se debe revisar con un análisis cuantitativo, por ejemplo el rendimiento máximo, y evaluar los atributos de calidad.
- El diseño arquitectónico debe procurar ser incremental para facilitar la integración de futuros elementos.

Se deben especificar áreas de recursos limitados, por ejemplo, si la utilización de la red es un área de preocupación, hay que proporcionar guías al equipo de desarrollo para que hagan el menor uso posible de los recursos de red, estas guías deberán especificar como lograr un desarrollo correcto y eficiente.

Recomendaciones estructurales.

- La arquitectura debe tener módulos bien definidos que cumplan con los principios de separación de responsabilidades y encapsulamiento de la información.
- Los módulos deben contar con interfaces bien definidas, ocultando la implementación, con el fin de que los equipos trabajen de forma independiente.
- Se deben aplicar tácticas arquitectónicas que puedan ayudar a satisfacer los atributos de calidad.
- El diseño no debe depender de una herramienta específica o versión, de otro modo hay que procurar que el cambio de un producto sea lo más transparente posible.
- Separar los módulos que producen datos de los que los consumen. Se reduce la posibilidad de que la modificación de un módulo de bajo nivel tenga un impacto en los módulos que lo utilizan.
- En procesamientos paralelos, hay que identificar y analizar los puntos críticos.
- Los procesos y tareas deben ser documentados, de tal forma que si hay la necesidad de realizar cambios, estos puedan ser ubicados y realizados.
- La arquitectura debe tener un número pequeño de patrones de interacción, es decir, se deben hacer las mismas cosas de la misma forma en todas partes. Esto demuestra un diseño integral, mejorando la legibilidad, el tiempo de desarrollo, la fiabilidad, y mantenimiento del sistema.

### 1.2.3 Patrón de arquitectura

Describe un problema que ocurre frecuentemente en un entorno. Para resolverlo es necesario describir el núcleo de la solución del problema, de tal manera que ésta pueda ser utilizada un sinnúmero de veces sin requerir de un rediseño.

Los patrones deben contener cuatro secciones fundamentales, nombre del patrón, problema, solución y consecuencias. Esto favorece la reutilización de código, ahorro de tiempo y disminución de posibles problemas.

El incremento en el conocimiento de los patrones permite realizar agrupaciones de patrones sólidos, llamados *lenguajes de patrones*. Los lenguajes de patrones definen los estilos arquitectónicos con los que se forman sistemas completos y robustos.

### 1.2.4 Estilo de arquitectura

Es un conjunto de componentes y patrones que definen la transferencia y control de la información, también dicta los tipos de componentes que pueden utilizarse y su interacción. El estilo de la arquitectura es una vista a un nivel más abstracto que los patrones.

### 1.2.5 Modelo de referencia

No se trata propiamente de una arquitectura. Es una descomposición estándar de un problema en unidades funcionales que resuelven una tarea específica: cuando interactúan conjuntamente resuelven el problema mayor.

### 1.2.6 Arquitectura de referencia

Esta puede verse como la aplicación de un modelo de referencia en los componentes de software, no necesariamente con una relación uno a uno. La arquitectura de referencia define la infraestructura común a todos los sistemas del dominio (componentes, subsistemas, interfaces).

El considerar una arquitectura de referencia facilita el desarrollo de nuevos sistemas, debido a que se integran guías de cómo desarrollar, facilitando la reutilización de modelos y componentes.

## 1.3 Arquitectura Orientada a Servicios, SOA

SOA nace a partir de la necesidad de desarrollar sistemas que no tengan una alta dependencia de la infraestructura tecnológica o de algún otro sistema. Realizar cambios en un sistema para adaptar nueva funcionalidad, realizar modificaciones de negocio o cambiar componentes tiene un elevado costo, tanto económicamente, como de tiempo y recursos.

SOA es un estilo de arquitectura de software basado en la operación conjunta y coordinada de servicios, los cuales son la base de este modelo de arquitectura.

Un servicio es la encapsulación de una funcionalidad de negocio bien definida, diseñado de tal manera que pueda ser reutilizado con facilidad. Un servicio a su vez puede estar formado por otros servicios, trabajar de forma distribuida abstrayendo la complejidad asociada a las comunicaciones, incluso puede estar conformado por una o más operaciones de más bajo nivel.

Ejemplos de servicios pueden ser la verificación del saldo de una cuenta, el envío de un mensaje o la validación de una clave CURP.


**Ilustración 1-2 Representación de un servicio**

SOA tiene como objetivo el desarrollo de sistemas que tengan la facilidad y la rapidez para soportar cambios tecnológicos o de lógica de negocio, incluso ser capaces de integrarse con sistemas externos o legados, estas características dan como ventaja el ahorro de tiempo y reducción de costos.

Los servicios cuentan con una interfaz en la que describen los parámetros de entrada y de salida, por lo que la implementación queda oculta. La comunicación entre ellos se debe realizar a través del intercambio de mensajes estandarizados con el fin de disminuir el acoplamiento entre los mismos.

En muchas ocasiones cuando se habla de SOA, por error suele relacionarsele inmediatamente con servicios web (web-services), si bien esta tecnología brinda un gran soporte para el desarrollo de servicios distribuidos con una implementación final oculta, no son estríctamente necesarios para desarrollar una SOA.

Es posible hacer SOA con alguna otra tecnología de comunicación remota o incluso dentro del mismo sistema se puede tener servicios si cumplen con las características y con los principios de diseño que se mencionan a continuación.

## 1.4 Principios de diseño de servicios

El diseño de una SOA tiene como objetivos ser robusta, escalable, integra y reutilizable. Para lograr dichos objetivos es buena práctica que los servicios que la componen estén diseñados

siguiendo ciertos principios. Aunque estos principios no son obligatorios sí es recomendable seguirlos en la medida de lo posible.

Los primeros cuatro son fundamentales, los últimos derivan indirectamente de los primeros:

- Los servicios son reusables
- Los servicios comparten un contrato formal
- Los servicios están débilmente acoplados
- Los servicios son compuestos
- Los servicios son autónomos
- Los servicios deben ser abstractos
- Los servicios no guardan estado
- Los servicios deben poder ser descubiertos

### 1.4.1 Los servicios son reusables

Los servicios tienen y expresan una lógica agnóstica con lo que pueden ser utilizados por múltiples servicios y/o sistemas.

**Objetivos**

- ✓ Permitir la adecuación rápida y de bajo costo a las nuevas necesidades del negocio
- ✓ Generar un catálogo de servicios genéricos, capaces de ser utilizados por distintos clientes

**Características**

- ✓ Su diseño debe ser agnóstico a la tecnología
- ✓ Su lógica debe ser genérica y puntual, de modo que pueda ser utilizada en distintos escenarios, servicios o sistemas

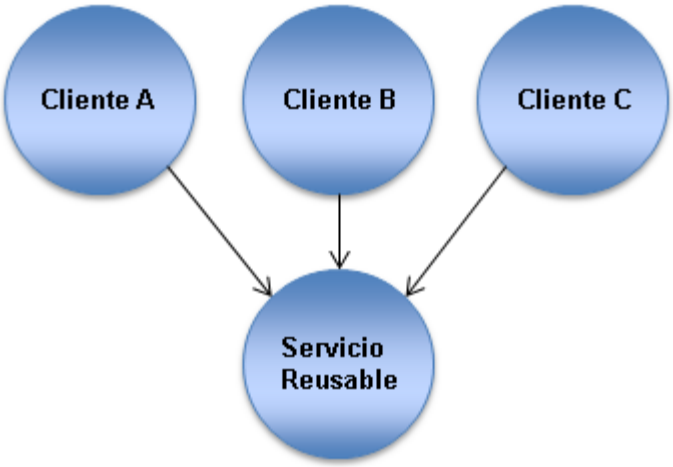


Ilustración 1-3 Principio de reusabilidad.

- ✓ Deben poder ser consumidos de forma simultánea
- ✓ Deben tener contratos genéricos capaces de procesar una amplia gama de entradas y salidas

**Ejemplo**

Un sistema empresarial tiene el servicio de administración de personal el cual tiene operaciones para crear, dar de baja, consultar y modificación del personal, este servicio se presta a ser altamente reusable por ejemplo, si dentro de la organización hay un sistema la impartición de cursos al personal: se poder realizar una integración con el servicio de Administración personal de modo que se tenga la información de las personas y obtener de ellas cierta información cómo podría ser, departamento o gerencia, puesto, puesto, entre otros.

### 1.4.2 Los servicios comparten un contrato formal

Un contrato de servicio es la especificación del servicio, en el cual se debe mencionar:

Cómo debe ser consumido

Su ubicación

Reglas y características de las operaciones (métodos) que ofrece, entre ellas los mensajes de entrada y de salida

**Objetivos**

- ✓ Proporcionar la información necesaria para que el servicio pueda ser utilizado de forma correcta

**Características**

- ✓ Debe procurar usar estándares de comunicación
- ✓ Debe estar debidamente documentado:
- ✓ Nombre del servicio
- ✓ Propietario del servicio
- ✓ Tipo de servicio
- ✓ Ubicación
- ✓ Fecha de liberación
- ✓ Versión
- ✓ Dominio
- ✓ Propósito
- ✓ Operaciones del servicio
- ✓ Mensaje de entrada
- ✓ Mensaje de salida
- ✓ Etc.

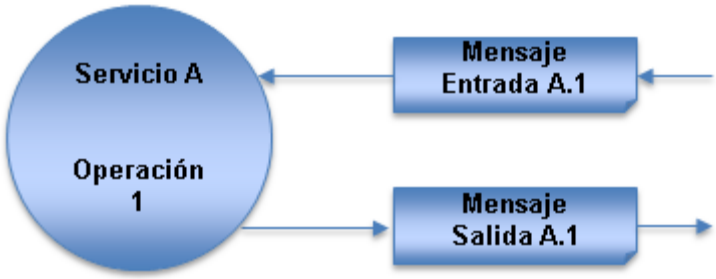


Ilustración 1-4 Principio de contrato formal

**Ejemplo**

Contrato para un servicio que genera el CURP.

| SERVICIO GENERACIÓN DE CURP. | |
|---|---|
| Propietario del servicio | Área de RH |
| Tipo de servicio | Utilitario |

| | | | | |
|---|---|---|---|---|
| Ubicación | http://empresa.com.mx/servicios/utileria/curp | | | |
| Fecha de liberación | 10/01/2013 | | | |
| Versión | 1.0 | | | |
| Descripción | Obtener la clave del CURP de una persona | | | |
| **Operación** | **Obtener CURP** | | | |
| **Mensaje de entrada** | **Campo** | **Tipo de dato** | **Tamaño** | **Obligatorio** |
| | Apellido paterno | Alfanumérico | 40 | Sí |
| | Apellido materno | Alfanumérico | 40 | Sí |
| | Nombre | Alfanumérico | 40 | Sí |
| | Fecha de nacimiento | Numérico<br>Formato (AAAADDMM) | 8 | Sí |
| | Género | Alfanumérico<br>Femenino: F<br>Masculino: M | 1 | Sí |
| | Entidad de Nacimiento<br>Clave conforme al catálogo de Claves Alfanuméricas de Entidades Federativas | Alfanumérico | 2 | Sí |
| **Mensaje de salida** | **Campo** | **Tipo de dato** | **Tamaño** | **Obligatorio** |
| | CURP | Alfanumérico | 18 | Sí |

**Tabla 1-2 Petición de CURP.**

| CLAVES ALFANUMÉRICAS DE ENTIDADES FEDERATIVAS | | | | | |
|---|---|---|---|---|---|
| ESTADO | CLAVE | ESTADO | CLAVE | ESTADO | CLAVE |
| AGUASCALIENTES | AS | GUERRERO | GR | QUINTANA ROO | QR |
| BAJA CALIFORNIA | BC | HIDALGO | HG | SAN LUIS POTOSI | SP |
| BAJA CALIFORNIA SUR | BS | JALISCO | JC | SINALOA | SL |
| CAMPECHE | CC | MÉXICO | MC | SONORA | SR |
| COAHUILA | CL | MICHOACAN | MN | TABASCO | TC |
| COLIMA | CM | MORELOS | MS | TAMAULIPAS | TS |
| CHIAPAS | CS | NAYARIT | NT | TLAXCALA | TL |
| CHIHUHUA | CH | NUEVO LEON | NL | VERACRUZ | VZ |
| DISTRITO FEDERAL | DF | OAXACA | OC | YUCATÁN | YN |
| DURANGO | DG | PUEBLA | PL | ZACATECAS | ZS |
| GUANAJUATO | GT | QUERETARO | QT | NACIDO EN EL EXTRANJERO | NE |

**Tabla 1-3 Claves de entidades federativas**

### 1.4.3 Los servicios están débilmente acoplados

Los servicios deben idealmente ser capaces de adaptarse a cambios en sus dependencias en el menor tiempo posible, a bajo costo y de manera transparente.

El bajo acoplamiento significa tener poca dependencia entre servicios, de modo que si un servicio sufre un cambio o es sustituido por otro, los servicios dependientes se vean afectados lo menos posible teniendo los mismos o mejores resultados.

**Objetivos**

- ✓ Disminuir las dependencias fuertes entre servicios
- ✓ Poder realizar cambios a bajo costo y en el menor tiempo posible

**Características**

- ✓ Para lograr bajo acoplamiento las llamadas a los servicios se hacen a través de interfaces, de este modo la implementación queda oculta, de modo que si se realiza un cambio, el servicio cliente no se vea afectado

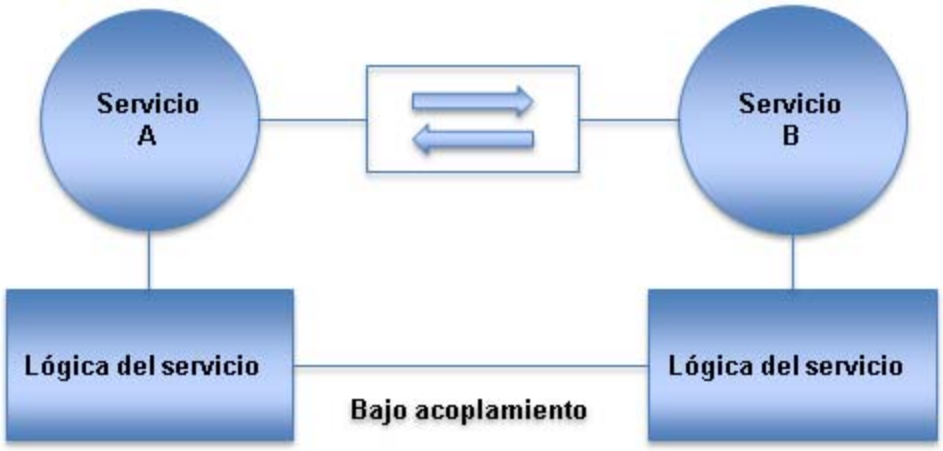


Ilustración 1-5 Principio de bajo acoplamiento.

**Ejemplo**

Un sistema que depende de un servicio de envío de correo electrónico hace los envíos de correos a través de Yahoo! con el dominio de la empresa, sin embargo, con el pasar de los años encontraron que Google ofrecía mejores paquetes para el servicio de correo y por ello deciden migrar su dominio para que puedan enviar el correo con la misma dirección.

El realizar este cambio solo se modifica la lógica del servicio del correo, su contrato y la lógica del sistema que usa este servicio no se ve afectada y puede continuar trabajando como si nada hubiera ocurrido.

### 1.4.4 Los servicios son compuestos

En ocasiones en un servicio puede estar compuesto por otros, de modo este conjunto de servicios pueden encargarse de resolver una tarea más compleja. Este principio puede ser un poco contrario al principio de autonomía explicado más adelante.

**Objetivos**

- ✓ Reducir código
- ✓ Facilitar la ejecución de tareas complejas

**Características**

- ✓ Los servicios que componen el servicio compuesto deben ser suficientemente eficientes para manejar la concurrencia
- ✓ Los contratos de los servicios deben ser flexibles para poder ser consumidos por otros servicios, o intercambiar tipos de datos

**Ejemplo**

Suponiendo que se desea crear una cita al médico, este servicio puede ser compuesto por tres servicios: guardar el registro de la cita, enviar una notificación electrónica con los datos de la cita y guardar los *logs* del proceso.

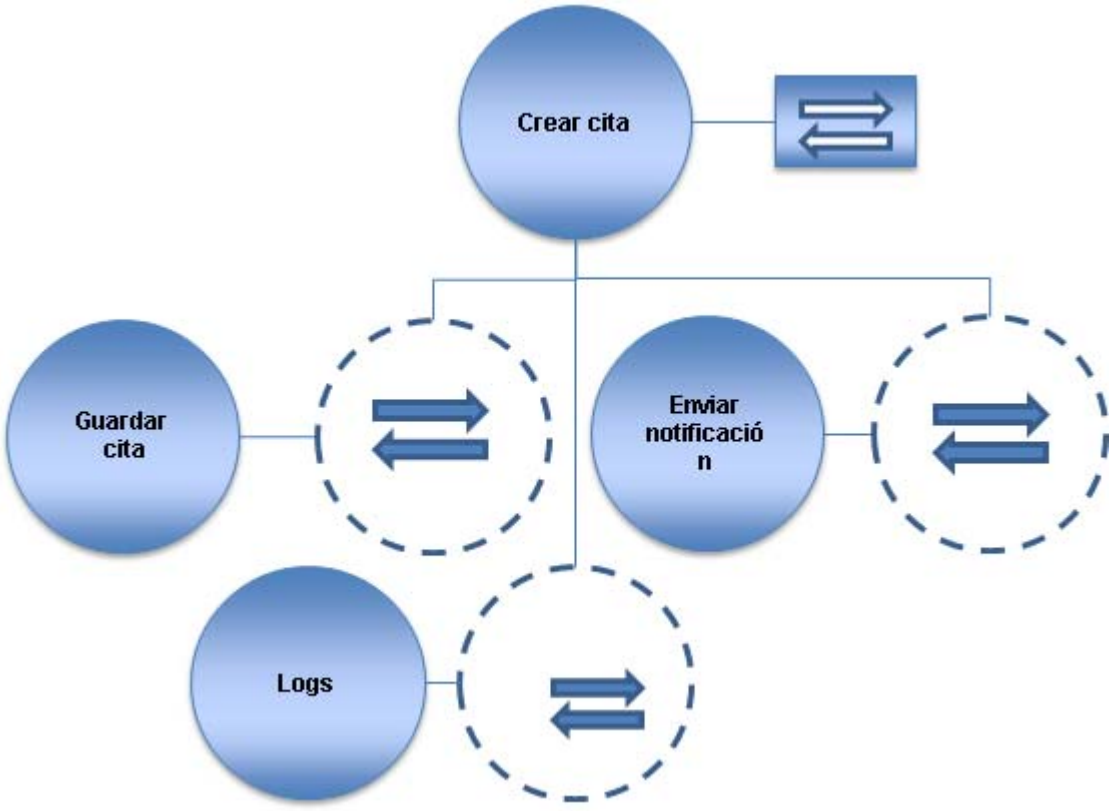


Ilustración 1-6 Principio de composición.

### 1.4.5 Los servicios son autónomos

Se debe procurar que los servicios tengan control de sí mismos para realizar sus tareas sin tener dependencias de factores o actores externos a su entorno.

**Objetivos**

- ✓ Aumentar el rendimiento del servicio
- ✓ Dar mayor autocontrol al servicio

**Características**

- ✓ Los servicios tienen un contrato en el que se determina explícitamente su funcionalidad a la que no puede superponerse otro servicio
- ✓ El ambiente donde los servicios son desplegados debe permitir una alta concurrencia y ser escalable

**Ejemplo**

En una empresa el Departamento de Financiamiento debe cumplir con las siguientes tareas para procesar un financiamiento.

- ✓ Comprobar nombre
- ✓ Verificar antecedentes
- ✓ Verificar ubicación

El Departamento de financiamiento tiene en su poder la información para poder validar el nombre y la ubicación, sin embargo, la revisión de los antecedentes es manejada por el departamento de pagos, los cuales pueden ser consultados por el área de financiamiento, esto implica que la autonomía del Departamento de Financiamiento se ve un poco afectada.

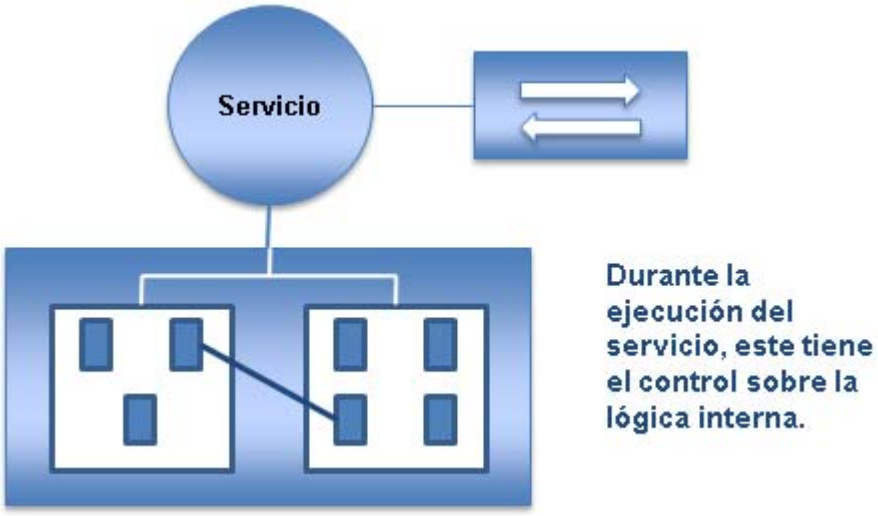


Ilustración 1-7 Principio de autonomía.

### 1.4.6 Los servicios guardan estado

Cada petición al servicio debe ejecutarse de forma independiente a las peticiones previas, no debe guardar información de sesión de quién lo haya ejecutado.

**Objetivos**

- ✓ Incrementar la escalabilidad
- ✓ Aumentar la lógica agnóstica del servicio

**Características**

- ✓ Los servicios deben ser de bajo acoplamiento y altamente independientes a la lógica del negocio
- ✓ Los contratos de los servicios deben ser flexibles capaces de permitir la transmisión de información del estado en tiempo de ejecución

**Ejemplo**

Los servicios no deben guardar un estado o sesión de quien los invoca, esto incrementaría el acoplamiento y por lo tanto la escalabilidad también se vería afectada. El rendimiento de un servicio se ve afectado al tener que estar almacenando una sesión por cada uno de los clientes, el estado que guarden los servicios solo es momentáneo, durante el tiempo en el que entregan una respuesta, después la sesión o estado se pierde.

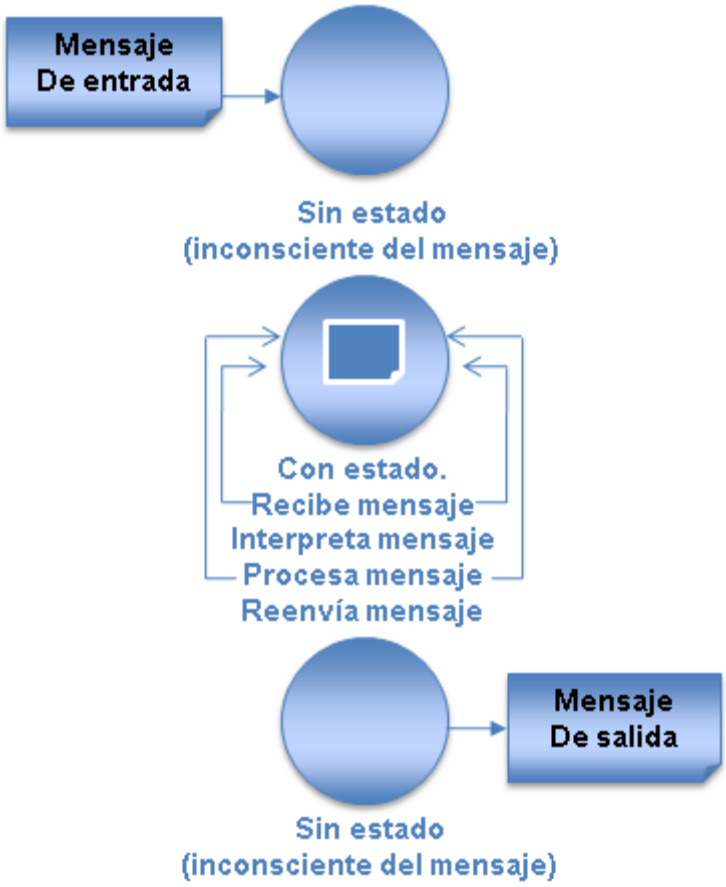


Ilustración 1-8 Principio de no guardar estado.

### 1.4.7 Los servicios deben poder ser descubiertos

Este principio está muy ligado al contrato de los servicios, debe existir un mecanismo de inventario de servicios en los que se especifique su funcionalidad y cómo poder acceder a ellos.

**Objetivos**

- ✓ Exponer las funcionalidades de los servicios
- ✓ Evitar la creación de servicios redundantes duplicando lógica

**Características**

- ✓ Se debe contar con un registro de los servicios existentes y agregar los servicios nuevos que son creado

**Ejemplo**

Cuando se desarrolla un sistema se crea un directorio de servicios, en el que se puede consultar su contrato, de esta forma cuando se necesite cierta lógica se puede consultar si ya existe un servicio que cumpla los requerimientos, de lo contrario, proseguir con el desarrollo necesario.

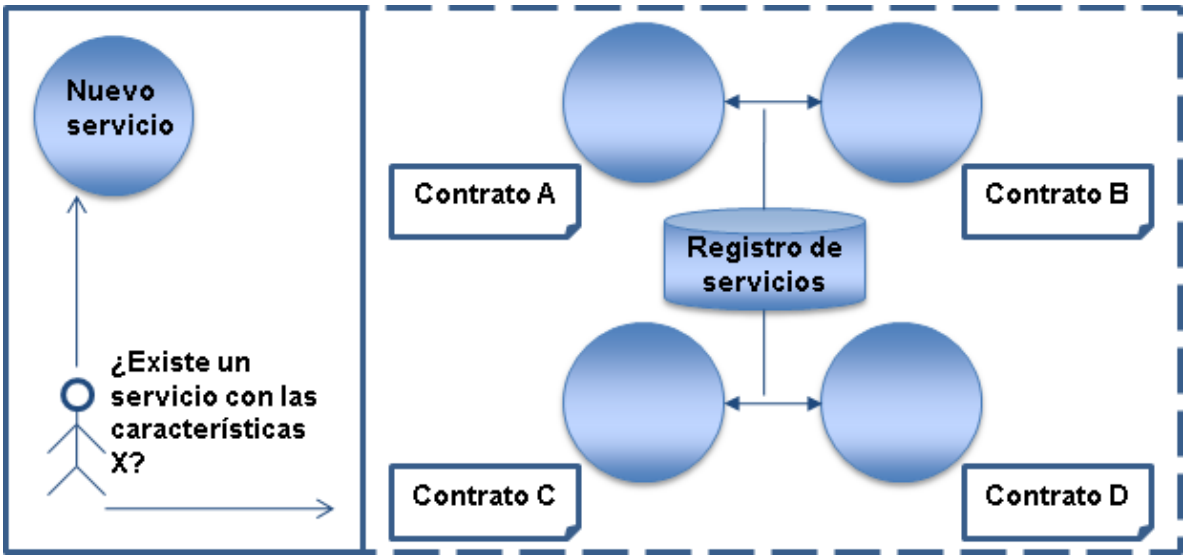


Ilustración 1-9 Principio de servicios descubiertos.

### 1.4.8 Los servicios deben ser abstractos

El propósito de un servicio es realizar sus tareas sin importar cómo opera internamente, simplemente debe arrojar los resultados obtenidos, de esta forma parecerán como una caja negra, dando importancia al **¿qué?** y no al **¿cómo?**

**Objetivos**

- ✓ Ocultar el funcionamiento interno del servicio a los clientes
- ✓ Proporcionar la información requerida para la ejecución del mismo

**Características**

- ✓ Los servicios presentan puntualmente la información requerida para su ejecución
- ✓ El contrato del servicio no muestra información de cómo está construido

**Ejemplo**

Suponiendo que el ejemplo del servicio de generación del CURP fuera expuesto a través un servicio web, el cual podría ser consumido sin ningún problema sin saber el lenguaje que utiliza por dentro, ya sea Java, .Net, C++, etc.

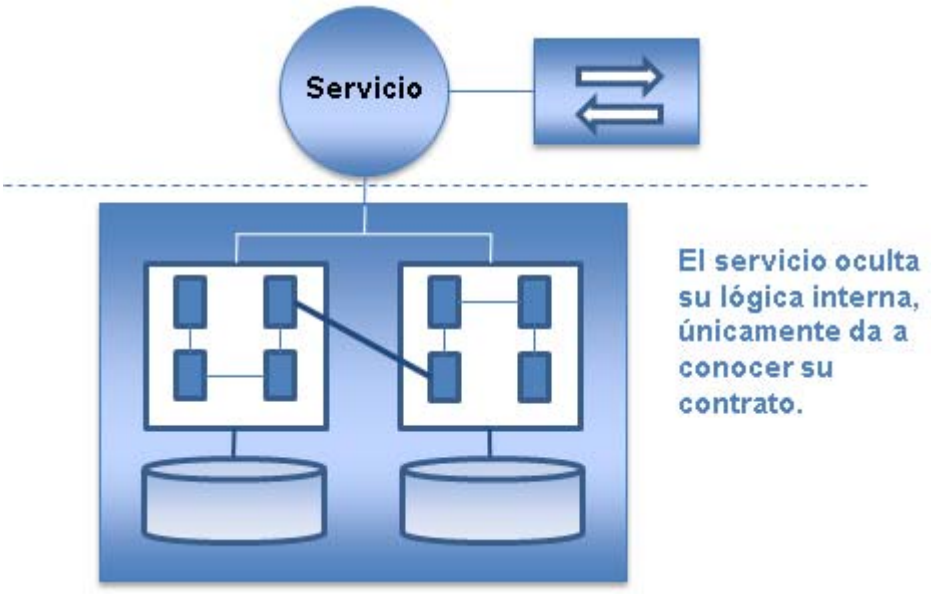


Ilustración 1-10 Los servicios deben ser abstractos.

# CAPÍTULO 2. MSOAM, UNA METODOLOGÍA GENÉRICA PARA SOA

En este Capítulo se describe la metodología MSOAM (Mainstream SOA Methodology) propuesta por Thomas Erl [2], el motivo principal por el que fue elegida es que es bastante genérica y permite ser adecuada a las necesidades del proyecto en el que se esté implementando, también permite una libre elección de herramientas tecnológicas utilizadas.

Análisis y diseño son los dos primeros pasos de MSOAM. Estos pasos definen las características de cómo se debe desarrollar una arquitectura orientada a servicios.

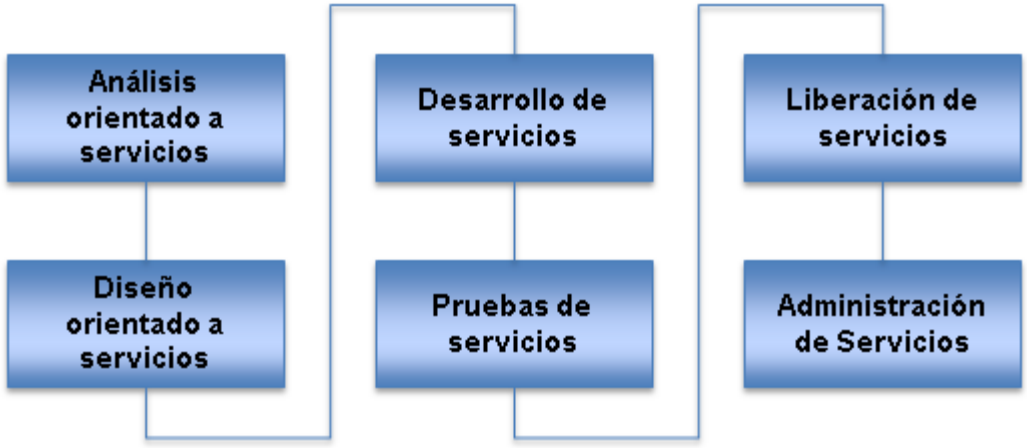


**Ilustración 2-1 Metodología MSOAM.**

Los pasos de esta metodología deben ser organizados en un proceso que:

- Satisfaga las preferencias de qué tipos de capa de servicios deseamos ofrecer
- Coordine la liberación de las aplicaciones y servicios en tiempo y forma
- Apoye la transición hacia una arquitectura SOA al mismo tiempo que ayuda a satisfacer las necesidades inmediatas, específicas del sistema

Para llevar a cabo los puntos anteriores hay que seguir una estrategia basada en las prioridades de la organización y mediar los objetivos de la migración a largo plazo con los requisitos a corto plazo. MSOAM se basa en una estrategia *top – down*.

En el enfoque *Top – down* los servicios son analizados a fondo y diseñados con el fin de ser escalables y reutilizados. Esta estrategia requiere tiempo y dinero debido a que los primeros pasos

pueden tomar un tiempo considerable, por lo cual los resultados no son evidentes de manera inmediata.

## 2.1 Modelo de servicios

Modelo de servicios es la clasificación de los tipos de servicios que existen en un sistema de acuerdo a tres características básicas:

- ✓ Tipo de lógica que encapsulan
- ✓ Grado de reutilización
- ✓ Relación de la lógica con los dominios del negocio

Conforme a estas características los servicios pueden agruparse en tres principales categorías:

- ✓ Servicios de entidad
- ✓ Servicios de tarea
- ✓ Servicios de utilidad

Dependiendo del análisis y necesidades requeridas puede ser necesario crear una subcategoría para hacer frente a una necesidad más específica.

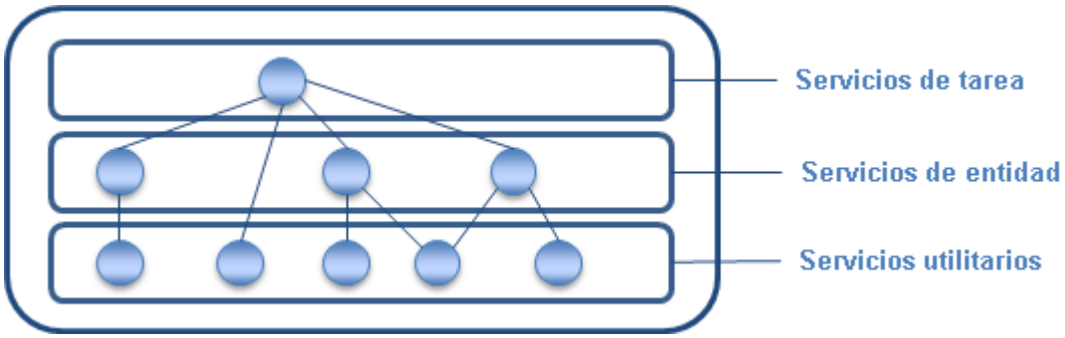


**Ilustración 2-2 Modelo de servicios.**

Un **servicio candidato** es aquel que se identifica en el análisis inicial, una vez que se determina que realmente debe ser desarrollado y si será expuesto o interno, se le llamará formalmente servicio.

De igual manera se aplica a las operaciones, inicialmente son **operaciones candidatas** que serán evaluadas para determinar si serán implementadas.

### 2.1.1 Servicios de entidad

También conocidos como **servicios de negocio centrados en entidad** o **servicios de entidad de negocio**. En inglés: **entity services**, **entity centric business services** o **business entity services**.

Su finalidad es dar soporte a las operaciones de administración de las entidades de negocio, tales como operaciones CRUD.

Este tipo de servicios suele tener un alto grado de reutilización debido a que los servicios de tarea hacen uso de estos para darle sentido al negocio de la organización.

Ejemplos de servicios de entidad:

- Empleado
- Cliente
- Factura


**Ilustración 2-3 Servicio de entidad.**

### 2.1.2 Servicios de tarea

Estos servicios realizan tareas un tanto más complejas que contienen propiamente el negocio y/o reglas de negocio, usualmente están compuestos por otros servicios, tanto de entidad como de tarea. Cuando su lógica es altamente compleja tienden a ser menos reutilizables.

Ejemplos de servicios de tarea:

- Contaduría
- Administración de usuarios
- Mensajería


Ilustración 2-4 Servicio de tarea.

Un conjunto de servicios pueden formar parte de un proceso de negocio padre, es posible que estos estén desarrollados como componentes independientes o como servicios web, incluso estar alojados en una plataforma especializada en orquestación, para el último caso las características de diseño son un tanto distintas debido a la naturaleza de la tecnología subyacente, es conveniente nombrarlo como servicio de orquestación o de tarea orquestada.

Los servicios de tarea también son llamados servicios de negocio centrados en tarea, servicios de proceso de negocio (task services, task-centric business services, business process services).

Mientras que a los servicios orquestados se les conoce por otros nombres como servicios de tarea orquestados, servicios de proceso de negocio (En inglés, process services, business process services, orchestration services).

### 2.1.3 Servicios de utilidad

Los servicios utilitarios también son conocidos como **servicios de aplicación**, **servicios de infraestructura** o **servicios de tecnología**, en inglés **utility services**, **application services**, **infrastructure services**, **technology services**.

Estos servicios no encapsulan lógica de negocio, se dedican a proporcionar funcionalidad reutilizable, transversal, por ejemplo, registro de eventos, mensajería, manejo de excepciones, etc. Estos pueden ser fácilmente reutilizados por otros sistemas ya que son de uso general. En este tipo de servicios hay una especialización llamada servicios *wrapper*, estos servicios hacen las veces de un adaptador para interactuar con sistemas legados o externos, exponiendo una interfaz de la funcionalidad heredada que envuelven.

Ejemplos de servicios utilitarios:

- Log

- Exportación de archivo
- Enviar correo


Ilustración 2-5 Servicio de utilidad.

## 2.2 Estrategia Top - down

En esta estrategia primero se realiza un análisis y promueve el desarrollo de tres capas de servicios, capa de aplicación, capa de negocio y capa de orquestación. Top – down se basa en los pasos mostrados en la siguiente figura, es posible que no se implementen todos ellos, los requisitos de negocio deben estar ya establecidos ya que no son parte de esta estrategia.

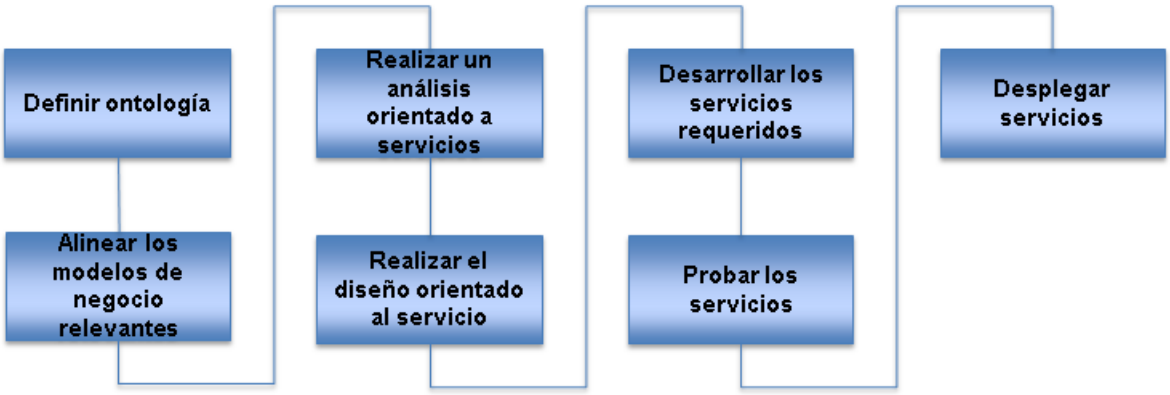


Ilustración 2-6 Estrategia top – down.

- Paso 1. Definir ontología

  Se debe clasificar la información con la que se trabaja con el fin de tener un vocabulario común y cómo se relacionan los diferentes términos, para que todos los involucrados en la solución tengan el mismo contexto de los términos.

- Paso 2. Alinear los modelos de negocio relevantes
  Aquí se generan o ajustan modelos que brinden una correcta interpretación del negocio. Entre los modelos que pueden emplearse se encuentran diagramas de entidades que pueden ser utilizados como base para desarrollar los servicios de entidad.

- Paso 3. Realizar un análisis orientado a servicios
  Este paso está subdividido en tres, en el primero se identifican las necesidades de negocio para identificar que se requiere construir; en el segundo se identifica si hay sistemas o servicios existentes que puedan seguir siendo utilizados. Por último se realiza el modelado de los servicios, es decir, se da forma a los servicios con base en la información recabada previamente.

  Este se abordará a profundidad en el "CAPÍTULO 3. ANÁLISIS ORIENTADO A SERVICIOS, PASO 3".

- Paso 4. Realizar el diseño orientado al servicio
  En este paso se realizan las especificaciones finales de los servicios, entre ellas las interfaces de los servicios, el cómo deben interactuar entre ellos, se especifican las capas de la solución que contendrán a los distintos tipos de servicios.
  Este se abordará a profundidad en el "CAPÍTULO 4. DISEÑO ORIENTADO A SERVICIOS, PASO 4".

- Paso 5. Desarrollar los servicios requeridos
  En este paso se realiza el desarrollo de los servicios con forme a las especificaciones de diseño del Paso 4.

- Paso 6. Pruebas de los servicios
  En este paso se realizan pruebas a los servicios para validar que cumplen con la calidad y especificaciones requeridas, se revisa que los servicios satisfagan las necesidades de todos sus consumidores.

- Paso 7. Despliegue de los servicios
  Los servicios son desplegados en producción, para ello debe considerarse que el ambiente en el que son alojados es el adecuado para que trabajen satisfactoriamente.

Para tener un panorama general de todos los pasos de MSOAM se agrega la siguiente tabla.

**MSOAM**

1. Definir ontología.
2. Alinear los modelos de negocio relevantes.
3. Realizar un análisis orientado a servicios.
   - 3.1. Definir las necesidades de automatización del negocio.
   - 3.2. Identificar sistemas de automatización existentes.
   - 3.3. Modelado de servicios candidatos.
     - 3.3.1. Descomponer el proceso de negocio.
     - 3.3.2. Identificar operaciones de los servicios candidatos.
     - 3.3.3. Abstraer la lógica de orquestación.
     - 3.3.4. Crear servicios de negocio candidatos.
     - 3.3.5. Refinar y aplicar los principios de orientación a servicios.
     - 3.3.6. Identificar composiciones de servicios candidatos.
     - 3.3.7. Revisar la agrupación de las operaciones de servicio de negocio.
     - 3.3.8. Analizar los requerimientos de procesamiento de aplicación.
     - 3.3.9. Identificar las operaciones de servicios de aplicación candidatos.
     - 3.3.10. Crear servicios de aplicación candidatos.
     - 3.3.11. Revisar composiciones de servicios candidatos.
     - 3.3.12. Revisar la agrupación de los servicios de aplicación candidatos.
     - 3.3.13. (Opcional) Mantener un inventario de servicios candidatos.
4. Realizar el diseño orientado al servicio.
   - 4.1. *Compose* SOA.
     - 4.1.1. Seleccionar capas de servicio.
     - 4.1.2. Estándares.
     - 4.1.3. Extensiones SOA.
   - 4.2. Diseño de servicios de entidad.
     - 4.2.1. Revisar servicios existentes.
     - 4.2.2. Definir los parámetros (mensajes) de entrada y salida.
     - 4.2.3. Especificar una interfaz inicial del servicio.
     - 4.2.4. Aplicar los principios de orientación a servicios.
     - 4.2.5. Estandarizar y refinar la interfaz del servicio.
     - 4.2.6. Extender el diseño del servicio.
     - 4.2.7. Identificar el procesamiento requerido.
   - 4.3. Diseño de servicios de aplicación.
     - 4.3.1. Revisar servicios existentes.

4.3.2. Confirmar el contexto.
4.3.3. Especificar una interfaz inicial del servicio.
4.3.4. Aplicar principios de orientación a servicios.
4.3.5. Estandarizar y refinar la interfaz del servicio.
4.3.6. Especular características funcionales al servicio.
4.3.7. Identificar limitaciones técnicas.

4.4. Diseño de servicios de negocio o de tarea.
4.4.1. Definir la lógica del flujo de trabajo (workflow).
4.4.2. Especificar una interfaz inicial del servicio.
4.4.3. Aplicar principios de orientación a servicios.
4.4.4. Estandarizar y refinar la interfaz del servicio.
4.4.5. Identificar el procesamiento requerido.

4.5. Diseño de servicio orientados al proceso de negocio.
4.5.1. Mapear los escenarios de interacción.
4.5.2. Diseñar interfaz del servicio de proceso.
4.5.3. Formalizar conversaciones entre servicios asociados
4.5.4. Definir lógica del proceso.
4.5.5. Alinear escenarios de interacción y perfeccionar el proceso.

5. Desarrollar los servicios requeridos
6. Probar los servicios.
7. Desplegar los servicios.

# CAPÍTULO 3. ANÁLISIS ORIENTADO A SERVICIOS, PASO 3

En el proceso de análisis se determina cuáles y cómo pueden ser soportadas las tareas de negocio dentro de una arquitectura orientada a servicios. El análisis busca dar respuesta a las preguntas:

- ¿Qué servicios se deben construir?
- ¿Qué lógica debe encapsular cada servicio?

Los objetivos del análisis orientado a servicios son:

- Definir un conjunto preliminar de operaciones candidatas
- Agrupar las operaciones en contextos lógicos, estos contextos constituyen los servicios candidatos
- Definir los límites de los servicios preliminares, evitando que se superpongan con otros
- Identificar la lógica con alto grado de reutilización
- Asegurar que el contexto de la lógica encapsulada es adecuado para su uso previsto
- Definir modelos de composición preliminares

El proceso de análisis se subdivide en tres pasos:

1. Definir las necesidades de automatización del negocio
2. Identificar sistemas de automatización existentes
3. Modelado de servicios candidatos


Ilustración 3-1 Proceso de análisis de servicios.

**Paso 3.1. Definir las necesidades de automatización del negocio.**

En este paso se realiza el análisis de los requerimientos de negocio, tomando en cuenta principalmente el alcance que tendrá la solución.

Para que un proceso de automatización pueda ser definido es necesario que los requerimientos del negocio tengan la suficiente madurez. La información obtenida en este paso puede ser utilizada como punto de entrada para el paso 3.

**Paso 3.2. Identificar sistemas de automatización existentes.**

En este paso se revisan los sistemas ya implantados con el objetivo de validar la lógica de negocio existente. En este punto aun no es necesario saber cómo está construida la implementación, aquí

se necesita saber cuáles sistemas pueden verse afectados y cuáles pueden catalogarse como servicios candidatos.

### Paso 3.3. Modelado de servicios candidatos.

Las operaciones de los servicios candidatos se agrupan de acuerdo a la lógica de su contexto.

Aquí se hace la organización de información obtenida en los pasos previos. Las fuentes de información pueden ir desde modelos en documentos de negocio hasta juntas con los expertos de negocio.

El modelado de servicios candidatos se subdivide en 12 pasos.

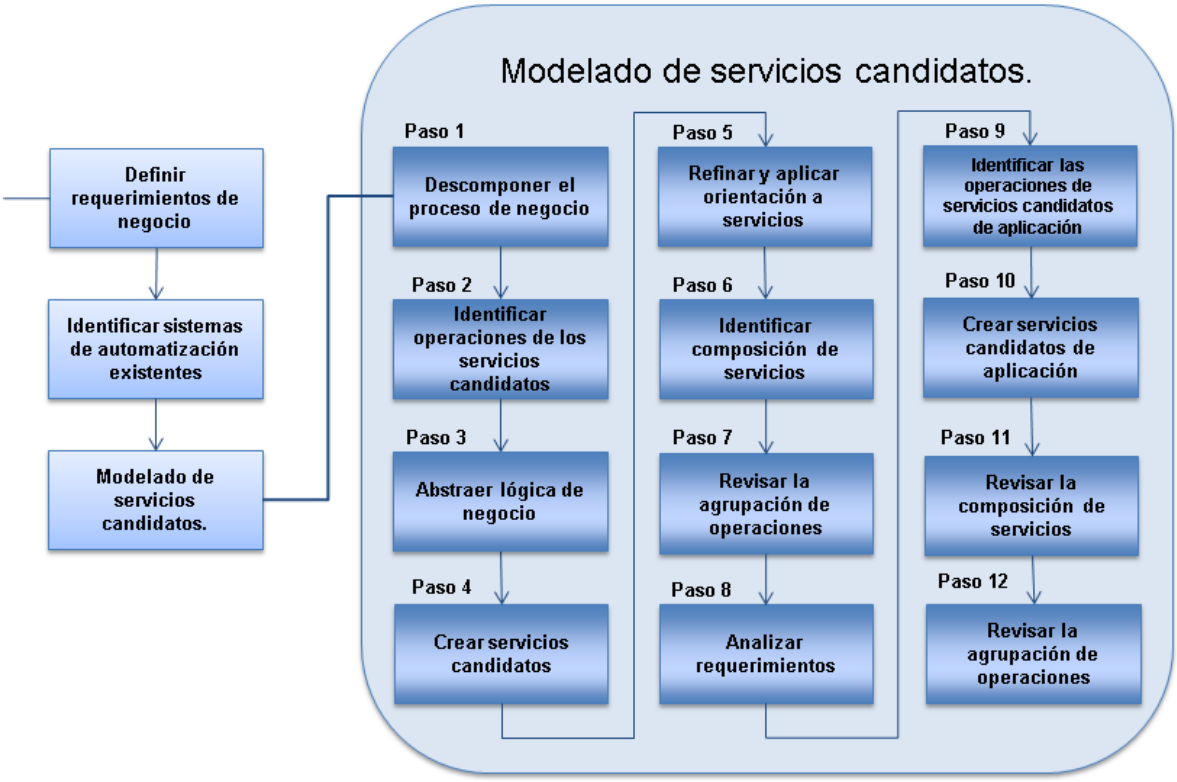


Ilustración 3-2 Pasos del modelado de servicios.

## 3.1 Modelado de servicios

**Paso 3.3.1: Descomponer el proceso de negocio.**

Una vez identificado el proceso de negocio completo se debe segmentar en subprocesos más específicos y granulares, capaces de especificar las tareas que verdaderamente dan sentido a la automatización del proceso de negocio.

**Paso 3.3.2: Identificar operaciones de los servicios candidatos.**

En el proceso de negocio existen pasos que no pueden ser automatizados, por lo tanto no pueden ser encapsulados en un servicio candidato.

Por ejemplo:

- Tareas manuales.
- Tareas realizadas por lógica legada existente.

**Paso 3.3.3: Abstraer la lógica de orquestación.**

Este paso puede omitirse en caso de no requerir una capa de orquestación.

Deben identificarse las reglas de negocio, secuencias complejas de cómo los servicios deben ser secuenciados, controles a flujos de excepción; una vez identificados serán útiles para los flujos de trabajo que deben seguirse para el desarrollo de la solución.

**Paso 3.3.4: Crear servicios de negocio candidatos.**

Se debe realizar una revisión completa de los pasos del proceso de negocio y agruparlos en contextos lógicos. En los servicios de entidad es recomendable tomar en cuenta posibles operaciones adicionales que ayuden a su reusabilidad.

**Paso 3.3.5: Refinar y aplicar los principios de orientación a servicios.**

Para lograr que los servicios candidatos estén al nivel de una arquitectura SOA se debe hacer una revisión a la lógica interna de las operaciones, con base en los principios de diseño de servicios vistos en la sección “1.4 Principios de diseño de servicios”. (principalmente de los cuatro básicos).

- Reutilización Etapa de modelado
- Autonomía Etapa de modelado
- Sin estado Etapa de diseño

- Detectabilidad Etapa de diseño

En el proceso de análisis se debe procurar alcanzar los dos primeros principios, de acuerdo en lo visto en la sección “1.3 Arquitectura Orientada a Servicios, SOA”.

**Paso 3.3.6: Identificar composiciones de servicios candidatos.**

En este paso se identifican los escenarios más comunes en el proceso de negocio. Éstos serán ilustrados para ver más de cerca cómo se componen. Realizar las composiciones de los servicios permite:

- Dar una buena idea de si la agrupación propuesta es conveniente
- Identificar las posibles relaciones entre los servicios de orquestación y de negocio.
- Identificar composiciones de servicios potenciales.
- Identificar si hay lógica del proceso de trabajo que se haya omitido.

Se deben tomar en cuenta las posibles excepciones que puedan existir. En este punto las capas de servicios siguen siendo preliminares.

**Paso 3.3.7: Revisar la agrupación de las operaciones de servicio de negocio.**

Después de haber realizado la composición de servicios del paso anterior es conveniente volver a revisar la agrupación de los pasos del proceso de negocio, por si es necesario hacer una reorganización de los servicios candidatos.

**Paso 3.3.8: Analizar los requerimientos de procesamiento de aplicación.**

Este paso puede requerirse (por lo que se considera opcional) en procesos de negocio complejos y entornos grandes. Se realiza un análisis a fondo de las necesidades de procesamiento interno de los servicios candidatos para abstraer posibles dependencias tecnológicas que pudieran trasladarse a un servicio de aplicación. Las siguientes preguntas deben formularse:

- ¿Qué lógica interna se necesita para ejecutar la operación?
- ¿La lógica de la aplicación ya existe o se requiere un nuevo desarrollo?
- ¿La lógica requerida extiende los límites de la aplicación?, es decir, ¿se requiere más que un sistema de información para realizar esta acción?

**Paso 3.3.9: Identificar las operaciones de servicios de aplicación candidatos.**

En este paso se deben analizar los requisitos que deben cubrir los servicios de aplicación. Para los nombres de las operaciones se recomienda no hacer referencia a la lógica de negocio que los utiliza, con el fin de que pueden ser utilizados por otros sistemas o servicios.

**Paso 3.3.10: Crear servicios de aplicación candidatos.**

Las operaciones son agrupadas para conformar los servicios candidatos, la agrupación se debe realizar de acuerdo a la relación lógica entre operaciones, por ejemplo:

- Asociación con un sistema legado
- Asociación con uno o más componentes de la aplicación
- Agrupación lógica de acuerdo al tipo de función

La agrupación propuesta en este paso no es definitiva, dado que durante el proceso de diseño pueden hacerse adecuaciones.

**Paso 3.3.11: Revisar composiciones de servicios candidatos.**

El procedimiento realizado en el paso 5 se efectúa con los servicios de aplicación candidatos. Dando seguimiento de cómo los servicios de negocio candidatos se enlazan con los servicios de aplicación subyacentes.

**Paso 3.3.12: Revisar la agrupación de los servicios de aplicación candidatos.**

Después de realizar el paso 11 puede que haya la necesidad de realizar cambios en la agrupación de los servicios de aplicación y validar que no se haya omitido algún paso.

**Paso 3.3.13: (Opcional) Mantener un inventario de servicios candidatos.**

Se sugiere crear un inventario de servicios que sirva de referencia para cuando el sistema crezca, se realicen modificaciones o se comience a interactuar con futuros proyectos. El inventario facilitará el verificar que funciones se pueden reutilizar o que servicios pueden adecuarse.

# CAPÍTULO 4. DISEÑO ORIENTADO A SERVICIOS, PASO 4

En este proceso se concreta el diseño físico de los servicios candidatos, realizar el diseño orientado a servicios tiene como objetivos:

- Determinar el conjunto básico de extensiones de la arquitectura
- Establecer límites de la arquitectura
- Identificar las normas de diseño requeridas
- Definir diseños de interfaz de servicios abstractos
- Identificar las composiciones de servicios potenciales
- Evaluar el soporte de los principios de orientación de servicios

El diseño orientado a servicios está conformado por los siguientes pasos:

- Paso 4.1: *Compose* SOA
- Paso 4.2: Diseño de servicios de negocio centrados en entidad, explicado en la sección “4.2 Paso 4.2: Diseño de servicios de entidad”.
- Paso 4.3: Diseño de servicios de aplicación, explicado en la sección “4.3 Paso 4.3: Diseño de servicios de aplicación”.
- Paso 4.4: Diseño de servicios de negocio centrados en tareas, explicado en la sección “4.4 Paso 4.4: Diseño de servicios de negocio o de tarea”.
- Paso 4.5: Diseño de servicio orientados al proceso de negocio, explicado en la sección “4.5 Paso 4.5: Diseño de servicios de proceso de negocio”.

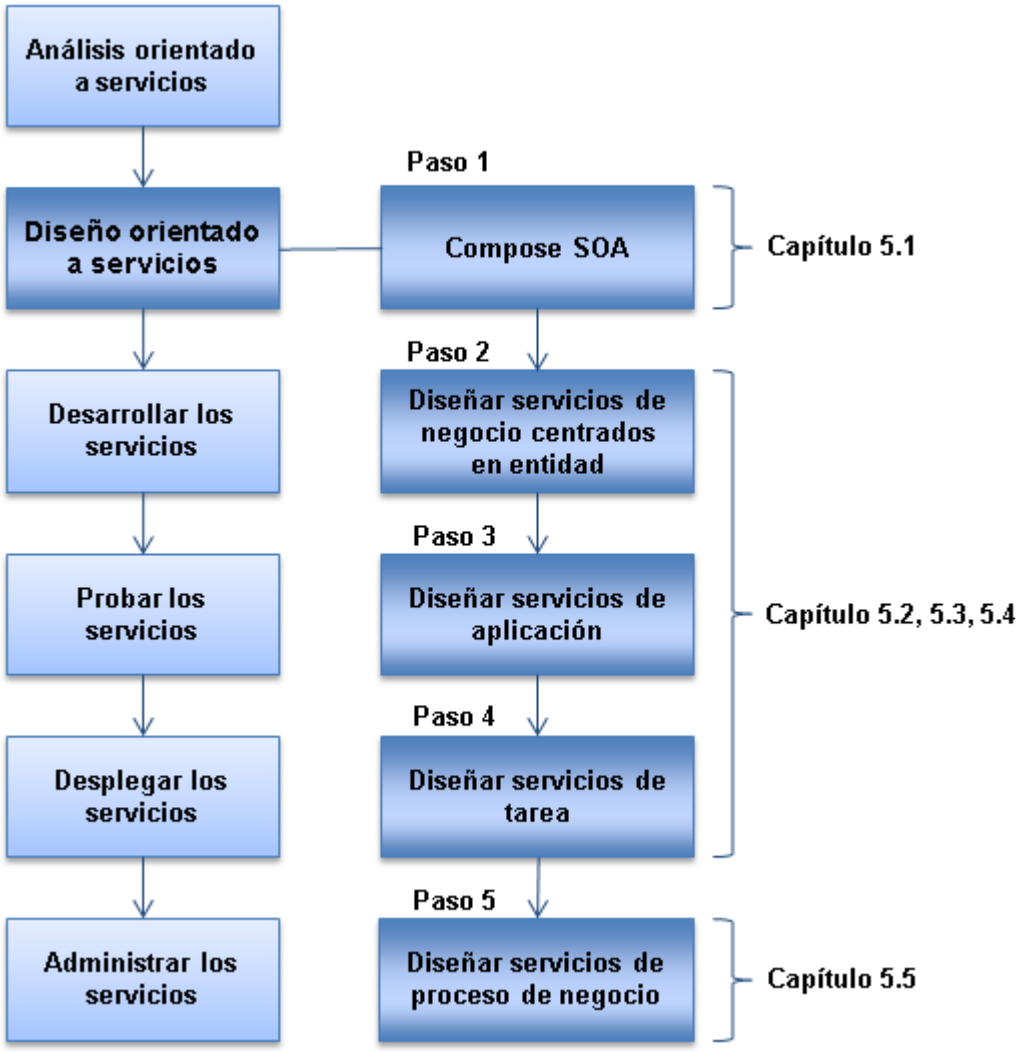


Ilustración 4-1 Proceso de diseño de servicios.

## 4.1 Paso 4.1: *Compose* SOA.

C*ompose* se refiere a la acción de integrar distintos componentes para crear un ente más grande, en este caso una arquitectura orientada a servicios.

Este paso consiste en tres sub pasos, que son tratados en la sección “7.1 Paso 4.1: *Compose* SOA”.

- Paso 4.1.1 Elegir las capas de los servicios
- Paso 4.1.2 Posición estándares SOA principales
- Paso 4.1.3 Elegir extensiones SOA

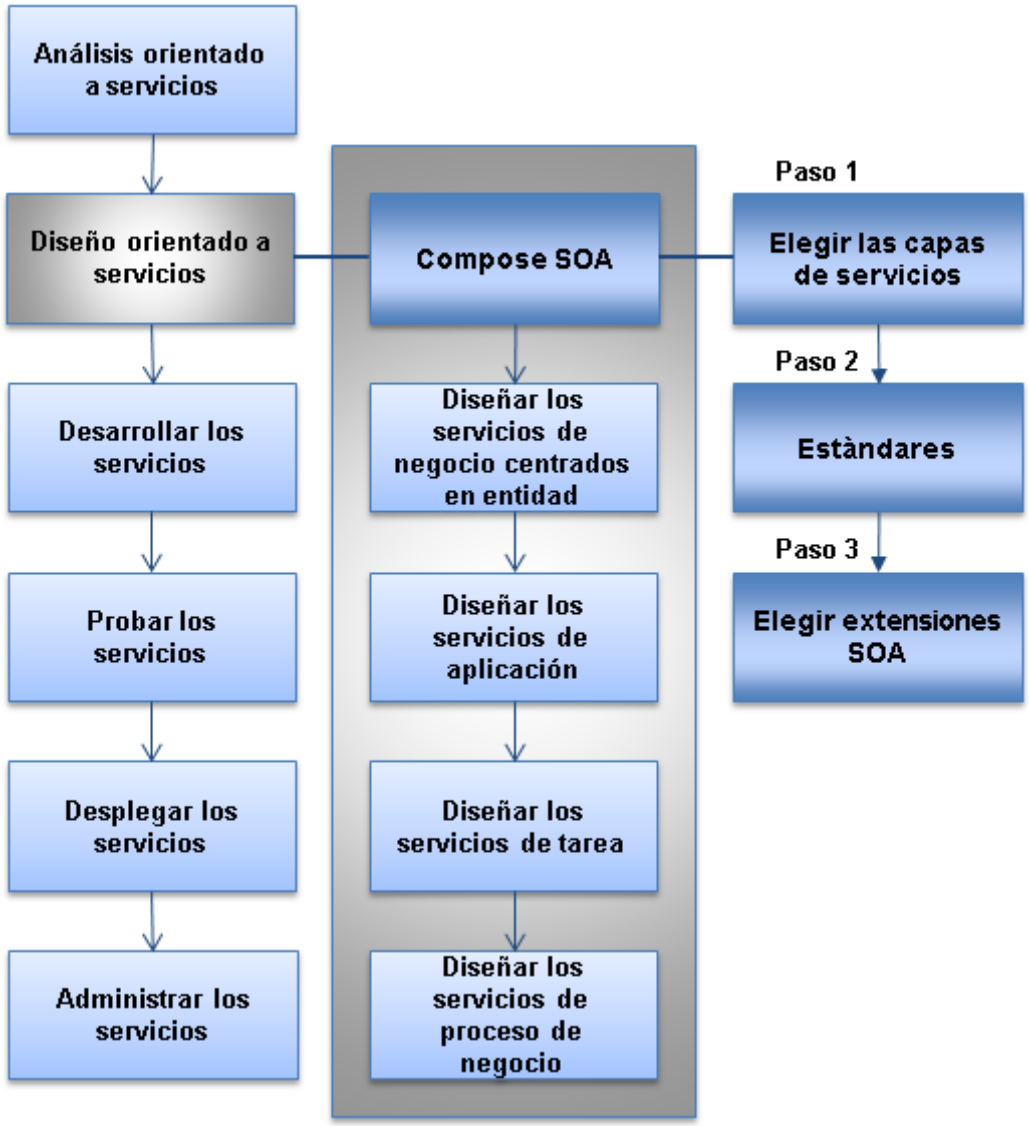


Ilustración 4-2 *Compose* SOA.

Entre las decisiones que deben tomarse al desarrollar una arquitectura de software se encuentran:

- Las tecnologías que se van a utilizar
- ¿Cómo se van organizar los componentes?
- ¿Qué tipos de servicios serán construidos y cómo serán organizados entre las distintas capas?

**Paso 4.1.1. Seleccionar capas de servicio.**

Se debe realizar una evaluación de qué capas van a integrar la solución, algunos puntos que deben ser tomados en cuenta son: existe la necesidad de enlazarse con otros módulos o sistemas, adaptarse a lineamientos existentes en la organización.

La capa de interface de servicios puede ser dividida en tres, como se vio en el la sección “2

Modelo de servicios.

- Capa de servicios de aplicación
  - Servicios de aplicación
- Capa de servicios de negocio
  - Servicios de negocio o de tarea
  - Servicios de entidad
- Capa de orquestación de servicios
  - Servicios de negocio

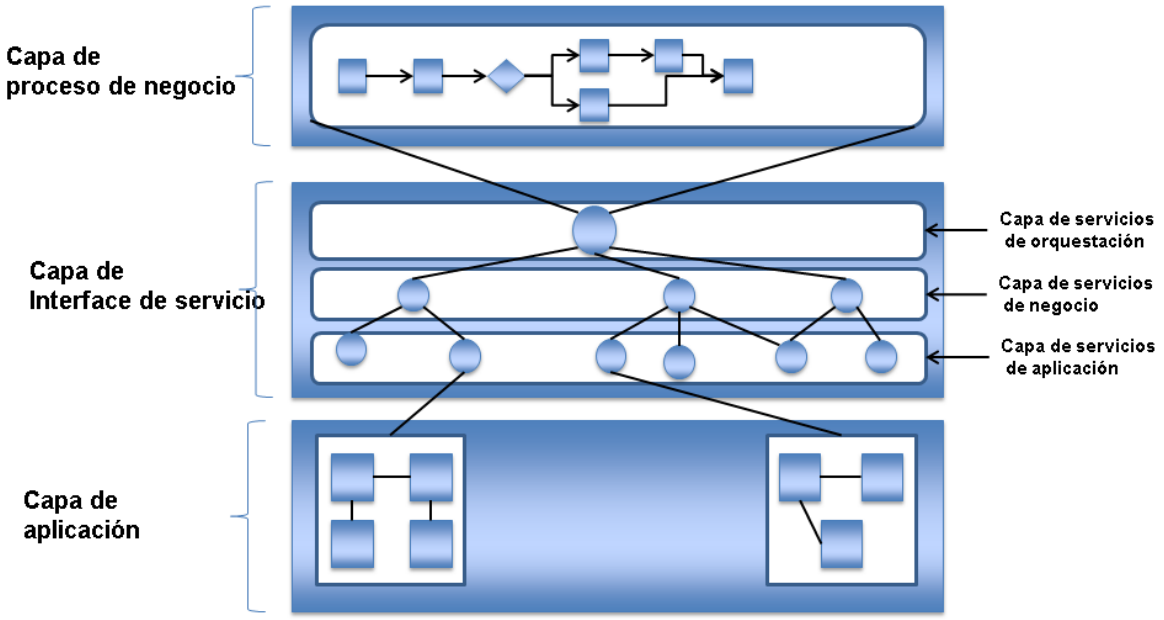


Ilustración 4-3 Capas de servicios.

**Paso 4.1.2: Estándares.**

Se recomienda que en el diseño de la solución se utilicen estándares, ya que promueven reglas normalizadas que describen los requisitos que deben ser cumplidos por un producto, proceso o

servicio, con el objetivo de establecer un mecanismo base para permitir que distintos elementos de hardware o software que lo utilicen sean compatibles entre sí.

Algunos estándares que son utilizados para la interconexión entre sistemas o servicios son: RMI, EJB remotos, servicios web por SOAP o Rest.

**Paso 4.1.3: Extensiones SOA.**

En este paso se debe verificar si la solución propuesta requiere de tecnologías o herramientas específicas que brinden la suficiente robustez para que esta pueda adecuarse a los posibles cambios, estos cambios son donde puede notarse visiblemente el poder de SOA.

Existen herramientas, algunas comerciales, tales como los motores BPEL, BUS de servicio, estándares para servicios web, como WS-security, WS-Transacction.

En el presente trabajo no se utilizaran herramientas extra como BPEL o BUS, solo se utilizaran algunos de los estándares para los servicios web, esto se verá más adelante.

## 4.2 Paso 4.2: Diseño de servicios de entidad

Es conveniente comenzar por el diseño de este tipo de servicios debido a que presentan un alto grado de reusabilidad por servicios de capas superiores.

A continuación se presentan los pasos para realizar el diseño de estos servicios, hay que recordar que la metodología MSOAM puede ser adecuada a las necesidades existentes, sin perder de vista el diseño orientado a servicios.

**Paso 4.2.1: Revisar servicios existentes.**

El primer paso es verificar si es necesario construir el servicio. Para ello se valida si ya existe algún servicio que implementa, total o parcialmente, la funcionalidad requerida.

**Paso 4.2.2: Definir los parámetros (mensajes) de entrada y salida.**

Una vez que se ha determinado la necesidad de crear el servicio se procede a definir los mensajes de entrada y de salida.

Cabe mencionar que no necesariamente debe existir una relación uno a uno entre los servicios de entidad y las entidades que conforman el modelo de entidad.

**Paso 4.2.3: Especificar una interfaz inicial del servicio.**

Para definir la interfaz inicial del servicio se siguen los pasos siguientes:

- Verificar que las operaciones son genéricas y reutilizables, asegurándose que la granularidad de la lógica encapsulada es adecuada
- Asignar nombres a las operaciones de los servicios
- Formalizar los valores de entrada y de salida para cada una de las operaciones

**Paso 4.2.4: Aplicar los principios de orientación a servicios.**

En este paso se verifica que los servicios de entidad estén cumpliendo con los cuatro principios básicos.

- Reutilización
- Autonomía
- No guardar estado
- Detectables

**Paso 4.2.5: Estandarizar y refinar la interfaz del servicio.**

Si la organización cuenta con normas o directrices de diseño establecidas hay que alinearse a ellas. Ejemplos incluyen: el cómo deben ser nombrados los servicios y operaciones, los tipos de datos a utilizar, el cumplimiento de estándares específicos, etc.

**Paso 4.2.6: Extender el diseño del servicio.**

Con el fin de lograr una mayor reutilización de los servicios se puede hacer una especulación de qué otras características debería cubrir el servicio, por ejemplo agregar nuevas operaciones o parámetros a las operaciones.

Si se decide agregar operaciones nuevas deben repetirse los pasos anteriores.

**Paso 4.2.7: Identificar el procesamiento requerido.**

En este paso se debe verificar que lo realizado en el proceso de análisis de servicios cumple con las necesidades de negocio. De ser necesario, se pueden crear nuevos servicios para cumplir con la funcionalidad requerida.

## 4.3 Paso 4.3: Diseño de servicios de aplicación

El diseño de servicios de aplicación no requiere de un análisis de negocio, básicamente su objetivo es abstraer la funcionalidad de otros sistemas o identificar utilidades genéricas.

**Paso 4.3.1: Revisar servicios existentes.**

De igual manera que en el diseño de servicios centrados en entidad se debe revisar que la funcionalidad que se prende construir o parte de ella no esté ya implementada.

**Paso 4.3.2: Confirmar el contexto.**

Reevaluar que las operaciones asignadas a los servicios de aplicación verdaderamente están ubicadas en el servicio correcto, de no ser así se deben asignar al contexto al que pertenecen.

**Paso 4.3.3: Especificar una interfaz inicial del servicio.**

Para este paso se siguen las mismas normativas que para el pasó homólogo en el diseño de servicios de entidad.

**Paso 4.3.4: Aplicar principios de orientación a servicios.**

Para este paso se siguen las mismas normativas que para el pasó homólogo en el diseño de servicios de entidad.

**Paso 4.3.5: Estandarizar y refinar la interfaz del servicio.**

Para este paso se siguen las mismas normativas que para el pasó homólogo en el diseño de servicios de entidad.

**Paso 4.3.6: Especular características funcionales al servicio.**

Para este paso se siguen las mismas normativas que para el pasó homólogo en el diseño de servicios de entidad.

**Paso 4.3.7: Identificar limitaciones técnicas.**

Se debe evaluar la eficacia de estos servicios para hacer frente a los posibles problemas que puedan presentarse.

- La disponibilidad ante los sistemas subyacentes
- Los tiempos de respuesta
- Problemas de seguridad con entidades externas
- Existencia de APIs para su implementación

## 4.4 Paso 4.4: Diseño de servicios de negocio o de tarea

El proceso de diseño para estos servicios requiere menos esfuerzo que los anteriores debido a que son más específicos y en consecuencia menos reutilizables.

**Paso 4.4.1: Definir la lógica del flujo de trabajo (*workflow*).**

Estos servicios contienen la lógica del flujo de trabajo que se requiere para completar las tareas del negocio. Como primer paso hay que definir la lógica de los posibles escenarios que pueden presentarse, para este paso es posible apoyarse en el paso 6 del proceso de análisis de servicios.

Este paso puede auxiliarse del uso de diagramas de secuencia para plasmar la lógica de negocio.

**Paso 4.4.2: Especificar una interfaz inicial del servicio.**

Para realizar la interfaz de este tipo de servicios se recomienda lo siguiente:

1. Apoyarse de diagramas de secuencia generados en el paso anterior
2. Documentar los valores de entrada y salida de las operaciones del servicio

**Paso 4.4.3: Aplicar principios de orientación a servicios.**

Para este paso se siguen las mismas normativas que para el pasó homólogo en el diseño de servicios de entidad.

**Paso 4.4.4: Estandarizar y refinar la interfaz del servicio.**

Para este paso se siguen las mismas normativas que para el pasó homólogo en el diseño de servicios de entidad.

**Paso 4.4.5: Identificar el procesamiento requerido.**

Para este paso se siguen las mismas normativas que para el pasó homólogo en el diseño de servicios de entidad.

## 4.5 Paso 4.5: Diseño de servicios de proceso de negocio

Una vez que se ha generado un inventario de los diseños de servicios se procede a crear la capa de orquestación, la cual se encarga de empatar los servicios con la lógica de procesos de negocio. Este paso define la lógica del flujo de trabajo (*workflow*).

Para lograr un buen diseño de la solución es necesario comprender correctamente los requisitos del proceso de negocio, para esto también es necesario revisar las variaciones o flujos alternos que pueden ocurrir y como se responderá ante estos casos.

Existen herramientas que facilitan el diseño de diagramas con los procesos de negocio, y algunas de ellas generan el lenguaje WS-BPEL encargado de la orquestación de los servicios de negocio. El lenguaje WS-BPEL puede utilizarse para la orquestación de servicios web, sin embargo, la orquestación de servicios web puede lograrse con otras tecnologías, siempre que éstas garanticen conseguir el objetivo del flujo del negocio.

Los pasos de diseño para estos servicios son parecidos a los pasos de los servicios de negocio.

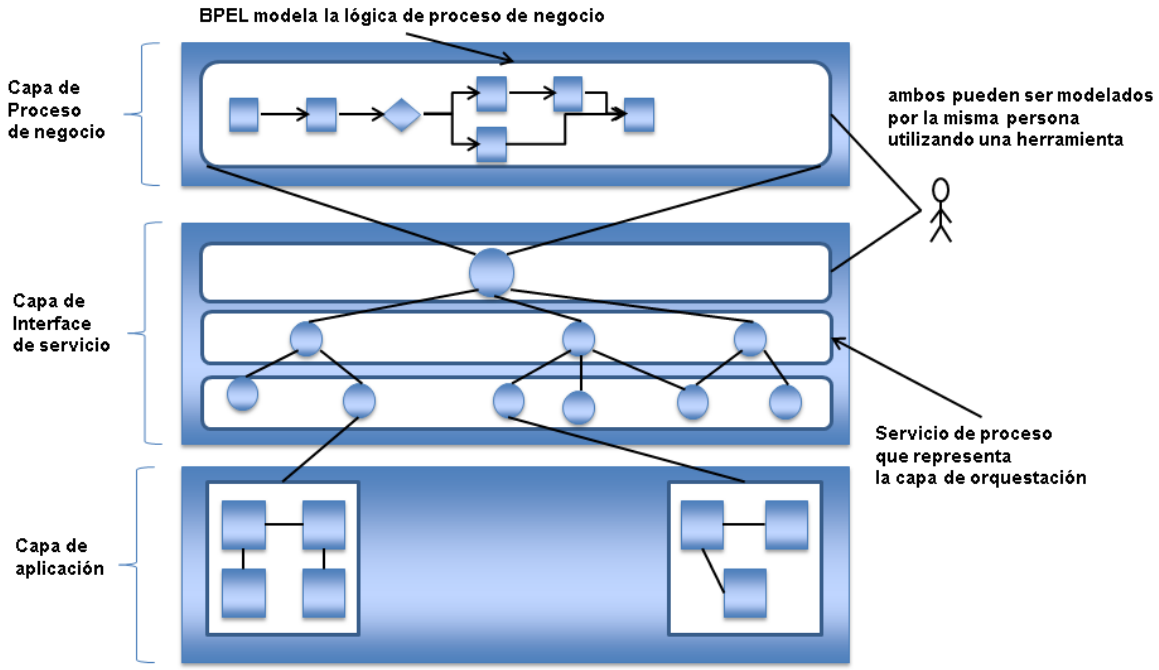


Ilustración 4-4 Diseño de servicios de proceso de negocio.

**Paso 4.5.1: Mapear los escenarios de interacción.**

Con apoyo de la información recabada en el “Capítulo 3, Paso 3.3. Modelado de servicios candidatos. Se recomienda el uso de diagramas que faciliten la identificación de los flujos que pueden tener las actividades.

**Paso 4.5.2: Diseñar interfaz del servicio de proceso.**

De igual forma que en los servicios de tarea, para generar las interfaces se recomienda:

1. Apoyarse de diagramas de secuencia generados en el paso 1
2. Documentar los valores de entrada y salida de las operaciones del servicio

**Paso 4.5.3: Formalizar la comunicación de los servicios asociados**

Revisar que la comunicación a través de los mensajes de entrada y salida entre los servicios de proceso y de negocio permita su interacción correcta, es decir, revisar si hay que hacer un tratamiento especial al resultado que arroje un servicio A y que deba ser utilizado como entrada por un servicio B.

**Paso 4.5.4: Definir lógica del proceso.**

Una vez que se tiene la definición del proceso, la información del flujo de trabajo puede plasmarse utilizando una herramienta especializada en BPEL o alguna otra herramienta.

**Paso 4.5.5: Alinear escenarios de interacción y perfeccionar el proceso.**

Una vez completados los procesos de trabajo es recomendable hacer una revisión de consistencia, en caso de encontrar problemas en los servicios pudiera ser necesario alinearlos. En algunos casos se pueden encontrar huecos de negocio, o simplemente oportunidad de hacer mejoras en los procesos de negocio.

# CAPÍTULO 5. ESPECIFICACIONES DEL SCMV.

En este capítulo se describe el alcance del Sistema de Consultorio Médico Virtual, SCMV, a través de este sistema y con lo visto en los capítulos anteriores se ejemplificará el cómo desarrollar una arquitectura orientada a servicios, concretamente aplicando la metodología MSOAM.

## 5.1 Alcance y requerimientos del sistema.

Recordando que este trabajo nació por el querer ejemplificar el cómo diseñar una arquitectura SOA, se decidió realizar un sistema para la administración médica por el gusto del autor hacia la medicina, el alcance y los requerimientos fueron propuestos con base a las observaciones cuando uno acude a un consultorio, además de sugerir funcionalidad para tener un trato más personalizado.

1. Al sistema podrá acceder el médico, asistentes y pacientes
    - El médico es quien dará de alta a los pacientes para que se les genere su acceso al sistema
2. Por cada paciente se debe generar un historial clínico
3. El expediente debe contener la siguiente información:
    - Nombre completo del paciente
    - Fecha de nacimiento
    - Padecimiento de alguna enfermedad
    - Tipo de sangre
    - Alergias
    - Dirección
    - Teléfono de casa
    - Número de celular
    - Correo electrónico
    - Twitter
4. Por cada consulta que se realice, en el historial médico del paciente se debe almacenar la siguiente información:
    - Datos Básicos:
        - Temperatura
        - Presión

    - Masa corporal
    - Talla
  - Padecimientos
  - Observaciones
  - Medicamento y especificaciones
  - Debe ser posible agregar documentos y/o imágenes
5. Se debe enviar una copia por correo electrónico de la prescripción médica
6. El sistema debe ser capaz de registrar citas a los pacientes, estas pueden ser creadas por el médico, un asistente del médico, o por un paciente que ya cuente con registro en el sistema
7. Cuando falten cierto número de días u horas para la cita se deben enviar recordatorios al paciente, ya sea por twitter, correo electrónico o mensajes SMS
8. El sistema debe ser capaz de generar formatos para solicitudes de estudios médicos, estas solicitudes deben quedar registradas en el sistema
9. El historial médico solo puede ser consultado por el médico titular o por el paciente
10. Se debe poder generar diversos tipos de reportes estadísticos, ya sean por grupos o individuales
11. Se debe contar con un catálogo de medicamentos y cuadros de enfermedades

## 5.2 Diseño de la arquitectura del sistema

De acuerdo a la sección “2.2 Estrategia Top - down”.

**Paso 1. Definir ontología.**

- Un médico tiene pacientes a su cargo
- Cada paciente tiene un historial o expediente clínico
- Los datos del paciente son:
    - Nombre completo del paciente
    - Fecha de nacimiento
    - Padecimiento de alguna enfermedad
    - Tipo de sangre
    - Dirección
    - Número de celular
    - Correo electrónico
    - Twitter
- Consulta, por cada consulta:

    - Datos Básicos:
        - Fecha
        - Talla
        - Temperatura
        - Presión arterial
        - Frecuencia cardiaca
        - Frecuencia respiratoria
    - Observaciones
    - Medicamento y especificaciones
    - Puede agregar documentos y/o imágenes
- Médico
    - Nombre completo
    - Cédula profesional
- Administrador (Root del sistema)
    - Usuario
    - Contraseña
- Historial médico
    - Contendrá toda la información del paciente
- Asistente
    - Persona que se encargará de apoyar al médico, en el sistema puede crear citas

**Paso 2. Alinear los modelos de negocio relevantes.**

En este paso nos apoyaremos de un diagrama de entidades para comprender como se relacionan cada una de las entidades en SCMV, el diagrama de entidades puede consultarse en “ANEXO B. ENTIDADES”.

# CAPÍTULO 6. ANÁLISIS ORIENTADO A SERVICIOS, PASO 3

**Paso 3.1. Definir las necesidades de automatización del negocio.**

Para este momento este paso ya se ha visto realizado con base en la información del "Capítulo 5

Alcance y requerimientos del sistema.

**Paso 3.2. Identificar sistemas de automatización existentes.**

Este sistema que parte de cero no tiene desarrollos legados que puedan ser utilizados, pero se especifica que debe poder tener interacción con envió de correo electrónico, mensajes SMS y twitter.

Correo electrónico: Se carece de un servidor propio de correo por lo que se permitirá hacer uso del servicio de correo electrónico de Gmail.

Mensajes SMS: Las operadoras telefónicas proveen servicios de envíos de mensajes SMS a través de su infraestructura en la cual se le debe proporcionar el número celular del destinatario con el mensaje de texto que se desea enviar.

Twitter: Para poder tener interacción con mensajes a través de esta red social se puede usar una de las varias APIs disponibles, por lo que, en lo que se refiere a requerimientos tecnológicos, la integración es presumiblemente factible.

Archivo digital: Para el almacenamiento de los documentos de apoyo para el historial médico como imágenes, documentos escaneados, etc. se almacenaran en un sistema especial de archivero digital.

## 6.1.1 Paso 3.3. Modelado de servicios candidatos

**Paso 3.3.1: Descomponer el proceso de negocio.**

GUARDAR MÉDICO

- ✓ El administrador del sistema realiza el registro de un usuario tipo médico
- ✓ La información que se guarda es:
  - o Nombres
  - o Apellido paterno
  - o Apellido materno
  - o Fecha de nacimiento
  - o Cédula profesional (este es el nombre de usuario para acceder al sistema)

- correo electrónico
- número de celular
- Twitter
- Dirección:
  - Calle
  - Número exterior
  - Número interior
  - Estado
  - Ciudad o municipio
  - Colonia
  - Código postal

✓ De forma automática se genera una contraseña y se le envía por correo electrónico y mensaje SMS

OBTENER MÉDICO

✓ El usuario administrador podrá consultar la información del médico además del propio médico

ACTUALIZAR (GUARDAR) MÉDICO

✓ El administrador del sistema es el único que podrá hacer modificación en los datos generales del médico
✓ El administrador podrá modificar todos los datos, excepto cédula profesional

ELIMINAR MÉDICO

✓ El administrador del sistema es el único que puede realizar la eliminación de algún médico

GUARDAR AUXILIAR DE MÉDICO

✓ El médico es el único que puede dar de alta un auxiliar
✓ La información del auxiliar que se guarda es:
  - Nombres
  - Apellido paterno
  - apellido materno
  - fecha de nacimiento
  - correo electrónico
  - número de celular
  - Twitter
  - Dirección:
    - Calle
    - Número exterior

- Número interior
- Estado
- Ciudad o municipio
- Colonia
- Código postal

✓ Se genera automáticamente una contraseña y se le envía por correo electrónico y mensaje SMS

OBTENER AUXILIAR DE MÉDICO

✓ Solo el médico puede recuperar la información del auxiliar

ACTUALIZAR (GUARDAR) AUXILIAR DE MÉDICO

✓ El médico es el único que puede modificar la información del auxiliar

ELIMINAR AUXILIAR DE MÉDICO

✓ El médico es el único que puede eliminar a sus auxiliares

RECUPERAR CONTRASEÑAS

✓ La acción de recuperación de contraseña como tal no existe. Lo que se hace en este caso es restaurar una contraseña predeterminada, la cual se envía por correo electrónico y SMS al usuario. El administrador puede solicitar la restauración de la contraseña de los médicos, el médico puede ejecutar la restauración de contraseña de los auxiliares, el médico y el auxiliar pueden restaurar la contraseña de los pacientes

GENERAR CONTRASEÑA AUTOMÁTICAMENTE

✓ Cuando se crea un nuevo usuario en el sistema la contraseña asignada es la fecha de nacimiento con el formato: AAAAMMDD, por ejemplo:

Si nació el 5 de julio de 1989 la contraseña será: 19860505[1]

✓ Cuando ésta es generada se envía por correo electrónico y mensaje SMS

GUARDAR PACIENTE

✓ El médico registra un nuevo paciente
✓ Los datos que se guardan son:
  - Nombre
  - Apellido paterno

[1] La generación de la contraseña con el formato de la fecha de nacimiento solo es a modo de ejemplo, en caso de una implementación en producción se generaría aleatoriamente.

- Apellido materno
- Fecha de nacimiento
- Género
- Dirección:
  - Calle
  - Número exterior
  - Número interior
  - Estado
  - Ciudad o municipio
  - Colonia
  - Código postal
  - Correo electrónico
  - Número de celular
  - Twitter
  - Tipo de sangre
  - Se genera un número de Folio para el paciente, y se envía por correo electrónico y SMS, junto con la contraseña.

GUARDAR CITA

- ✓ El médico, auxiliar y paciente pueden agendar las citas, debe indicarse la fecha y hora
- ✓ El paciente puede agendar únicamente citas con su médico asignado, si la cita es creada para un usuario que no tiene registro se asigna a un consultorio y posteriormente este será asignado a un médico
- ✓ Si no es un paciente registrado en el sistema se solicita nombre completo, y alguno de los siguientes campos: número celular, twitter o correo electrónico
- ✓ Se envían notificaciones de la creación de la cita

OBTENER CITA

- ✓ El médico puede consultar la información de las citas que están a su cargo
- ✓ El auxiliar puede consultar las citas que están registradas en el consultorio en el que labora
- ✓ El paciente puede revisar la información de sus citas

ELIMINAR CITA

- ✓ El médico, el auxiliar o el propio paciente pueden realizar la cancelación de una cita

GUARDAR CONSULTA

- ✓ Las consultas solo pueden hacerse a pacientes registrados en el sistema
- ✓ El médico registra los datos vitales, observaciones

- ✓ Ingresa los medicamentos e indicaciones de la prescripción médica
- ✓ Opcionalmente puede agregar una solicitud de estudio
- ✓ Opcionalmente puede agregar documentos digitales
- ✓ Cuando se concluye la consulta el médico puede imprimirla y de forma automática se envía una copia digital por correo electrónico al paciente

AGREGAR SOLICITUD DE ESTUDIOS (OPCIONAL)

- ✓ Si el médico requiere mandar a realizar estudios clínicos al paciente
- ✓ Nombres de los estudios que deben realizarse

AGREGAR DOCUMENTOS (OPCIONAL)

- ✓ Durante la consulta el médico puede agregar distintos tipos de archivos al historial
- ✓ Los archivos son enviados al servicio de archivero virtual que van ligados de acuerdo al paciente y la consulta

ENVIAR NOTIFICACIÓN Y RECORDATORIOS

- ✓ Un proceso busca las citas con fecha posterior a "N" días, por cada una se envía una notificación al paciente para recordar la fecha y hora de la cita

**Paso 3.3.2: Identificar operaciones de los servicios candidatos.**

GUARDAR MÉDICO

- ✓ El administrador del sistema realiza el registro de un usuario tipo médico. (PASO MANUAL DEL ADMINISTRADOR)
- ✓ La información que se guarda es:
    - o Nombres
    - o Apellido paterno
    - o Apellido materno
    - o Fecha de nacimiento
    - o Cédula profesional (este es el nombre de usuario para acceder al sistema)
    - o Correo electrónico
    - o Número de celular
    - o Twitter
    - o Dirección:
        - ▪ Calle
        - ▪ Número exterior
        - ▪ Número interior
        - ▪ Estado

        - Ciudad o municipio
        - Colonia
        - Código postal
- ✓ De forma automática se genera una contraseña y se le envía por correo electrónico y mensaje SMS

OBTENER MÉDICO

- ✓ El usuario administrador podrá consultar la información del médico además del propio médico (NO APORTA NEGOCIO)

ACTUALIZAR (GUARDAR) MÉDICO

- ✓ El administrador del sistema es el único que podrá hacer modificación en los datos generales del médico. (PASO MANUAL)
- ✓ El administrador podrá modificar todos los datos, excepto cédula profesional

ELIMINAR MÉDICO

- ✓ El administrador del sistema es el único que puede realizar la eliminación de algún médico

GUARDAR AUXILIAR DE MÉDICO

- ✓ El médico es el único que puede dar de alta un auxiliar (PASO MANUAL DEL MÉDICO)
- ✓ La información del auxiliar que se guarda es:
    - o Nombres
    - o Apellido paterno
    - o Apellido materno
    - o Fecha de nacimiento
    - o Correo electrónico
    - o Número de celular
    - o Twitter
    - o Dirección:
        - Calle
        - Número exterior
        - Número interior
        - Estado
        - Ciudad o municipio
        - Colonia
        - Código postal
- ✓ Se genera una contraseña y se le envía por correo electrónico y mensaje SMS

OBTENER AUXILIAR DE MÉDICO

- ✓ Solo el médico puede recuperar la información del auxiliar. (NO APORTA NEGOCIO)

ACTUALIZAR (GUARDAR) AUXILIAR DE MÉDICO

- ✓ El médico es el único que puede modificar la información del auxiliar (NO APORTA NEGOCIO)

ELIMINAR AUXILIAR DE MÉDICO

- ✓ El médico es el único que puede eliminar a sus auxiliares

RECUPERAR CONTRASEÑAS

- ✓ La acción de recuperación de contraseña como tal no existe. Lo que se hace en este caso es restaurar una contraseña predeterminada, la cual se envía por correo electrónico y SMS al usuario. El administrador puede solicitar la restauración de la contraseña de los médicos, el médico puede ejecutar la restauración de contraseña de los auxiliares, el médico y el auxiliar pueden restaurar la contraseña de los pacientes

GENERAR CONTRASEÑA AUTOMÁTICAMENTE

- ✓ Cuando se crea un nuevo usuario en el sistema la contraseña asignada es la fecha de nacimiento con el formato: AAAAMMDD, por ejemplo:

  Si nació el 5 de julio de 1989 la contraseña será: 19860505
- ✓ Cuando esta es generada se envía por correo electrónico y mensaje SMS

GUARDAR PACIENTE

- ✓ El médico registra un nuevo paciente
- ✓ Los datos que se guardan son:
  - o Nombre
  - o Apellido paterno
  - o Apellido materno
  - o Fecha de nacimiento
  - o Género
  - o Dirección:
    - ▪ Calle
    - ▪ Número exterior
    - ▪ Número interior
    - ▪ Estado
    - ▪ Ciudad o municipio
    - ▪ Colonia
    - ▪ Código postal

- Correo electrónico
- Número de celular
- Twitter
- Tipo de sangre
- Se genera un número de Folio para el paciente, y se le envía por correo electrónico y SMS, junto con la contraseña

GUARDAR CITA

- ✓ El médico, auxiliar y paciente pueden agendar las citas (NO APORTA NEGOCIO), debe indicarse la fecha y hora
- ✓ El paciente puede agendar únicamente citas con su médico asignado, si la cita es creada para un usuario que no tiene registro se asigna a un consultorio y posteriormente este será asignado a un médico
- ✓ Si no es un paciente registrado en el sistema se solicita nombre completo, y alguno de los siguientes campos: número celular, twitter o correo electrónico
- ✓ Se envían notificaciones de la creación de la cita

OBTENER CITA

- ✓ El médico puede consultar la información de las citas que están a su cargo
- ✓ El auxiliar puede consultar las citas que están registradas en el consultorio en el que labora
- ✓ El paciente puede revisar la información de sus citas

ELIMINAR CITA

- ✓ El médico, el auxiliar o el propio paciente pueden realizar la cancelación de una cita

GUARDAR CONSULTA

- ✓ Las consultas solo pueden hacerse a pacientes registrados en el sistema (TAREA MANUAL DEL MÉDICO)
- ✓ El médico registra los datos vitales, observaciones (TAREA MANUAL DEL MÉDICO)
- ✓ Ingresa los medicamentos con respectivas indicaciones (TAREA MANUAL DEL MÉDICO)
- ✓ Opcionalmente puede agregar una solicitud de estudio
- ✓ Opcionalmente puede agregar documentos digitales
- ✓ Cuando se concluye la consulta el médico puede imprimirla y de forma automática se envía una copia digital por correo electrónico al paciente

AGREGAR SOLICITUD DE ESTUDIOS (OPCIONAL)

- ✓ Si el médico requiere mandar a realizar estudios clínicos al paciente
- ✓ Nombres de los estudios que deben realizarse

AGREGAR DOCUMENTOS (OPCIONAL)

- ✓ Durante la consulta el médico puede agregar distintos tipos de archivos al historial (NO APORTA NEGOCIO)
- ✓ Los archivos son enviados al servicio de archivero virtual que van ligados de acuerdo al paciente y la consulta

ENVIAR NOTIFICACIÓN Y RECORDATORIOS

- ✓ Un proceso busca las citas con fecha posterior a "N" días, por cada una se envía una notificación al paciente para recordar la fecha y hora de la cita (NO APORTA NEGOCIO PARA SERVICIO)

**Paso 3.3.3: Abstraer la lógica de orquestación.**

Del administrador.

- El administrador es el único que puede registrar, modificar y eliminar información de los médicos, cuando se modifica la información se debe enviar notificación al médico
- El administrador es el único que puede crear, modificar y eliminar información de los consultorios.
- Restaurar contraseñas de médicos
- Cambiar su propia contraseña

De los médicos.

- Únicamente pueden interactuar con los pacientes que tienen asociados
- Es el encargado de registrar, modificar o eliminar información de sus auxiliares
- Restaurar contraseñas de pacientes y auxiliares
- Cambiar su propia contraseña

De los auxiliares del médico.

- Pueden crear, modificar, eliminar citas, restaurar contraseñas de los pacientes
- Cambiar su propia contraseña

De los pacientes.

- Cuando se registra un nuevo paciente se debe crear también el historial y con ello el registro de documentos, se auto genera una contraseña y se envía notificaciones al paciente
- Pueden crear, modificar, eliminar citas

- Consultar su expediente, consultas y citas
- Puede crear y eliminar citas
- Cambiar su propia contraseña

De la creación de citas.

- Esté o no esté registrado el paciente se le envía notificación para la cita y posteriormente se le enviará recordatorio

De la creación de una consulta.

- Únicamente el médico puede crear consulta a los pacientes asignados, en ella se puede agregar documentos y solicitudes de estudios, al finalizarla se envía automáticamente por correo electrónico

De la generación de la contraseña.

- Cada que se registra un usuario nuevo en el sistema se autogenera una contraseña y se le envía por correo electrónico y/o mensaje SMS
- La contraseña inicial es la fecha de nacimiento con el formato AAAAMMDD

**Paso 3.3.4: Crear servicios de negocio candidatos.**

| Servicios candidatos | |
|---|---|
| **Servicio** | **operaciones** |
| Servicio de administración de usuarios | • Obtener médicos<br>• Guardar médico (también actualiza)<br>• Eliminar médico<br>• Obtener pacientes<br>• Guardar paciente<br>• Eliminar paciente<br>• Obtener auxiliares<br>• Guardar auxiliar<br>• Eliminar auxiliar<br>• Reiniciar contraseña<br>• Cambiar contraseña |

| | |
|---|---|
| | • Obtener usuario<br>• Autenticar usuario |
| Servicio de administración de expediente | • Guardar expediente<br>• Buscar expediente<br>• Guardar consulta<br>• Obtener consultas por paciente<br>• Obtener solicitud de estudio<br>• Obtener consulta |
| Servicio de citas | • Guardar cita, usuario registrado<br>• Guardar cita, usuario no registrado<br>• Obtener citas<br>• Eliminar cita<br>• Buscar citas future |
| Servicio de consultorios | • Obtener consultorios<br>• Obtener consultorio<br>• Guardar consultorio<br>• Eliminar consultorio |
| Servicio de catálogos | • Obtener tipo sangre<br>• Obtener tipos sangre<br>• Obtener tipo estudio<br>• Obtener tipos estudio<br>• Obtener medicamento<br>• Obtener medicamentos<br>• Obtener enfermedad<br>• Obtener enfermedades |
| Servicio de utilerías | • Fecha a String<br>• Generar número aleatorio |

| | |
|---|---|
| Servicio de mensajería | • Enviar correo electrónico<br>• Enviar Twitt<br>• Enviar SMS |
| Servicio de documentos | • Enviar archivo<br>• Recuperar archivo |
| Batch*<br>* Batch como tal no es un servicio pero hay que recordar que es funcionalidad dentro del sistema | • Enviar consultas<br>• Enviar recordatorio cita |

**Tabla 6-1Servicios candidatos.**

**Paso 3.3.5: Refinar y aplicar los principios de orientación a servicios.**

| Servicios candidatos | |
|---|---|
| **Servicio** | **operaciones** |
| Servicio de administración de usuarios | • Obtener médicos<br>• Guardar médico (también actualiza)<br>• ~~Eliminar médico~~<br>• Obtener pacientes<br>• Guardar paciente (también actualiza)<br>• ~~Eliminar paciente~~<br>• Obtener auxiliares<br>• Guardar auxiliar (también actualiza)<br>• ~~Eliminar auxiliar~~<br>• Reiniciar contraseña<br>• Cambiar contraseña<br>• Obtener usuario (capaz de obtener administrador, médico, auxiliar y paciente) |

| | |
|---|---|
| | • Autenticar usuario |
| Servicio de administración de expediente | • Guardar expediente (también actualiza)<br>• Buscar expediente<br>• Guardar consulta<br>• Obtener consultas por paciente<br>• Obtener solicitud de estudio<br>• Obtener consulta |
| Servicio de citas | • Guardar cita, usuario registrado<br>• Guardar cita, usuario no registrado<br>• Obtener citas<br>• Eliminar cita<br>• Buscar citas futuro |
| Servicio de consultorios | • Obtener consultorios<br>• Obtener consultorio<br>• Guardar consultorio<br>• Eliminar consultorio |
| Servicio de catálogos | • Obtener tipo sangre<br>• Obtener tipos sangre<br>• Obtener tipo estudio<br>• Obtener tipos estudio<br>• Obtener medicamento<br>• Obtener medicamentos<br>• Obtener enfermedad<br>• Obtener enfermedades |
| Servicio de utilerías | • Fecha a String<br>• Generar número aleatorio |
| Servicio de mensajería | • Enviar correo electrónico |

| | |
|---|---|
| | • Enviar Twitt.<br>• Enviar SMS. |
| Servicio de documentos | • Enviar archivo<br>• Recuperar archivo |
| Servicio de Batch* | • Enviar consultas<br>• Enviar recordatorio cita |

Tabla 6-2 Servicios candidatos refinados.


Ilustración 6-1 Servicios del SCMV.

**Paso 3.3.6: Identificar composiciones de servicios candidatos.**

Siguiendo con el paso anterior encontramos que hay tres servicios de aplicación candidatos.

- Servicios de mensajería
- Servicio de documentos
- Servicio de utilerías

Los demás servicios son servicios de negocio que estarán compuestos entre ellos mismos y por servicios de aplicación.

- Servicio de administración de usuarios
- Servicio de expediente
- Servicio de citas
- Servicio de consultorios
- Servicio de catálogos

De acuerdo a las operaciones de cada uno de los servicios se puede determinar la siguiente composición, ilustrada en dos imágenes.

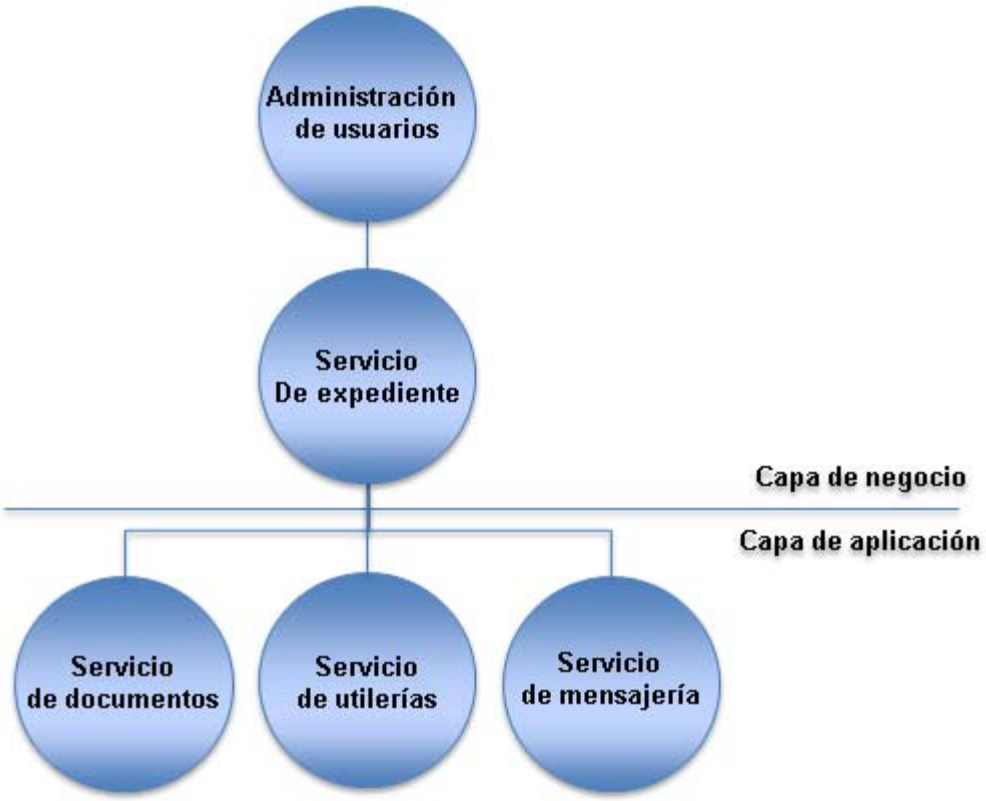


**Ilustración 6-2 Composición del servicio administración de usuarios.**

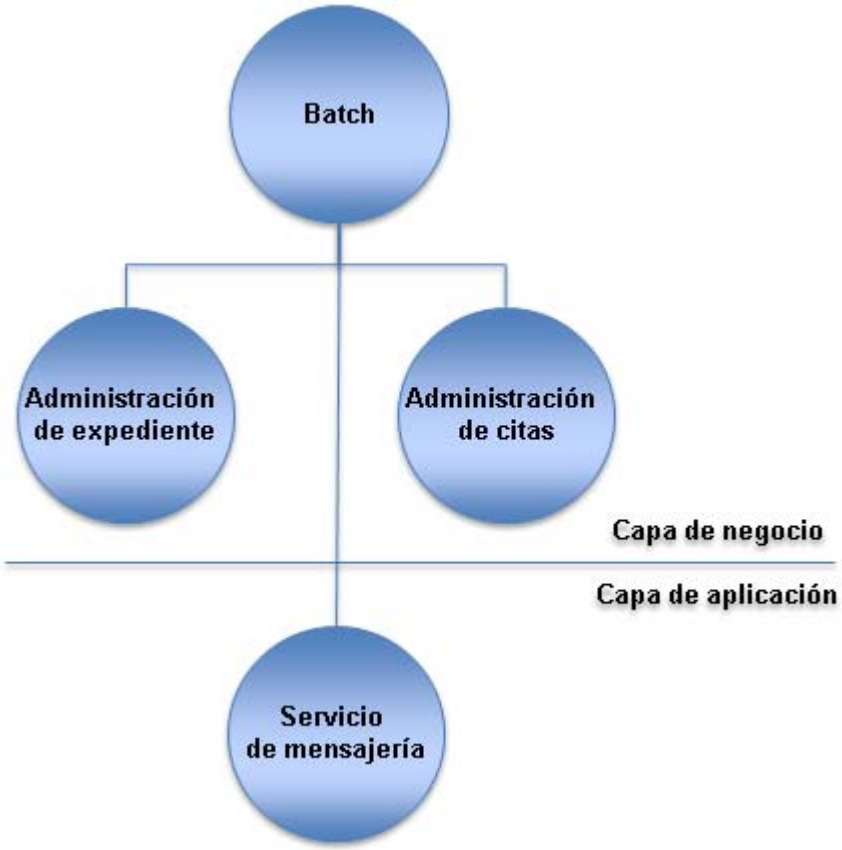


Ilustración 6-3 Composición del batch.

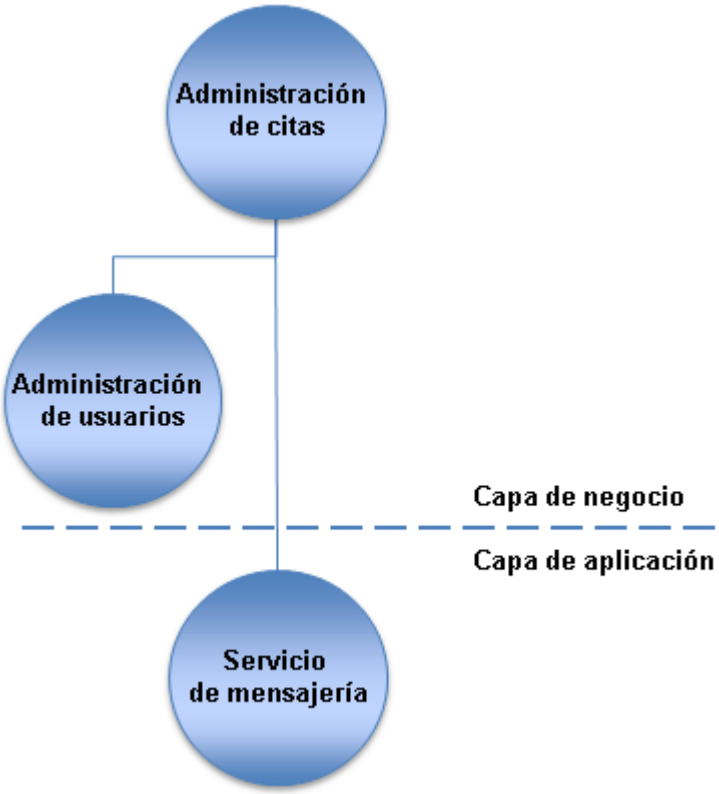


Ilustración 6-4 Composición del servicio administración de citas.

**Paso 3.3.7: Revisar la agrupación de las operaciones de servicio de negocio.**

Para este caso no hay la necesidad de reagrupar las operaciones de los servicios de negocio, por lo que se puede continuar con el siguiente paso.

**Paso 3.3.8: Analizar los requerimientos de procesamiento de la aplicación.**

En este paso se debe hacer un pequeño análisis en el que se debe tomar en cuenta lo siguiente:

- ¿Qué lógica subyacente necesita ser ejecutada para procesar la acción descrita por la operación candidata?
  De alguna manera ya se ha indicado en los inicios de este capítulo el cómo debe ejecutarse la lógica del sistema
- La lógica de la aplicación ya existe o se requiere un nuevo desarrollo.
  En capítulos anteriores ya se ha revisado este punto, en el que se especifica que todo el desarrollo es nuevo
- Si la lógica requerida extiende lo límites de la aplicación. Es decir, ¿se requiere más que un sistema para realizar esta acción?

De igual forma ya se ha visto que otros sistemas o servicios extra serán requeridos para completar el alcance del sistema, los servicios externos son correo electrónico, mensajes SMS y de twitter

**Paso 3.3.9 al 3.3.12:**

La identificación de las operaciones, creación de servicios candidatos, revisión de composiciones y agrupaciones de los servicios candidatos ya han sido realizados a la par de pasos previos.

**Paso 3.3.13 Mantener un inventario de servicios candidatos (opcional)**

El proceso que se estado trabajando ha iniciado de cero con lo que no se tenía un inventario de servicios, el cual puede empezar a generarse una vez que tenemos una primera iteración de modo que pueda ser útil para posibles reutilizaciones de funcionalidad y esto ayude en el ahorro de tiempo y costos.

# CAPÍTULO 7. DISEÑO DEL SISTEMA, PASO 4

En este capítulo se desarrolla el paso 5 descrito en la metodología MSOAM para el SCMV. A continuación se describen las capas a utilizarse, así como el diseño de cada uno de los servicios, de entidad, de negocio y de aplicación.

## 7.1 Paso 4.1: *Compose* SOA

**Paso 4.1.1. Seleccionar capas de servicio.**

En una arquitectura SOA el mayor peso de funcionalidad se ubica en el llamado “backend”, el cual típicamente provee de servicios a clientes diversos, tales como: aplicaciones web, aplicaciones de escritorio, servicios web, o aplicaciones móviles. El SCMV tiene como cliente primordial una aplicación web.

Más adelante, en el "Capítulo 8

Tecnologías utilizadas., se detallarán las tecnologías con las que se implementa la arquitectura propuesta. Cabe mencionar que la orquestación de los servicios se logró con las mismas tecnologías que los servicios de tarea, y no a través de herramientas especializadas esto debido que las necesidades funcionales del sistema son cubiertas con estas mismas tecnologías.

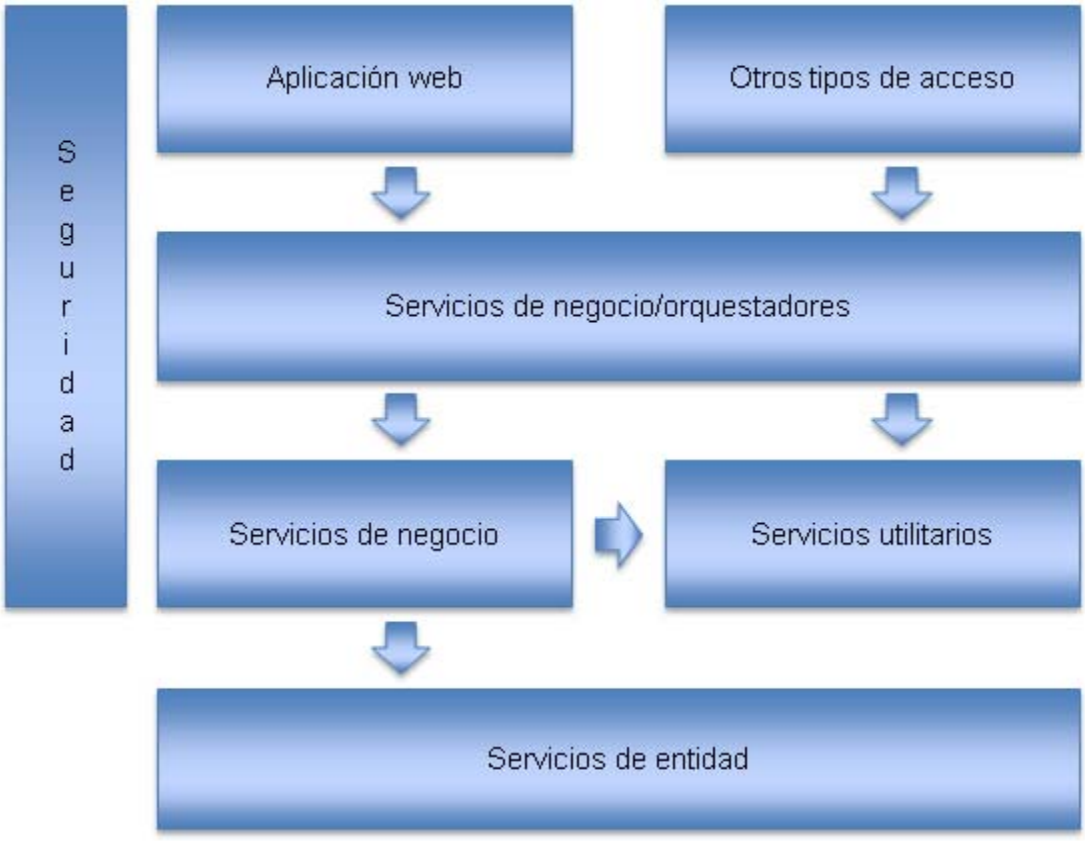


Ilustración 7-1 Diagrama SCMV.

Debido al alcance y objetivos del proyecto de tesis, el desarrollo se hizo en un solo proyecto que contiene al acceso web y la lógica de negocio, sin que con ello se transgredan los principios de una arquitectura SOA.

**Pasos 4.1.2 y 4.1.3: Estándares y extensiones SOA.**

Estos pasos son tratados más adelante, a lo largo del "CAPÍTULO 8. DESARROLLO DEL SISTEMA.

## 7.2 Paso 4.2: Diseño de servicios de entidad

A continuación se listan las entidades definidas para el sistema.

- Auxiliar
- Enfermedad, catálogo
- Medicamento, catálogo

- Tipo estudio, catálogo
- Tipo sangre, catálogo (catálogo no administrable)
- Cita
- Consulta
- Consultorio
- Datos contacto
- Dirección
- Expediente
- Médico
- Paciente
- Prescripción
- Usuario app
- Rol (catálogo no administrable)

Algunas relaciones se manejaron como entidades debido a que tenían atributos extra a las de las entidades relacionadas, por ejemplo el estatus.

- UsuarioRol
- RSolicitudEstudios

A continuación se muestra un diagrama entidad relación, para mayor detalle de las entidades puede consultar el “ANEXO B. ENTIDADES”.

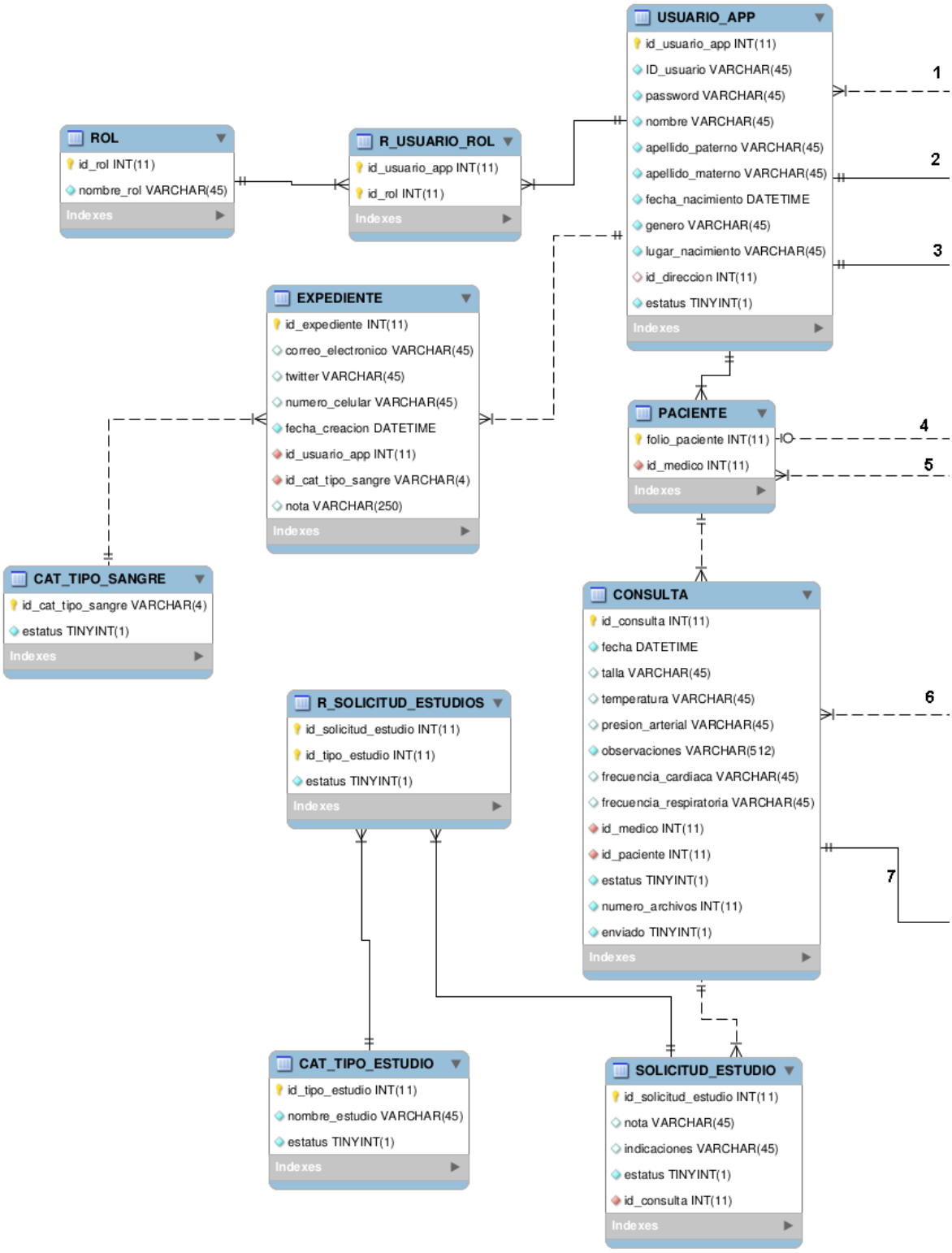


**Ilustración 7-2 Diagrama de entidades del SCMV parte 1.**

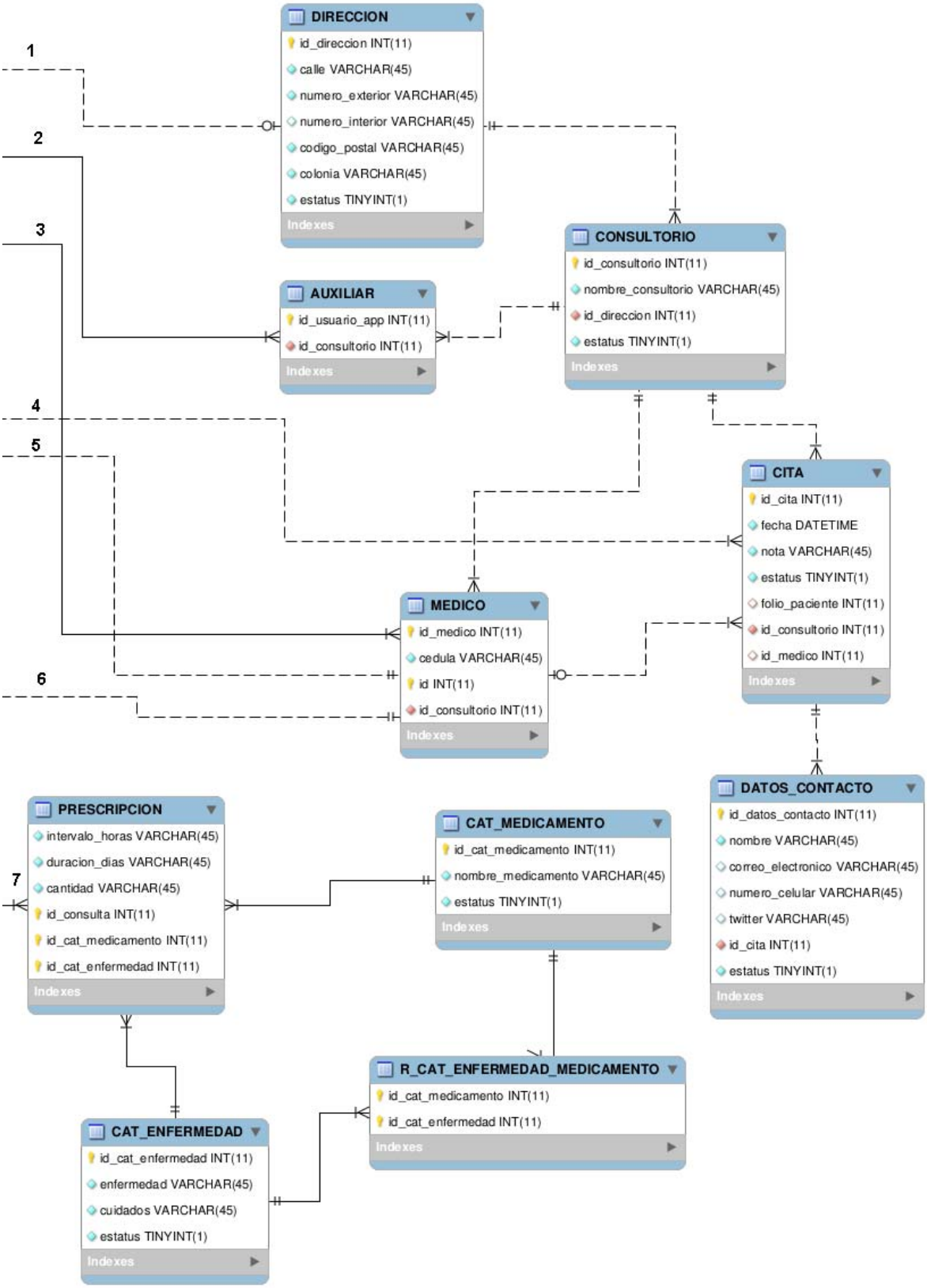


**Ilustración 7-3 Diagrama de entidades del SCMV parte 2.**

**Paso 4.2.1: Revisar servicios existentes.**

Al tratarse de un desarrollo nuevo, no existen servicios de entidad previamente construidos.

**Paso 4.2.2: Define the message schema types.**

Este paso se trata junto con el paso siguiente.

**Paso 4.2.3: Especificar una interfaz inicial del servicio.**

Se mostrarán dos interfaces de servicios de entidad a modo de ejemplo, todas las especificaciones de las interfaces pueden ser consultadas en ANEXO C. SERVICIOS DE ENTIDAD.

| **Servicio entidad tipo estudio** | | | |
|---|---|---|---|
| **Nombre del servicio** | CatTipoEstudioDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.CatTipoEstudioDAO | | |
| | | | |
| **Operación** | **obtenerTipoEstudio** | | |
| **Descripción** | Obtiene del catálogo de Tipo Estudio el estudio indicado por el ID | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idTipoEstudio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | CatTipoEstudio | | |
| | | | |
| **Operación** | **obtenerTipoEstudios** | | |
| **Descripción** | Obtiene del catálogo de Tipo Estudio todos los tipos de estudios con estatus activo, el primer campo indica la propiedad por la que serán ordenadas y el segundo si es verdadero (true) indica si el orden es ascendente, de lo contrario será descendente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |

| | | | |
|---|---|---|---|
| | Propiedad | String | No |
| | Ascendente | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatTipoEstudio> | | |

**Tabla 7-1 Servicio entidad tipo estudio.**

| **Servicio entidad consulta** | | | |
|---|---|---|---|
| **Nombre del servicio** | **ConsultaDAO** | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.ConsultaDAO | | |
| | | | |
| **Operación** | **obtenerConsultas** | | |
| **Descripción** | Obtiene las consultas de un paciente dado | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idPaciente | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Consulta> | | |
| | | | |
| **Operación** | **obtenerIdConsultaNoEnviada** | | |
| **Descripción** | Obtiene los IDs de las consultas que no han sido enviadas por correo electrónico a los pacientes, si es que se tiene el correo del paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | N/A | | |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Integer> | | |
| | | | |
| **Operación** | **guardarActualizar** | | |

| **Descripción** | Guarda o actualiza la información de una consulta realizada por el médico | | |
|---|---|---|---|
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Consulta | Consulta | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **obtenerPorId** | | |
| **Descripción** | Recupera la entidad Consulta buscándola por su ID. | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsulta | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | Consulta | | |

**Tabla 7-2 Servicio entidad consulta.**

**Paso 4.2.4: Aplicar principios de orientación a servicios.**

De los principios básicos se tiene:

- Los servicios de entidad son claramente reutilizables pues son requeridos para registrar un paciente y para obtener los datos de contacto para enviarle información.
- El diseño propuesto incluye dependencias de conexión a la base de datos, lo cual es típico de este tipo de aplicaciones. Desde este punto de vista se cumple con el principio de autonomía
- No necesitan guardar información temporal de su estado actual que deba ser necesaria para una posterior invocación

**Paso 4.2.5: Estandarizar y refinar la interfaz del servicio.**

En el paso 3 donde se propone la interfaz inicial ya se están empleando lineamientos en la elección de nombres, éstos se especifican en “ANEXO D. CONVENCIONES PARA ELECCIÓN DE NOMBRES”.

**Paso 4.2.6: Extender el diseño del servicio.**

En los servicios ejemplificados no es necesario agregar nuevas operaciones. En la primera revisión se decide que ya se han identificado todas las operaciones que podrían ser utilizadas en la aplicación.

**Paso 4.2.7: Identificar el procesamiento requerido.**

Los servicios ilustrados cumplen completamente con las necesidades básicas de las operaciones CRUD requeridas.

## 7.3 Paso 4.3: Diseño de servicios de aplicación

**Paso 4.3.1: Revisar servicios existentes.**

Este proyecto al ser un desarrollo nuevo no existen servicios de entidad construidos.

Los servicios de aplicación identificados son:

- Servicio de utilerías
  Este servicio realiza operaciones que no son propias del negocio, entre las acciones que realiza son validaciones de correo electrónico, validar campos alfanuméricos, numéricos, convertir fechas a cadenas de texto.

- Servicios de mensajería. Consume el Servicio de mensajería
  Este servicio se encarga de recibir información del SCMV, le realiza una transformación para que pueda ser enviada al servicio de mensajería. Por ejemplo recibe la entidad de Consulta y Expediente, obtiene el correo electrónico del paciente, convierte la entidad Consulta a texto plano para que pueda ser enviada como cuerpo del correo y la envía el servicio de mensajería.

- Servicio de documentos. Consume el Servicio de documentos
  De igual forma que el servicio de aplicación de mensajería del SCMV adapta la información y la envía al servicio de mensajería (el servicio web simulando un servicio externo). Para almacenar la foto de perfil se requiere còmo parámetro el nombre de usuario del paciente, para almacenar los archivos de las consultas requiere los archivos, el nombre de usuario del paciente y el identificador de la consulta a la que pertenecen.

Cuando necesita recuperar información es necesario indicar el nombre de usuario, puede agregar el número de consulta y recupera todos los archivos pertenecientes a la consulta o al perfil, hace una transformación y la regresa al SCMV para que este la pueda interpretar.

**Paso 4.3.2: Confirmar el contexto.**

Respecto a los servicios de mensajería y documentos únicamente realizan operaciones de su ámbito por lo que no hay ninguna duda del contexto de sus operaciones. El servicio de utilerías se le dio un nombre un tanto general, debido a que contendrá operaciones simples y de utilería, por ejemplo, generar la cadena de contraseña inicial para los usuarios.

**Paso 4.3.3: Especificar una interfaz inicial del servicio.**

| SERVICIO DE MENSAJERÍA | | | |
|---|---|---|---|
| **Nombre del servicio** | MensajeriaIntegracion | | |
| **Ubicación Interfaz** | mx.egc.scmv.integracion.MensajeriaIntegracion | | |
| | | | |
| **Operación** | **notificarCita** | | |
| **Descripción** | Enviar notificación de que se ha agendado una cita para un paciente que no tiene registro previo en el sistema<br>Se envía correo electrónico, twitter y SMS | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | DatosContacto | DatosContacto | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **notificarCita** | | |
| **Descripción** | Enviar notificación de que se ha agendado una cita para un paciente que ya tiene registro previo en el sistema<br>Se envía correo electrónico, twitter y SMS | | |

| Mensaje de entrada | Nombre del atributo | Tipo de dato | Opcional |
|---|---|---|---|
| | Cita | Cita | No |
| | Expediente | Expediente | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **registroNuevoUsuario** | | |
| **Descripción** | Envía una notificación a un usuario de que se le ha registrado en el sistema<br>Se envía correo electrónico, twitter y SMS | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | usuarioApp | UsuarioApp | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **reinicioContrasenia** | | |
| **Descripción** | Envía una notificación al usuario de que su contraseña se ha reiniciado<br>Se envía correo electrónico, twitter y SMS | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | usuarioApp | UsuarioApp | No |
| | Expediente | Expediente | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **enviarTwitter** | | |
| **Descripción** | Envia un mensaje por twiiter | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |

| | | | |
|---|---|---|---|
| | Destino | String | No |
| | Mensaje | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **enviarCorreo** | | |
| **Descripción** | Envia un correo electrónico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Destinatario | String | No |
| | Asunto | String | No |
| | Mensaje | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **enviarConsulta** | | |
| **Descripción** | Envía por correo electrónico la información de la consulta clínica | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Expediente | Expediente | No |
| | Consulta | Consulta | No |
| | Prescripciones | List<Prescripcion> | Sí |
| | Estudio | SolicitudEstudio | Sí |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla 7-3 SERVICIO DE MENSAJERÍA.**

| SERVICIO DE DOCUMENTOS | | | |
|---|---|---|---|
| **Nombre del servicio** | DocumentosIntegracion | | |
| **Ubicación Interfaz** | mx.egc.scmv.integracion.DocumentosIntegracion | | |
| | | | |
| **Operación** | **enviarArchivo** | | |
| **Descripción** | Guarda un archivo en el archivero digital | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Ruta | String | No |
| | Nombre | String | No |
| | Archivo | byte[] | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **recuperarArchivo** | | |
| **Descripción** | Recupera un archivo en el archivero digital | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Ruta | String | No |
| | Nombre | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | byte[] | | |
| | | | |
| **Operación** | **obtenerImagenPerfil** | | |
| **Descripción** | Obtiene la imagen de perfil de un usuario | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Usuario | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | byte[] | | |

| | | | |
|---|---|---|---|
| **Operación** | **guardarArchivoExpediente** | | |
| **Descripción** | Guarda un conjunto de archivos relacionados a una consulta | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Ruta | String | No |
| | Nombre | String | No |
| | Archivos | List<byte[]> | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **obtenerArchivosConsulta** | | |
| **Descripción** | Obtiene todos los archivos que pertenecen a una consulta de un paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Consulta | consulta | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<byte[]> | | |
| | | | |
| **Operación** | **guardarArchivoExpediente** | | |
| **Descripción** | Guarda un archivo perteneciente a una consulta | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Ruta | String | No |
| | nombreArchivo | String | No |
| | Archivo | byte[] | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla 7-4 SERVICIO DE DOCUMENTOS.**

**Paso 4.3.4: Aplicar principios de orientación a servicios.**

Al evaluar los servicios propuestos respecto de los cuatro principios básicos de orientación a servicios se identifica lo siguiente:

Los servicios de entidad son reutilizables ya que son consumidos para registrar un paciente y para obtener sus datos de contacto para enviarle información.

De acuerdo al diseño la única dependencia es la conexión a la base de datos por lo que su autonomía es bastante grande

No necesitan guardar información temporal de su estado actual que deba ser necesaria para una posterior invocación.

| Servicios de aplicación | |
|---|---|
| **Servicio** | **Nota** |
| Utilerías | La operación de generar una cadena de texto de cierto tamaño es una operación muy genérica<br><br>Su autonomía es grande pues no tiene dependencia subyacente |
| Mensajería | Este servicio pueden ser utilizado por algún otro cliente que desee enviar mensajes por los medios disponibles, únicamente hay que registrar parámetros desde la cuenta que serán enviados los mensajes<br><br>De alguna forma no es tan autónomo ya que depende de que los servicios externos estén disponibles, sin embargo, debe contemplarse que estos servicios podrán reanudar la comunicación sin perder los mensajes que deben entregar sin eliminar el principio de no guardar estado |
| Documentos | Puede llegar a ser utilizado por otro sistema que desee guardar documentos, pues no se centra solo en documentos médicos, únicamente habrá que tener derechos de escritura o lectura a su “archivero”<br><br>Propiamente este servicio será el encargado de almacenar la información por lo que no perderá autonomía |

**Tabla 7-5 Servicios de aplicación.**

**Paso 4.3.5: Estandarizar y refinar la interfaz del servicio.**

En el paso 3 donde se propone la interfaz inicial ya se están empleando lineamientos en la elección de nombres, éstos se especifican en “ANEXO D. CONVENCIONES PARA ELECCIÓN DE NOMBRES”.

**Paso 4.3.6: Especular características funcionales al servicio.**

En el paso 3 donde se propone la interfaz inicial ya se están empleando lineamientos en la elección de nombres, éstos se especifican en “ANEXO D. CONVENCIONES PARA ELECCIÓN DE NOMBRES”.

**Paso 4.3.7: Identificar limitaciones técnicas.**

- Servicio de utilitarias
  A primera vista no se identifican problemas de impacto, pues es una operación sencilla sin dependencias internas o externas.
- Servicios de mensajería
  Debe tener la capacidad de no perder la información que será enviada en caso de que los servicios externos tenga fallas, para esto puede almacenar temporalmente los mensajes que debe enviar, así en caso de problemas tendrá la información disponible para un futuro intento.
- Servicio de documentos
  El principal problema es quedarse sin espacio del almacenamiento para ello se requerirá un equipo con suficiente espacio de almacenamiento, y probablemente en un futuro se haga una adaptación al desarrollo interno del servicio para que pueda manejar varias unidades de almacenamiento sin que estas repercutan en los clientes del servicio.

## 7.4 Paso 4.4: Diseño de servicios de negocio o de tarea

Este caso solo se ejemplificará una operación de un servicio de tarea, **Enviar prescripción médica,** Esta operación pertenece al **servicio de expediente**, su función es obtener la información de la consulta realizada al paciente, obtener su correo electrónico y enviarla.

**Paso 4.4.1: Definir la lógica del flujo de trabajo (workflow).**

Proceso para enviar la prescripción médica por correo.

Parámetros de entrada:

- ✓ Identificador de consulta

Proceso:

1. Recuperar la información de la consulta; si no existe la consulta avisa que no existe
2. Recuperar las prescripciones (indicaciones de medicamento existentes)
3. Obtener las solicitudes de estudio en caso de existir
4. Recuperar datos del usuario y expediente
5. Si no se tiene registrado un correo electrónico, termina el proceso y avisa que no tiene correo electrónico
6. La información es preparada para ser enviada
7. Se hace una invocación asíncrona al servicio de mensajería

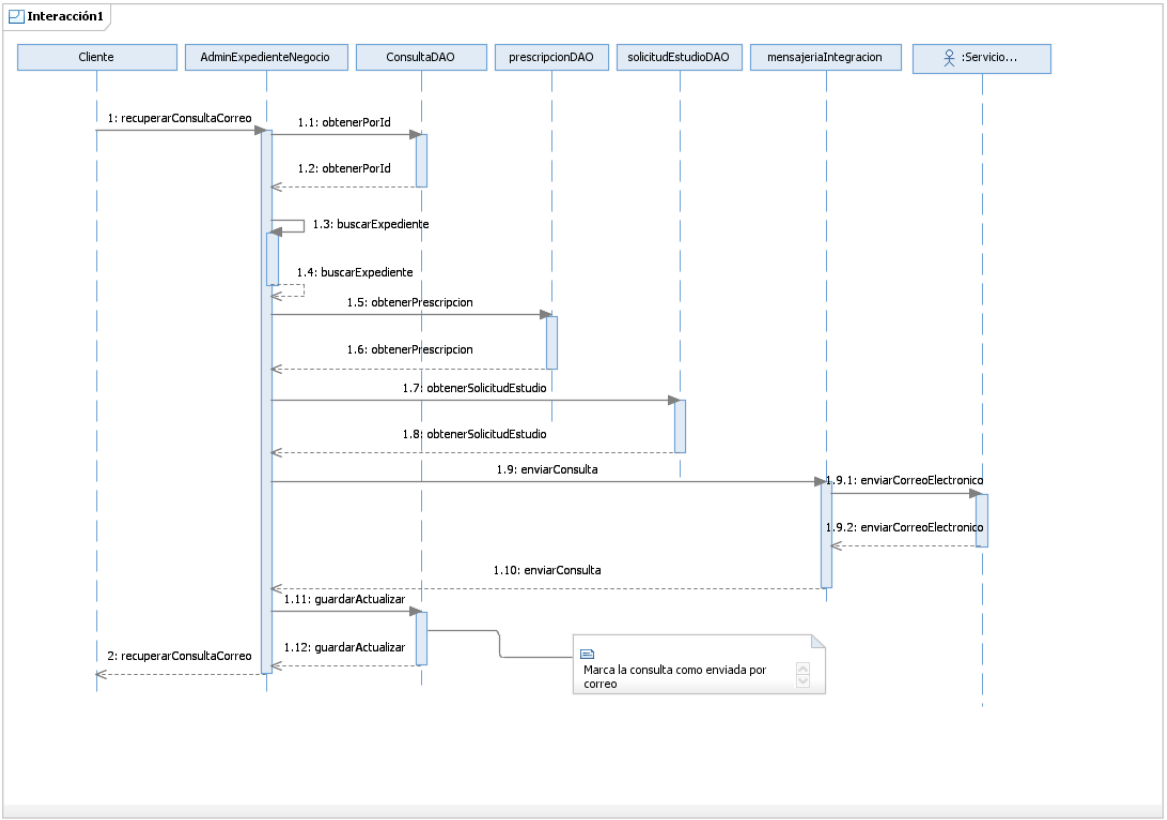


**Ilustración 7-4 Enviar prescripción médica**

El SCMV está conformado por cinco servicios de negocio:

- Administración de consultorios

  Este servicio se encarga de hacer las operaciones de altas, bajas, actualización de los consultorios, y únicamente puede ser utilizado por el administrador general del sistema.

- Administración de usuarios
  Este servicio se encarga de la administración de los usuarios del sistema, médicos, auxiliares, pacientes, permite la validación de los usuarios y cambios de contraseña.
- Administración de expediente
  El servicio de expediente se encarga de llevar el seguimiento de los pacientes, las consultas realizadas, indicaciones, las solicitudes de estudios y los documentos registrados en las consultas.
- Administración de citas
  El objetivo de este servicio es llevar el registro de las citas solicitadas ya sean citas para los usuarios con registro o pacientes que aún no están registrados en el sistema, recuperar las citas de un paciente, las asignadas a un médico o a un consultorio.
- Administración de catálogos
  Este servicio cubre la administración de los catálogos del sistema, para lo cual en total son cinco catálogos pero solo tres de ellos son administrables: enfermedad, medicamento y tipo de estudio, los otros dos catálogos que no son administrables son: rol y tipo de sangre.

**Batch**

- Esta funcionalidad fue implementada para buscar las citas programadas que tienen fecha posterior al día actual y se envíen recordatorios al paciente. En el mismo instante el procesamiento de batch busca las Consultas que no pudieron ser enviadas por correo electrónico en el momento, una vez identificadas vuelve a tratar de enviarlas

**Paso 4.4.2: Especificar una interfaz inicial del servicio.**

| **SERVICIOS DE ADMINISTRACIÓN DE CITAS.** | | | |
|---|---|---|---|
| **Nombre del servicio** | Administración de citas | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.negocio.AdminCitasNegocio | | |
| | | | |
| **Operación:** | **GuardarCitaUsuarioConRegistro** | | |
| **Descripción** | Esta operación permite al paciente agendar una cita con el médico, una vez guardada envía una notificación por mensajería | | |
| **Roles** | Paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Cita | Cita | No |

| **Mensaje de salida** | **Tipo de dato** | | |
|---|---|---|---|
| | N/A | | |
| | | | |
| **Operación** | **GuardarCitaUsuarioSinRegistro** | | |
| **Descripción** | Esta operación agenda una cita de un paciente que no esté registrado en el sistema, una vez guardada envía una notificación por mensajería | | |
| **Roles** | Médico, auxiliar | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | datosContacto | DatosContacto | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **ObtenerCitasPaciente** | | |
| **Descripción** | Obtiene las citas del paciente indicado | | |
| **Roles** | Paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idPaciente | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Cita> | | |
| | | | |
| **Operación** | **ObtenerCitasMedico** | | |
| **Descripción** | Obtiene las citas que un médico tiene agendadas a su cargo | | |
| **Roles** | Médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idMedico | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Cita> | | |

| Operación | ObtenerCitasConsultorio | | |
|---|---|---|---|
| **Descripción** | Obtiene las citas registradas en un consultorio | | |
| **Roles** | Auxiliar | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsultorio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Cita> | | |

| Operación | EliminarCita | | |
|---|---|---|---|
| **Descripción** | Elimina una cita | | |
| **Roles** | Médico, auxiliar, paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idCita | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

| Operación | ObtenerCitasFuturas | | |
|---|---|---|---|
| **Descripción** | Esta operación obtiene las citas agendadas en un intervalo de fechas, como primer parámetro se indica la fecha inicial, el segundo parámetro días son los días máximos a partir de la fecha indicada | | |
| **Roles** | Proceso batch | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Fecha | Date | No |
| | Días | Int | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Cita> | | |

**Tabla 7-6 SERVICIOS DE ADMINISTRACIÓN DE CITAS.**

| SERVICIO DE ADMINISTRACIÓN DE CONSULTORIOS | | | |
|---|---|---|---|
| **Nombre del servicio** | Administración de consultorios | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.negocio.AdminConsultoriosNegocio | | |
| | | | |
| **Operación** | **ObtenerConsultorios** | | |
| **Descripción** | Obtiene todos los consultorios activos | | |
| **Roles** | Root | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | N/A | N/A | N/A |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Consultorio> | | |
| | | | |
| **Operación** | **ObtenerConsultorio** | | |
| **Descripción** | Obtiene la información de un consultorio en específico | | |
| **Roles** | Root | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | IdConsultorio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | Consultorio | | |
| | | | |
| **Operación** | **EliminarConsultorio** | | |
| **Descripción** | Elimina el consultorio indicado | | |
| **Roles** | Root | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsultorio | Integer | No |

| Mensaje de salida | Tipo de dato | | |
|---|---|---|---|
| | N/A | | |
| | | | |
| **Operación** | **GuardarConsultorio** | | |
| **Descripción** | Crea un nuevo registro de un consultorio | | |
| **Roles** | Root | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Consultorio | Consultorio | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla 7-7 SERVICIO DE ADMINISTRACIÓN DE CONSULTORIOS.**

| SERVICIO DE ADMINISTRACIÓN DE EXPEDIENTE | | | |
|---|---|---|---|
| **Nombre del servicio** | AdminExpedienteNegocio | | |
| **Ubicación Interfaz** | mx.egc.scmv.negocio.AdminExpedienteNegocio | | |
| | | | |
| **Operación** | **GuardarExpediente** | | |
| **Descripción** | Guarda el expediente de una persona, ya sea médico, auxiliar o paciente | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Expediente | Expediente | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **ObtenerExpediente** | | |
| **Descripción** | Obtiene el expediente de un usuario (médico, auxiliar o paciente) | | |
| **Roles** | Root, médico | | |

| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
|---|---|---|---|
| | idUsuario | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | Expediente | | |
| | | | |
| **Operación** | **GuardarConsulta** | | |
| **Descripción** | Guarda una consulta realizada por el médico a un paciente | | |
| **Roles** | Médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Consulta | Consulta | No |
| | solicitudEstudio | SolicitudEstudio | Sí |
| | listTipoEstudio | List<CatTipoEstudio> | Sí |
| | Archivos | List<bytre[]> | Sí |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **ObtenerConsultas** | | |
| **Descripción** | Obtiene las consultas realizadas a un paciente | | |
| **Roles** | Médico, paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idPaciente | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Consulta> | | |
| | | | |
| **Operación** | **ObtenerPrescripcion** | | |
| **Descripción** | Obtiene los medicamentos e indicaciones asociados a una consulta de un paciente | | |

| **Roles** | Médico, paciente | | |
|---|---|---|---|
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsulta | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Prescripcion> | | |
| | | | |
| **Operación** | **EnviarConsultaCorreo** | | |
| **Descripción** | Envía por correo electrónico al paciente la consulta indicada | | |
| **Roles** | Médico, paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsulta | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **ObtenerSolicitudEstudio** | | |
| **Descripción** | Recupera la información de la solicitud de estudio, si es que existe, ligada a una consulta | | |
| **Roles** | Médico, paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsulta | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **ObtenerImagenPerfil** | | |
| **Descripción** | Obtiene la imagen de perfil del paciente | | |
| **Roles** | Médico, paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |

| | | | |
|---|---|---|---|
| | Usuario | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | Byte[] | | |
| | | | |
| **Operación** | **obtenerarchivosConsulta** | | |
| **Descripción** | Obtiene los archivos asociados a la consulta indicada | | |
| **Roles** | Médico, paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsulta | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<byte[]> | | |
| | | | |
| **Operación** | **obtenerIdConultaNoenviadas** | | |
| **Descripción** | Obtiene los IDs de las consultas que no fueron enviadas por correo electrónico | | |
| **Roles** | Proceso batch | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | N/A | N/A | N/A |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Integer> | | |

**Tabla 7-8 SERVICIO DE ADMINISTRACIÓN DE EXPEDIENTE.**

| **SERVICIO DE ADMINISTRACIÓN DE CATALOGOS** | |
|---|---|
| **Nombre del servicio** | Administración de catálogos |
| **Ubicación Interfaz** | mx.egc.scmv.app.negocio.AdminCatalogosNegocio |
| | |
| **Operación** | **cambiarEstatusCatEnfermedad** |

| **Descripción** | Cambia el estatus del registro indicado | | |
|---|---|---|---|
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | id | Integer | No |
| | estatus | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **guardarCatEnfermedad** | | |
| **Descripción** | Guarda una entidad del tipo catálogo | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | catEnfermedad | CatEnfermedad | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **obtenerCatEnfermedad** | | |
| **Descripción** | Obtiene el catálogo de enfermedades<br>Sí el parámetro de entrada es TRUE recupera únicamente registros activos, si ingresa FALSE recupera tanto activos como inactivos | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | activos | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatEnfermedad> | | |
| | | | |
| **Operación** | **obtenerCatEnfermedad** | | |

| **Descripción** | Obtiene la entidad del catálogo de enfermedades indicado | | |
|---|---|---|---|
| **Roles** | todos | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | id | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | CatEnfermedad | | |
| | | | |
| **Operación** | **cambiarEstatusCatMedicamento** | | |
| **Descripción** | Cambia el estatus del registro indicado | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | id | Integer | No |
| | estatus | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **guardarCatMedicamento** | | |
| **Descripción** | Guarda una entidad del tipo catálogo medicamento | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | catMedicamento | CatMedicamento | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **obtenerCatMedicamento** | | |
| **Descripción** | Obtiene el catálogo de medicamentos<br>Sí el parámetro de entrada es TRUE recupera únicamente registros | | |

| | | | |
|---|---|---|---|
| | activos, si ingresa FALSE recupera tanto activos como inactivos | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | activos | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatMedicamento> | | |
| | | | |
| **Operación** | **obtenerCatMedicamento** | | |
| **Descripción** | Obtiene la entidad del catálogo de medicamentos indicado | | |
| **Roles** | todos | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | id | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | CatMedicamento | | |
| | | | |
| **Operación** | **cambiarEstatusCatTipoEstudio** | | |
| **Descripción** | Cambia el estatus del registro indicado | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | id | Integer | No |
| | estatus | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **guardarCatTipoEstudio** | | |
| **Descripción** | Guarda una entidad del tipo catálogo tipo estudio | | |
| **Roles** | Root, médico | | |

| Mensaje de entrada | Nombre del atributo | Tipo de dato | Opcional |
|---|---|---|---|
| | catTipoEstudio | CatTipoEstudio | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **obtenerCatTipoEstudio** | | |
| **Descripción** | Obtiene el catálogo de tipo de estudio<br>Sí el parámetro de entrada es TRUE recupera únicamente registros activos, si ingresa FALSE recupera tanto activos como inactivos | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | activos | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatTipoEstudio> | | |
| | | | |
| **Operación** | **obtenerCatTipoEstudio** | | |
| **Descripción** | Obtiene la entidad del catálogo de tipo de estudios indicado | | |
| **Roles** | Todos | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | id | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | CatTipoEstudio | | |
| **Operación** | **obtenerTipoSangre** | | |
| **Descripción** | Recupera el registro del tipo de sangre indicado | | |
| **Roles** | Todos | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | tipoSangre | String | No |

| **Mensaje de salida** | **Tipo de dato** | | |
|---|---|---|---|
| | CatTipoSangre | | |
| | | | |
| **Operación** | **obtenerTiposSangre** | | |
| **Descripción** | Obtiene el catálogo de tipo de sangre<br>Sí el parámetro de entrada es TRUE recupera únicamente registros activos, si ingresa FALSE recupera tanto activos como inactivos | | |
| **Roles** | Root, médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | activos | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatTipoSangre> | | |

**Tabla 7-9 SERVICIO DE ADMINISTRACIÓN DE CATALOGOS**

**Paso 4.4.3: Aplicar principios de orientación a servicios**

Como se está ejemplificando solo una operación, el análisis restante se realizará sobre esta operación y no sobre el servicio de expediente completo.

La operación ***enviar prescripción por correo*** tiene un bajo grado de reusabilidad, hay tres momentos en los que esta puede ser invocada, a pesar de ello en los dos casos puede ejecutarse sin dificultad:

- Cuando el médico concluye la consulta
- Cuando un paciente requiere recuperar su prescripción
- Cuando el batch se ejecuta y busca las consultas que no se han enviado por correo del lado del sistema

Su funcionalidad interna está estructurada para tener el menor grado posible de dependencia de otros servicios, solo existe una dependencia del servicio de mensajería, explicado previamente.

**Paso 4.4.4: Estandarizar y refinar la interfaz del servicio.**

La interfaz del paso 2  ya fue creada con las reglas de nombrado que se especifican en el ANEXO D. CONVENCIONES PARA ELECCIÓN DE NOMBRES

**Paso 4.4.5: Identificar el procesamiento requerido.**

Las tareas que debe cumplir este servicio son realizadas correctamente, por lo que cumple el objetivo de la construcción del servicio.

# CAPÍTULO 8. DESARROLLO DEL SISTEMA.

En este capítulo se describe el desarrollo del sistema de consultorio médico virtual, anteriormente ya se ha explicado cómo es la estructura de las capas de la aplicación, en la que hay una separación clara del *negocio* del SCMV y que puede ser accedido desde una aplicación web, posiblemente una aplicación de dispositivo móvil, servicios web.

Por alcance y tiempo para el presente trabajo de tesis se ha desarrollado el acceso por web, el proyecto (WAR) que se desplegará en el servidor de aplicaciones contiene ambas funcionalidades, el negocio del SCMV y el acceso web, sin perder de vista el objetivo principal, que es diseñar una arquitectura orientada a servicios.

Los servicios de mensajería y documentos también han sido desarrollados como proyectos separados y exponen su funcionalidad a través de servicios web, SOAP.

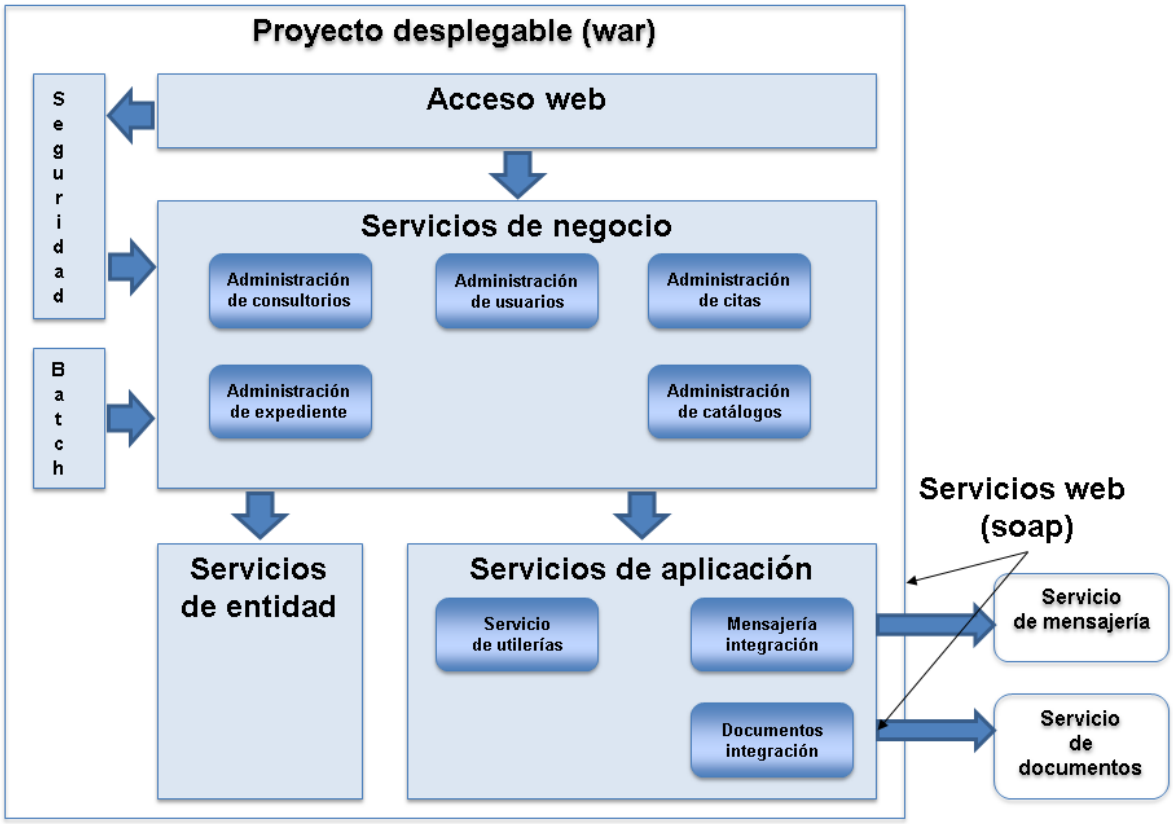


Ilustración 8-1 Diagrama de bloques del SCMV.

## 8.1 Tecnologías utilizadas.

A continuación se mencionan las tecnologías utilizadas para el desarrollo del SCMV, se da una breve descripción de las mismas, para qué sirven y en qué momento fueron utilizadas. Cabe destacar que todas ellas son software libre.

**Java**

Los tres proyectos, SCMV, SM y SD, fueron desarrollados bajo este lenguaje de programación. Las principales decisiones de utilizarlo son que no había curva de aprendizaje pues es un lenguaje conocido por el autor, existe una gran variedad de APIs y frameworks que facilitan el desarrollo, también es un lenguaje multiplataforma

**Apache Maven**

Es una herramienta para para la construcción de proyectos desarrollados con Java, *Maven* se encarga de gestionar el ciclo de vida de los proyectos, validación, generación de código, compilación, pruebas, empaquetamiento, pruebas de integración, verificación, instalación, despliegue y creación e implementación de sitios, también nos permite administrar las dependencias del proyecto. De igual forma, los tres proyectos están gestionados con *Maven*.

**Spring framework**

Comúnmente conocido solo como Spring, es un framework que ayuda a simplificar el desarrollo de aplicaciones, está conformado por diversos módulos, estos módulos aportan funcionalidad directa por el propio spring o facilitan la integración con otros frameworks o sistemas.

- Spring core
- Spring security
- Spring web
- Spring mvc
- Spring orm
- Spring aop
- Spring context

Entre las ventajas que aporta se encuentran:

1. No es intrusivo en el código, puede ser configurado a través de archivos XML o con anotaciones

2. Inyección de dependencias, el framework se encarga de instanciar y suministrar las dependencias que una clase necesita, se hace a través de una interfaz lo que permite que la implementación quede oculta

Para la seguridad del sistema, autenticación y autorización, se implementó spring security el cual permite integrar distintos sistemas de seguridad, pueden ser sistemas legados, customizados, externos. Para validación de usuario y contraseña en el SCMV se utiliza el *AuthenticationProider* que hace una consulta a la base de datos buscando el registro que coincide con dicho usuario y contraseña, si existe obtiene los roles asociados a dicho usuario.

**Hibernate**

Es un framework ORM (Mapeo Objeto Relacional), es decir, permite hacer la conversión del universo entidad relación al mundo de orientado a objetos, en este caso para Java.

- Abstrae la integración entre la aplicación y la base de datos
- Permite una independencia del motor de base de datos sin afectar en el código la forma de conexión, bastará con cambiar en el archivo de configuración el tipo de base de datos a la que se conecta
- Tiene licencia GNU y LGPL

**Java Server Faces**

JSF es una especificación para desarrollar interfaces de usuario basadas en web, JSF por debajo utiliza JSP lo que le permite hacer el despliegue de páginas. JSF se ejecuta del lado del servidor, al ser una especificación, existen distintas librerías que la implementan, por ejemplo: myFaces, rich faces, prime faces, entre otras.

JSF ofrece:

- Un conjunto de APIS para representar componentes de una interfaz de usuario y administrar su estado, manejar su estado, validar entradas, define un esquema de navegación de las páginas y da soporte para internacionalización y accesibilidad
- Un conjunto de componentes para la interfaz de usuario

**Prime faces**

Es una librería de componentes que implementa la especificación de JSF, además provee componentes y funcionalidades extra más avanzadas que las aportadas por default en la especificación de JSF.

Prime faces esta liberado bajo la licencia Apache 2.0.

**Apache CXF**

Es un marco de servicios de código abierto que ayuda a desarrollar servicios utilizando las APIs como JAX-WS y JAX-RS, estos servicios pueden comunicarse con una variedad de protocolos como SOAP, XML/HTTP, RESTful HTTP, o CORBA.

**Quartz**

Es una biblioteca de código abierto para aplicaciones Java, que permite la ejecución de tareas que deben ser iniciadas en momentos específicos del tiempo. Las tareas pueden ser ejecutadas de acuerdo a las siguientes condiciones:

- En un momento determinado del día (milisegundos)
- Ciertos días de la semana
- Ciertos días del mes
- Ciertos días del año
- No en ciertos días
- Repetido N número de veces
- Repetido hasta cierta hora/fecha especificada

Repetido indefinidamente

Repetido con un intervalo de retardo

**MySQL**

Las principales razones por la que se escogió mySQL como sistema de gestión de base de datos (SGBD) es que es ligero y potente, no requiere muchos recursos de hardware para poder trabajar y es adecuado para el presente proyecto de tesis.

**Twitter4J**

Es un API para poder integrar una aplicación java con el servicio de twitter, no es API oficial de Twitter. Está liberada bajo la licencia Apache 2.0

## 8.2 Implementación

A continuación se muestra cómo está constituido finalmente el sistema.

* En el alcance definido para este proyecto se excluyó la interfaz para la administración de los catálogos.

| Vistas en JSF | | |
|---|---|---|
| **Paquete/folder** | **Pantalla/Descripción.** | |
| webapp /consultorio | altaConsultorio.xhtml | Esta vista sirve para crear, modificar y ver el detalle de un consultorio |
| | consultorios.xhtml | Enlista todos los consultorios activos en el sistema |
| webapp/médicos | altaMedico.xhtml | Sirve para registrar y modificar la información de un médico |
| | detalleMedico.xhtml | Muestra la información de un médico |
| | medicos.xhtml | Lista los médicos registrados en el sistema |
| webapp/pacientes | agendarCita.xhtml | Aquí el médico o el auxiliar pueden agendar citas para pacientes que aún no están registrados en el sistema |
| | altaPaciente.xhtml | Sirve para registrar un nuevo paciente y también sirve para modificar su información |
| | citaPaciente.xhtml | Aquí un paciente que ya está registrado puede realizar una agendar una cita con su médico. |
| | citasMedico.xhtml | En lista las citas que el médico autenticado tiene asignadas |
| | pacientes.xhtml | Si el usuario autenticado es el médico, muestra los pacientes que tiene asignados a su cargo, si el usuario autenticado es un auxiliar |

| | | |
|---|---|---|
| | | muestra los pacientes pertenecientes al consultorio en el que está registrado |
| | consulta.xhtml | En esta vista el médico registra la información de una consulta para un paciente |
| webapp/auxiliar | altaAuxiliar.xhtml | Sirve para crear y modificar a los usuarios auxiliares |
| | auxiliares.xhtml | Enlista los auxiliares registrados pertenecientes al consultorio del médico autenticado |
| | detalleAuxiliar.xhtml | Muestra la información completa de los auxiliares |
| webapp/configuración | cambiarContrasenia.xhtml | Vista para hacer un cambio de contraseña |
| webapp/consultorio | altaConsultorio.xhtml | Permite registrar, modificar y ver la información de un consultorio |
| | consultorios.xhtml | Lista los consultorios registrados en el sistema |
| webapp/expediente | citasPaciente.xhtml | Lista todas las citas agendadas que tiene el paciente en cuestión. |
| | expediente.xhtml | Muestra la información general de un paciente y el listado de las consultas registradas |
| | detalleConsulta.xhtml | Muestra toda la información guardada en una durante una consulta previamente indicada |

**Tabla 8-1 Vistas en JSF.**

| Paquetes de clases de control de las vistas | |
|---|---|
| **Paquete** | **Descripción** |
| mx.egc.scmv.app.web | En este paquete se contienen las clases java que fungen como controladores de las JSF, se encargan de mandar a llamar los servicios de negocio requeridos en dicha pantalla |
| mx.egc.scmv.app.web.util | Contiene clases auxiliares de JSF para hacer conversión de datos, converters, entre la el frontend (JSF/html) con el backend (java)<br>Contiene una clase de apoyo para las vistas que tienen un alcance de vista, view scope |
| mx.egc.scmv.app.web.seguridad | Contiene una clase para administrar los atributos de sesión y una clase llamada autorizationProvider.java que se encarga de autenticar un usuario y proveer sus roles para el acceso web |

**Tabla 8-2 Paquete de clases de control de las vistas.**

| Servicios de tarea o de negocio | |
|---|---|
| **Paquete** | **Descripción** |
| mx.egc.scmv.app.negocio | Contiene las interfaces de los servicios de negocio<br>• AdmincitasNegocio.java<br>• AdminConsultoriosNegocio.java<br>• AdminExpedienteNegocio.java<br>• AdminUsuariosNegocio.java |
| mx.egc.scmv.app.negocio.impl | Las implementaciones de las interfaces de los servicios de negocio<br>• AdmincitasNegocioImpl.java<br>• AdminConsultoriosNegocioImpl.java<br>• AdminExpedienteNegocioImpl.java |

| | |
|---|---|
| | • AdminUsuariosNegocioImpl.java |
| mx.egc.scmv.app.negocio.catalogo | Contiene las interfaces de los servicios de administración de catálogos |
| mx.egc.scmv.app.negocio.catalogo.impl | Contiene las implementaciones de la administración de catálogos |
| mx.egc.scmv.utilerias | Contiene la interfaz del servicio de utilerías |
| mx.egc.scmv.utilerias.impl | Contiene la implementación del servicio de utilerías |

**Tabla 8-3 Servicios de tarea o de negocio.**

| Servicios de aplicación | |
|---|---|
| **Paquete** | **Descripción** |
| mx.egc.scmv.integracion | Contiene las interfaces de los servicios de aplicación |
| mx.egc.scmv.integracion.documentos | Contiene clases necesarias para consumir el servicio web de documentos |
| mx.egc.scmv.integracion.impl | Contiene las implementaciones de los servicios de aplicación |
| mx.egc.scmv.integracion.mensajeria | Contiene clases necesarias para consumir el servicio web de mensajería |

**Tabla 8-4 Servicios de aplicación.**

| Servicios de entidad | |
|---|---|
| **Paquete** | **Descripción** |
| mx.egc.scmv.app.modelo | Contiene las entidades de negocio mapeadas en la base de datos |
| mx.egc.scmv.app.dao | Contiene las interfaces de los servicios de entidad |
| mx.egc.scmv.app.dao.impl | Contiene las implementaciones de los servicios de entidad |

| | |
|---|---|
| mx.egc.scmv.app.dao.util | Contiene la interfaz InterfaceAuxDAO y la clase BaseUtilDAO con operaciones genéricas que son aplicables a todas las entidades con el fin de tener un solo código que lleve a cabo esas operaciones, la interfaz del servicio de entidad debe extender de esta interfaz y la implementación del servicio debe extender de la clase, tales operaciones son:<br>• obtenerPorId(Integer id)<br>• guardarActualizar(Objeto t)<br>• guardarActualizar(List<Object> listObject)<br>• obtenerActivos()<br>• obtenerActivos(String propiedad, boolean ascendente)<br>• eliminar(Integer id) |

**Tabla 8-5 Servicios de entidad.**

| Proceso Batch | |
|---|---|
| **Paquete** | **Descripción** |
| mx.egc.scmv.batch | Este paquete contiene las clases necesarias para poder arrancar los procesos batch, recordatorios de citas y envío de receta médica de forma electrónica |

**Tabla 8-6 Proceso Batch**

Diagramas de clase de servicios de negocio.

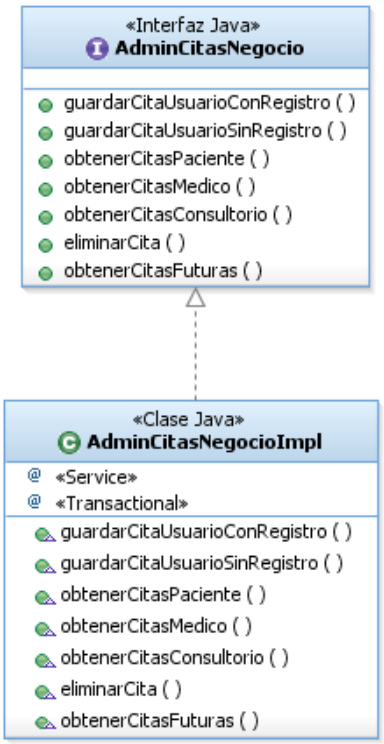


**Ilustración 8-2 Servicio de negocio, administración de citas.**

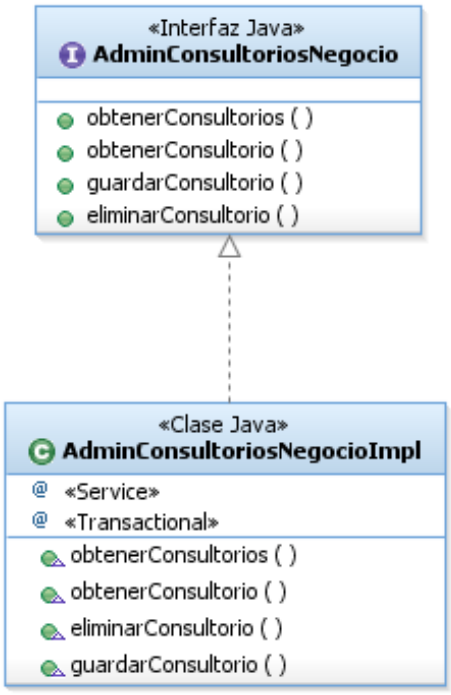


**Ilustración 8-3 Servicio de negocio, administración de consultorios.**


**Ilustración 8-4 Servicio de negocio, administración de usuarios y expediéntenle.**

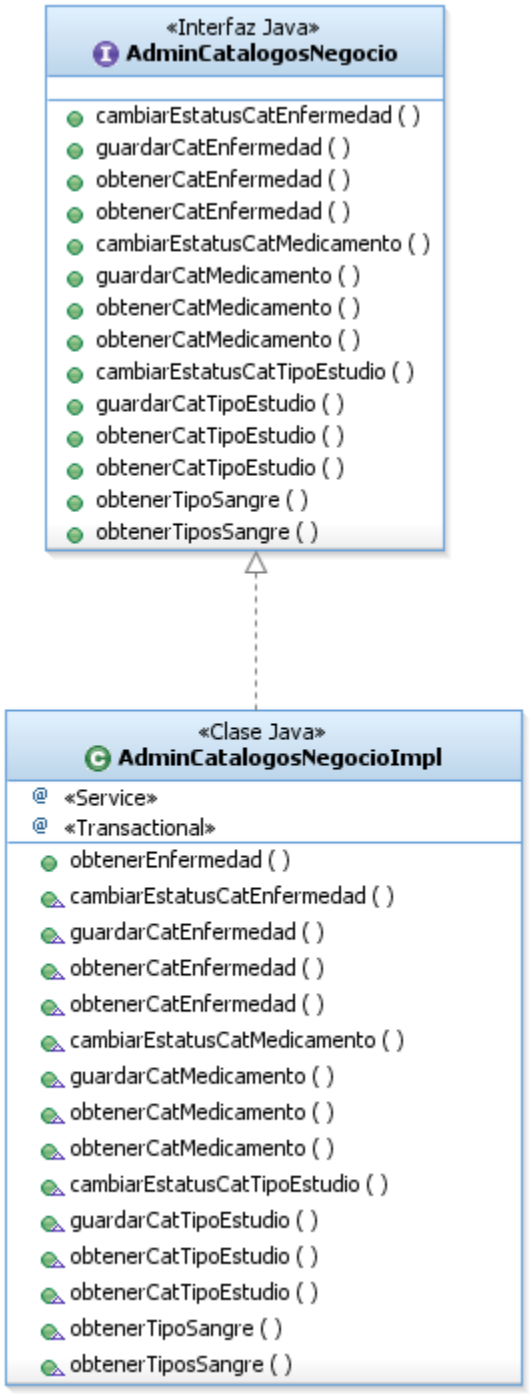


**Ilustración 8-5 Servicio de negocio, administración de catálogos.**

Diagramas de clases de los servicios de tarea.

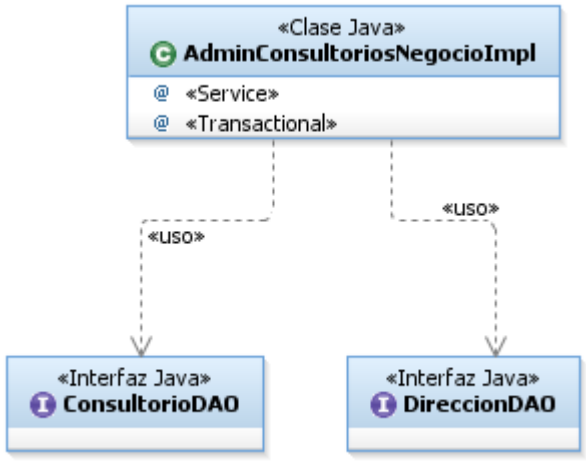


Ilustración 8-6 Composición de servicio, administración de consultorios.

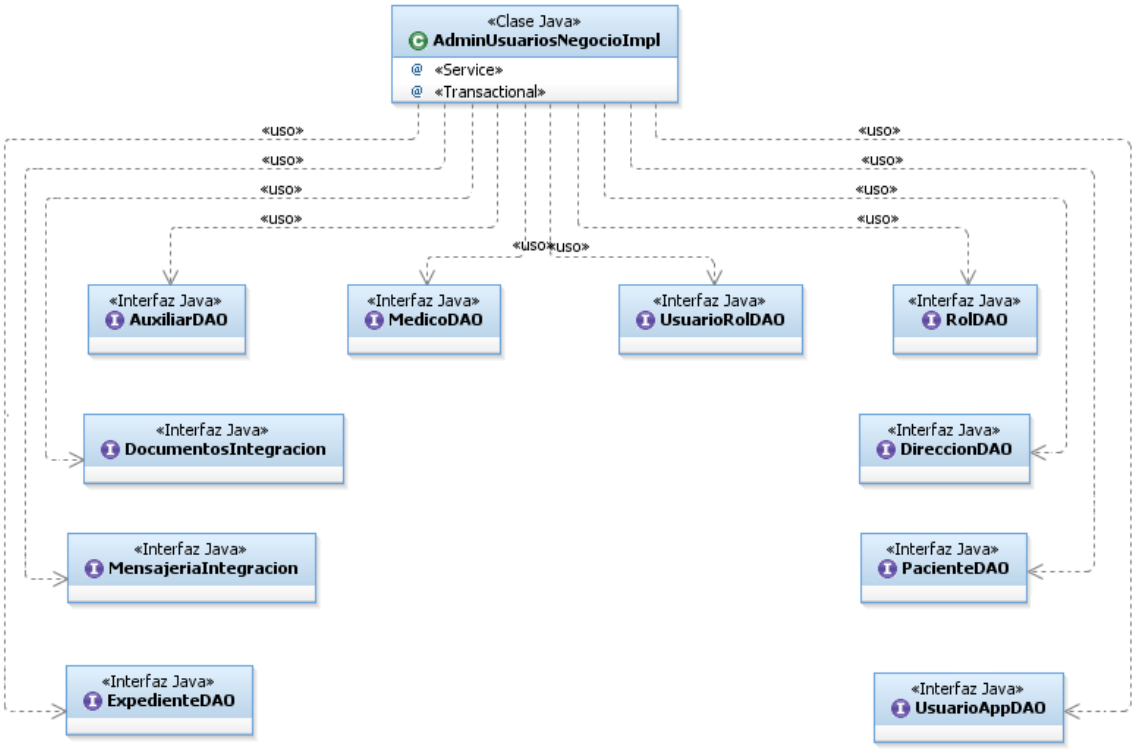


Ilustración 8-7 Composición de servicio, administración de usuarios.

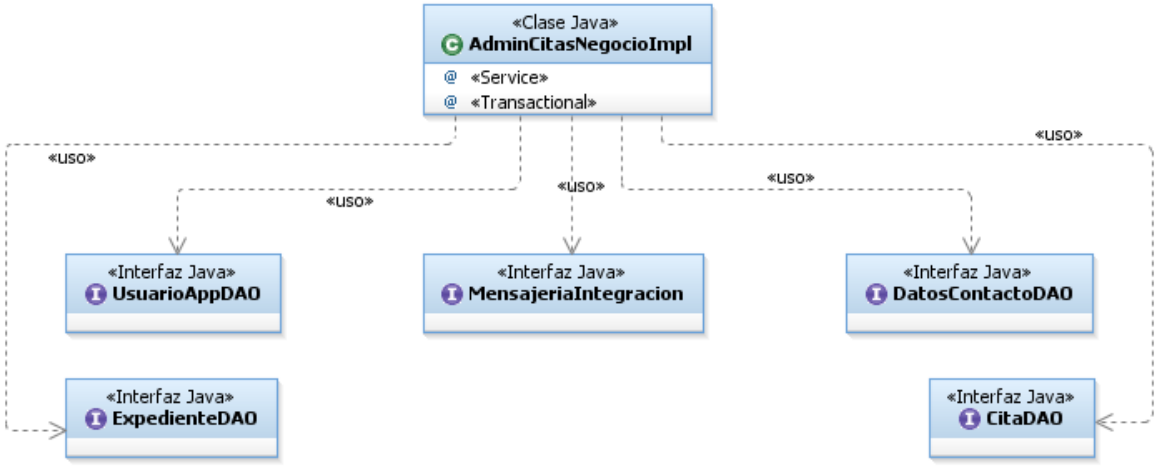


**Ilustración 8-8 Composición de servicio, administración de citas.**

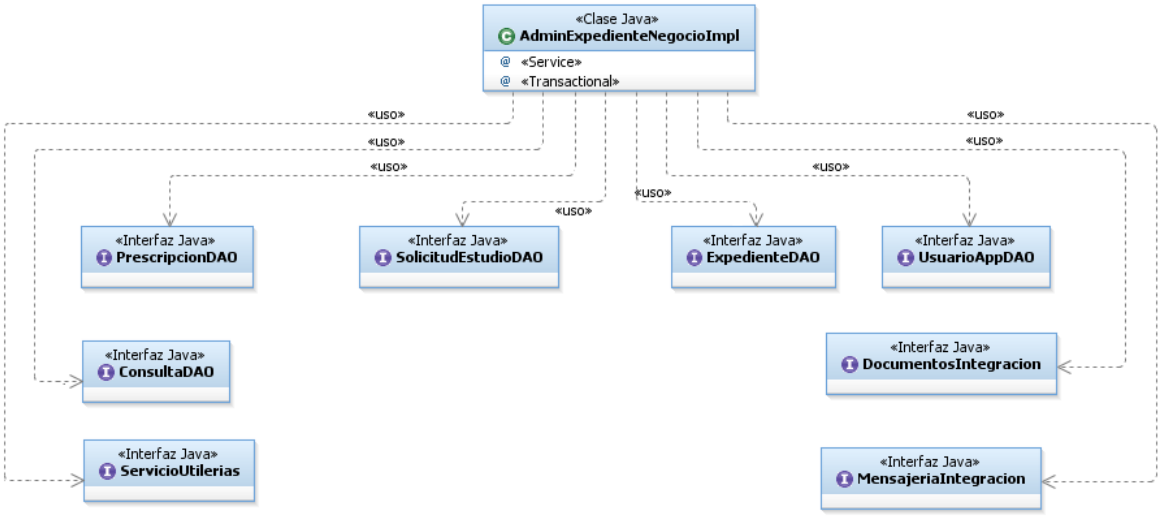


**Ilustración 8-9 Composición de servicio, administración de expediente.**

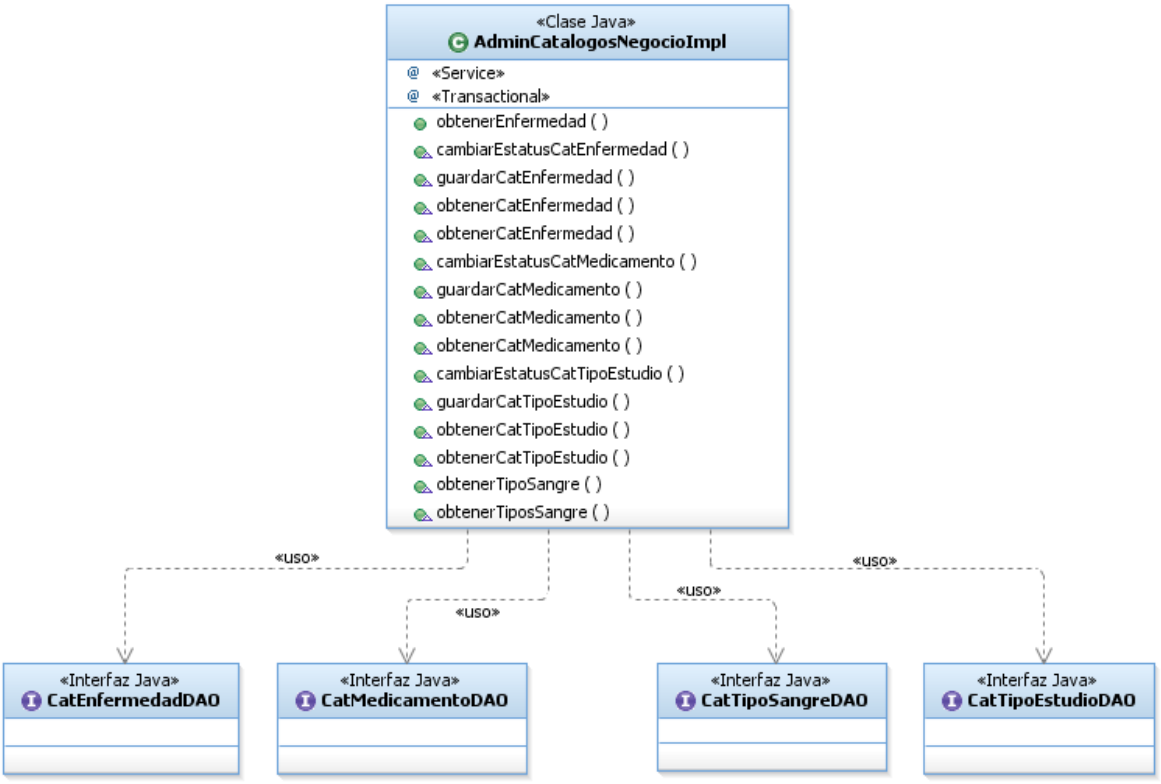


**Ilustración 8-10 Composición de servicio, administración de catálogos.**

## 8.3 Sistema Consultorio Médico Virtual, SCMV.

A continuación se muestra la apariencia del cliente web desarrollado.

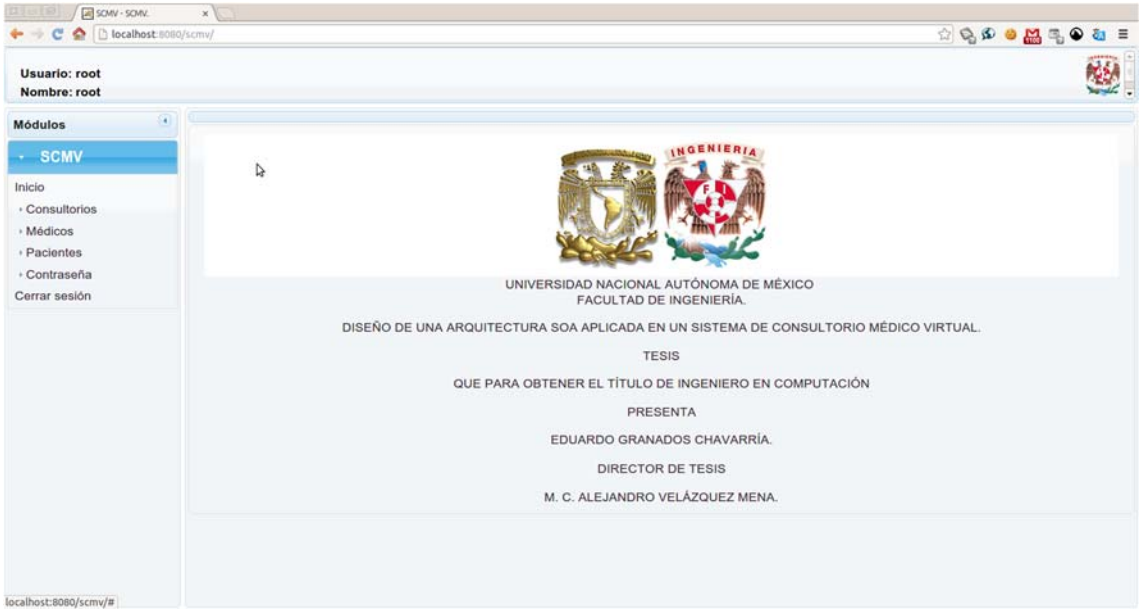


**Ilustración 8-11 Pantalla de presentación.**

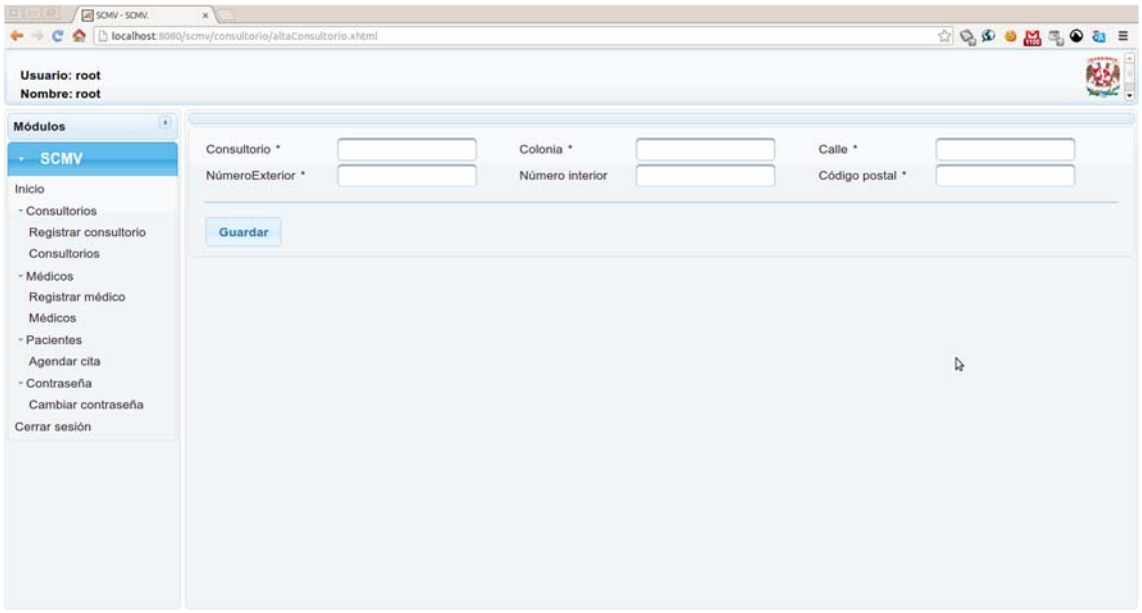


**Ilustración 8-12 Registrar consultorio.**

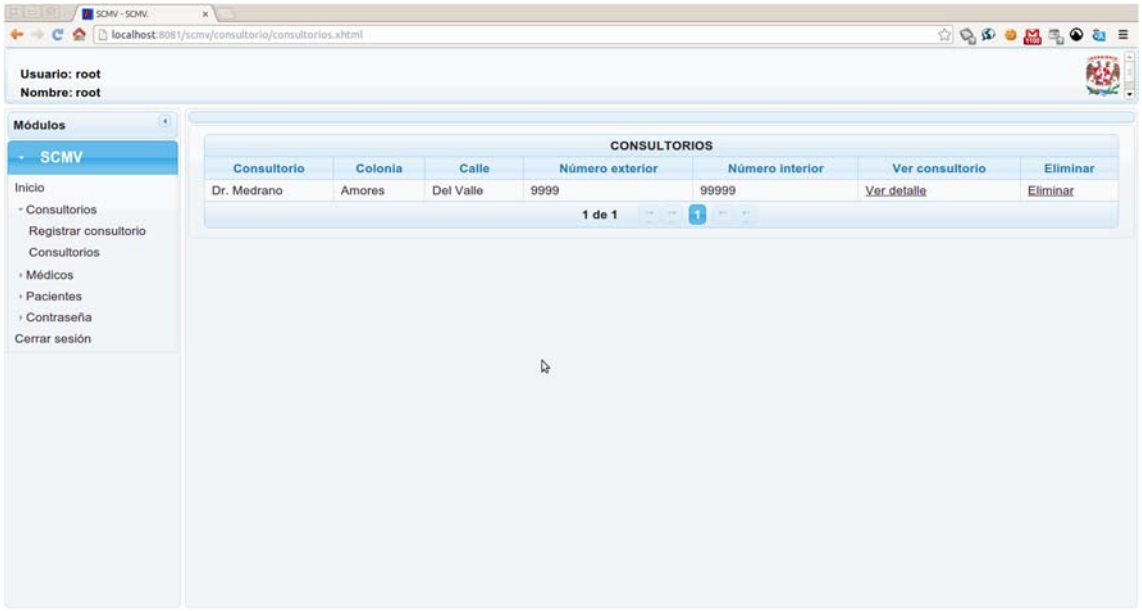


**Ilustración 8-13 Listado de consultorios.**

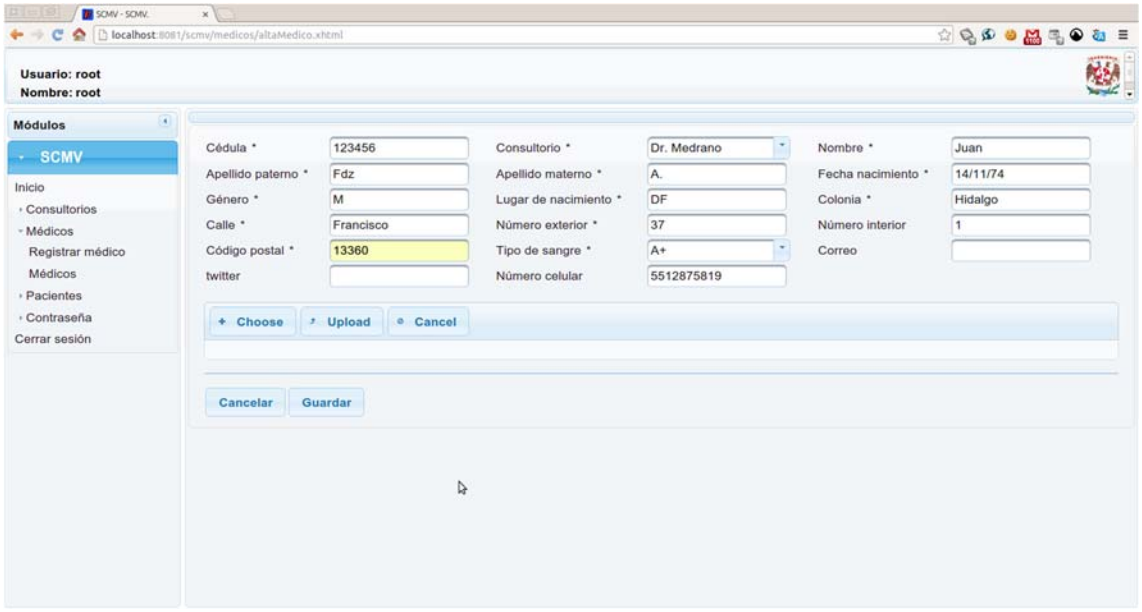


**Ilustración 8-14 Registrar médico.**

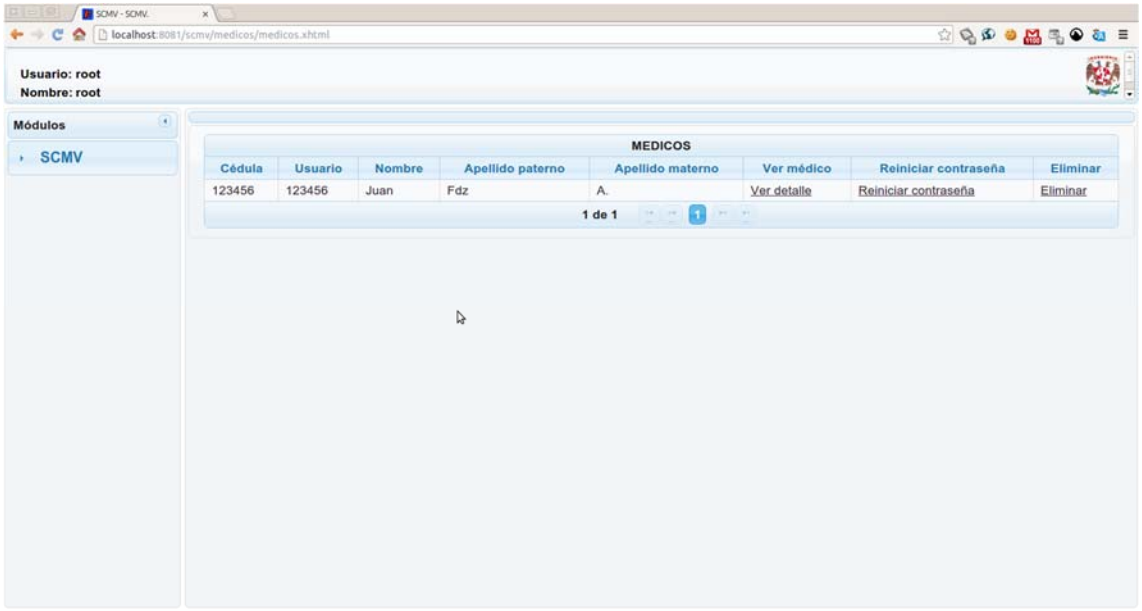


**Ilustración 8-15 Listado de médicos.**

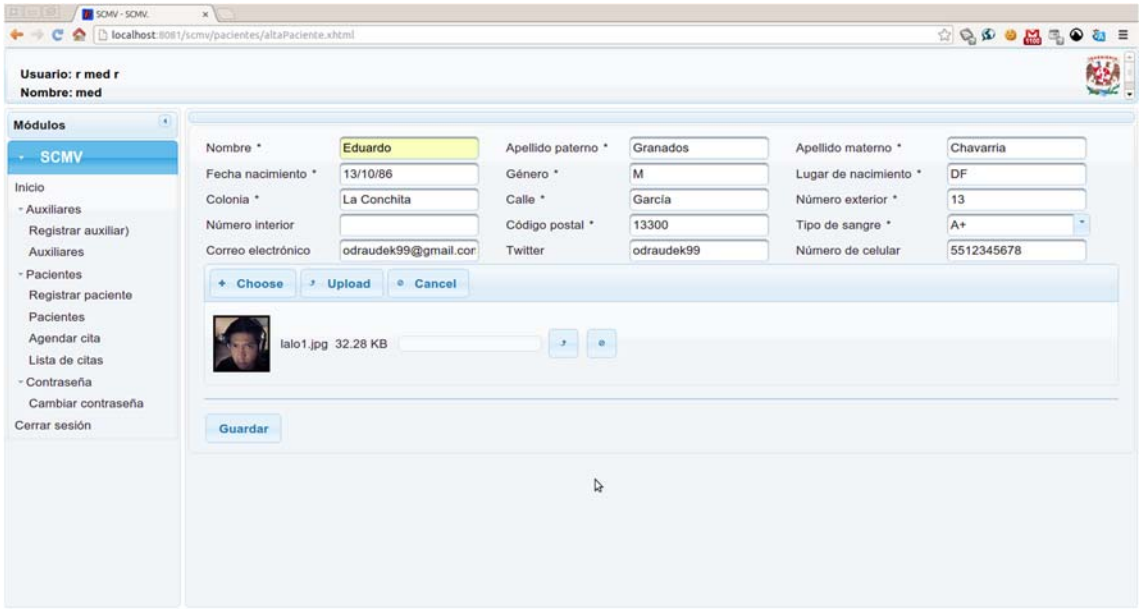


**Ilustración 8-16 Registrar paciente.**

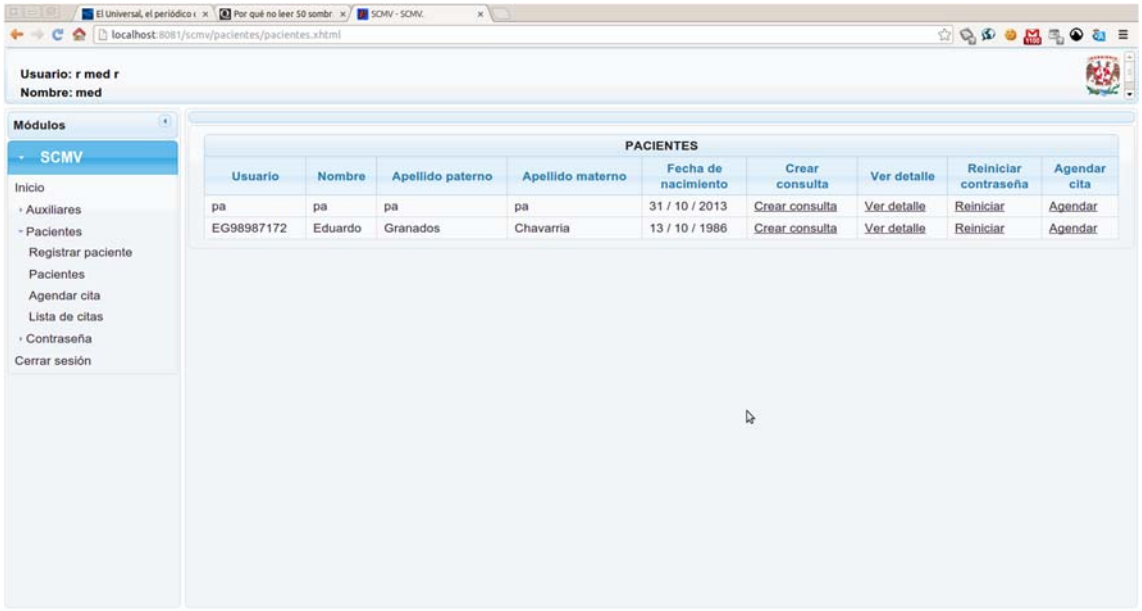


**Ilustración 8-17 Listado de pacientes.**

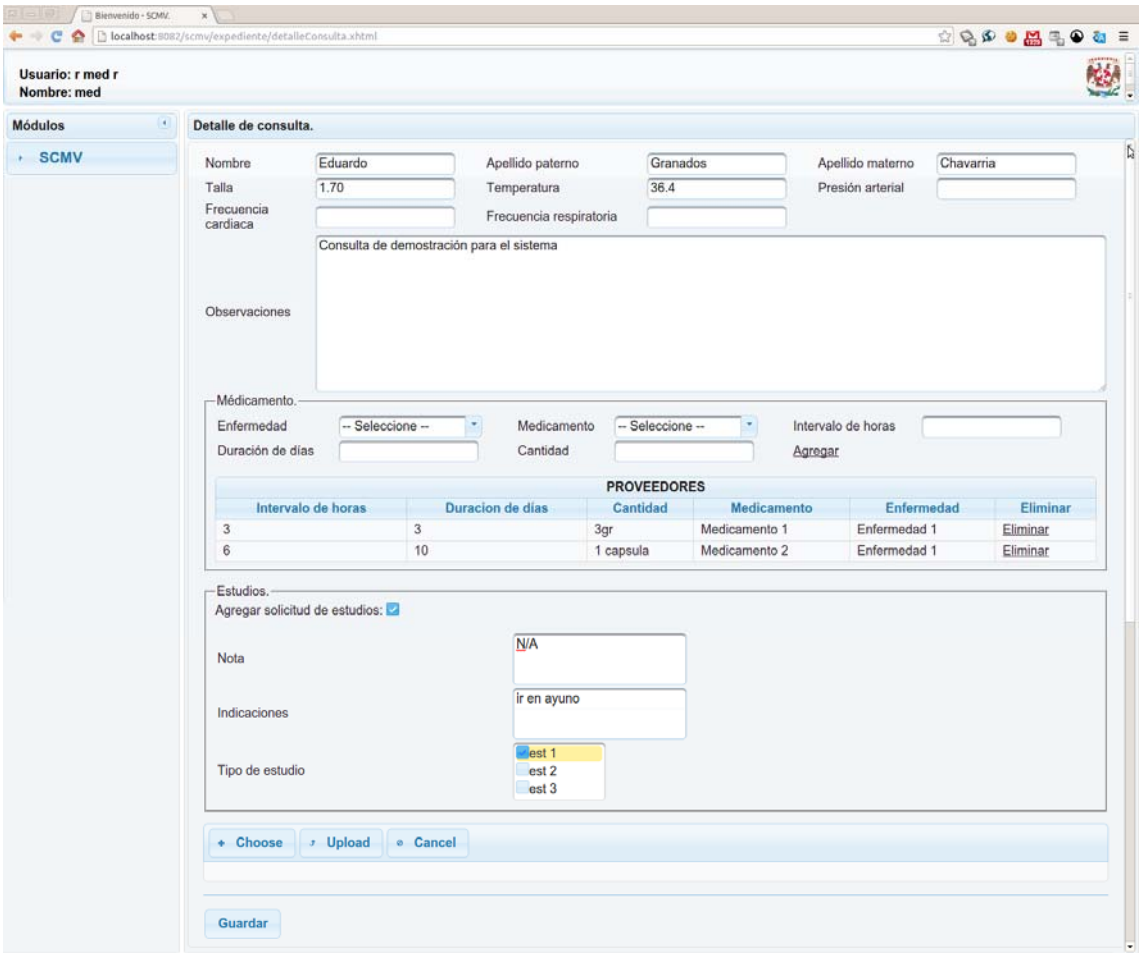


**Ilustración 8-18 Crear consulta.**

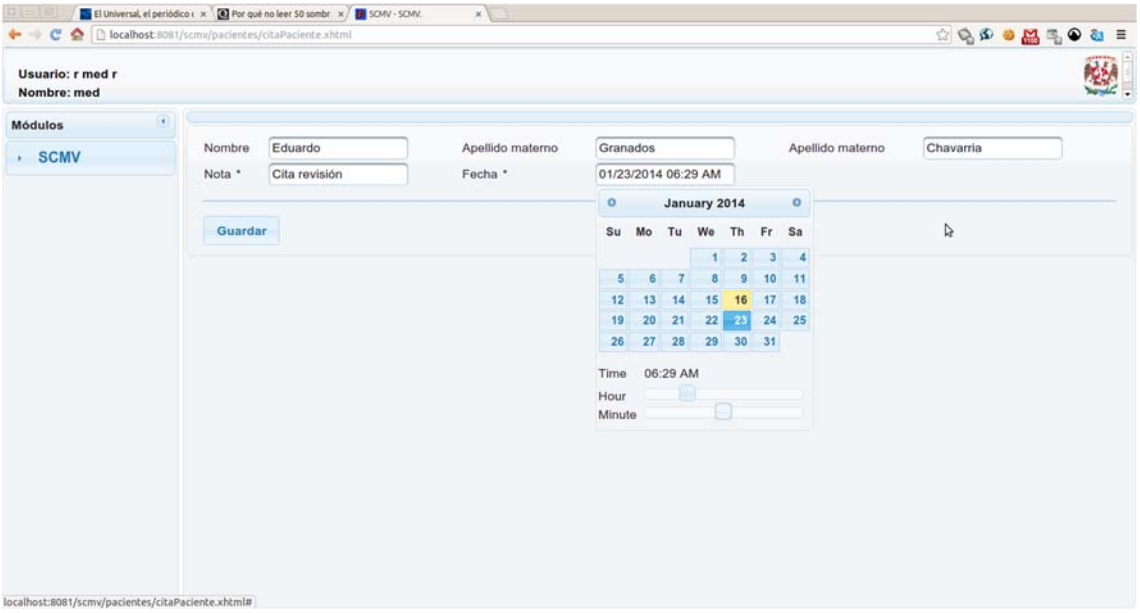


**Ilustración 8-19 Registrar cita de paciente registrado.**

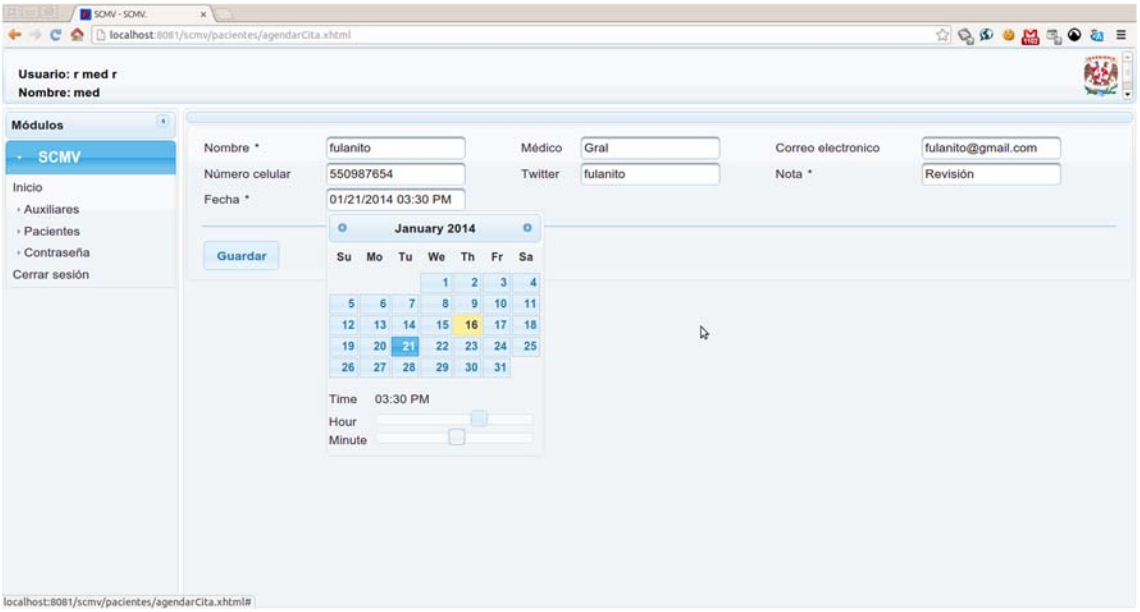


**Ilustración 8-20 Registrar cita de paciente no registrado.**

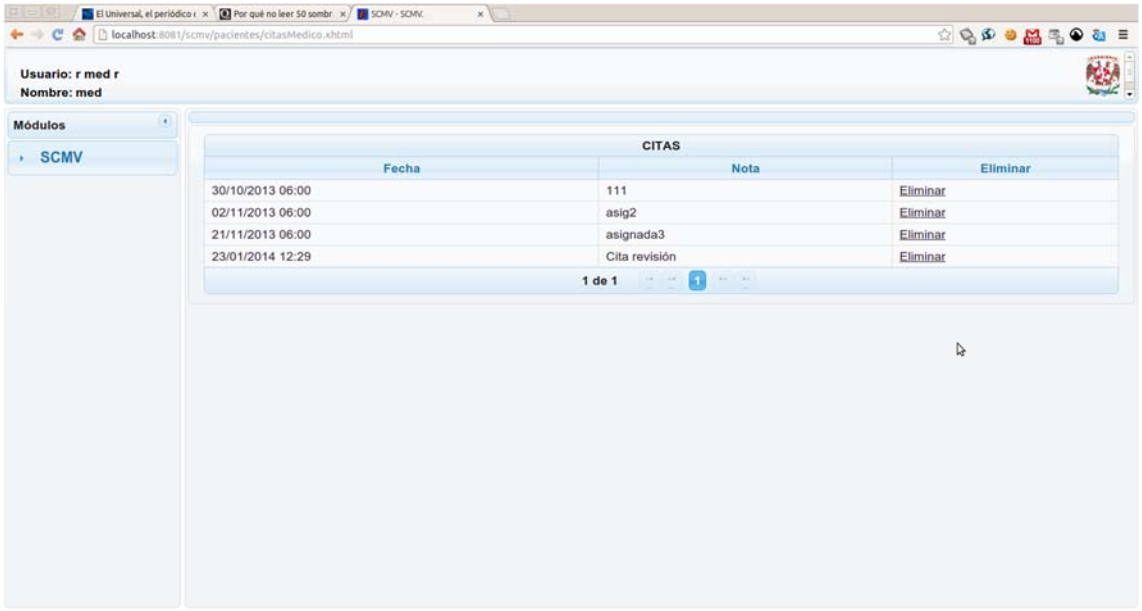


**Ilustración 8-21 Listado de consultas del paciente.**

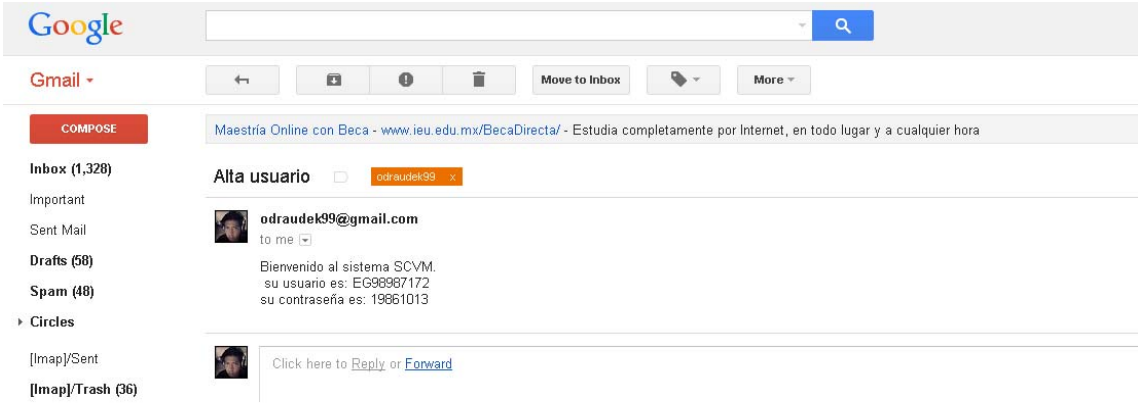


**Ilustración 8-22 Correo electrónico de notificación de registro.**

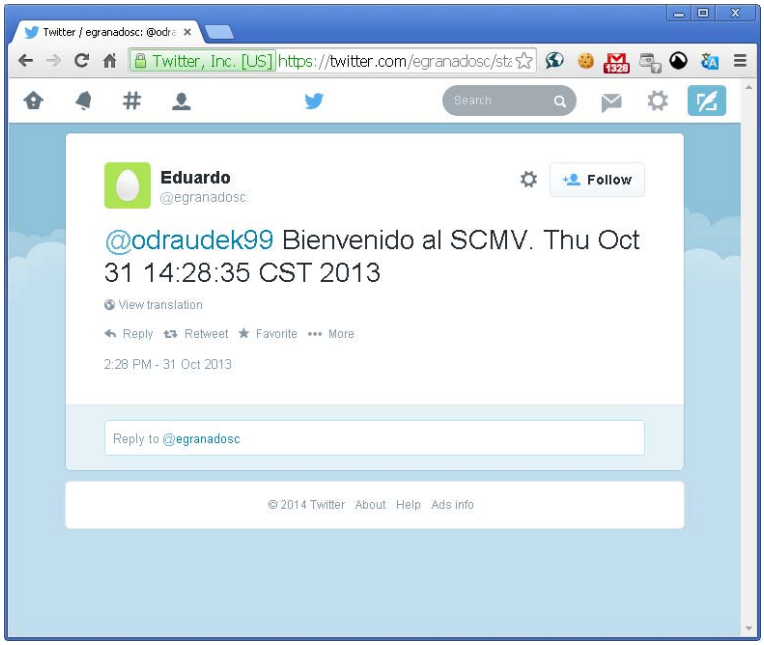


**Ilustración 8-23 Mensaje de Twitter de notificación de registro.**


**Ilustración 8-24 Notificación de Twitter de cita agendada.**


**Ilustración 8-25 Recordatorio de cita vía Twitter.**

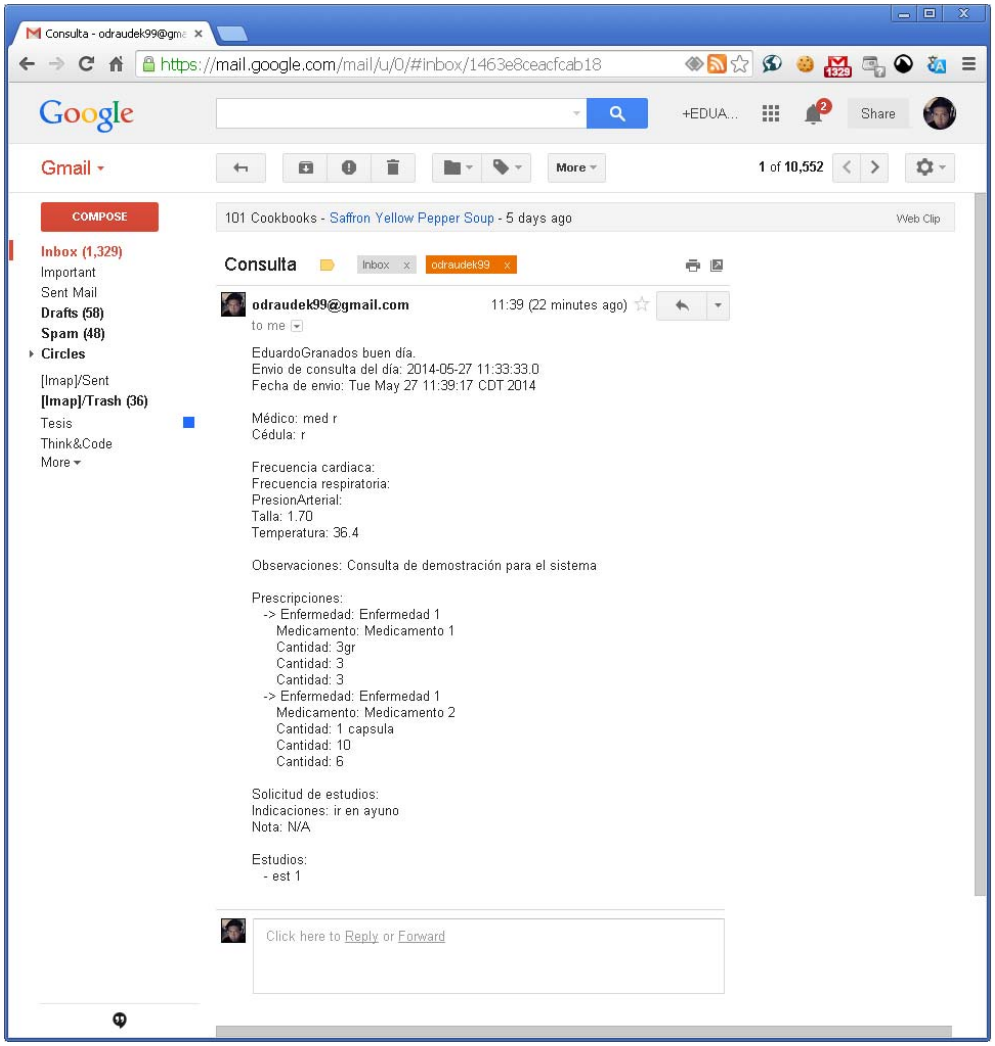


**Ilustración 8-26 Envió de consulta médica por medio de correo electrónico.**

# RESULTADOS Y CONCLUSIONES

Para lograr convertir las necesidades del negocio en una solución tecnológica es necesario que el arquitecto de software cuente con el conocimiento del negocio en cuestión, apoyado por los expertos del mismo, para poder representarlo en bloques bien definidos que resuelvan correctamente los requerimientos establecidos, posteriormente, tras el análisis y diseño éstos se transforman en módulos y/o servicios que siguen las mejores prácticas de desarrollo y programación.

Es importante resaltar que el desarrollo de software no debe iniciarse enseguida a “tirar código” o “trabajar a destajo”. Para lograr sacar adelante un proyecto como se mencionó en el párrafo anterior, es recomendable tener conocimiento general del negocio, subsecuentemente cada miembro del equipo debe conocer las actividades que tiene a su cargo, es decir, suministrar correctamente lo que otros necesitan de sus actividades y hacer llegar sus requerimientos a sus dependencias.

Hay que tener presente que al desarrollar una arquitectura SOA, esto no implica de manera tácita el utilizar de forma obligatoria ciertas herramientas o tecnologías, como desafortunadamente suele interpretarse. Es esencial, sin embargo, tener claro el concepto de servicio para que éstos puedan ser analizados y construidos conforme a los principios de diseño y siguiendo una metodología, en este caso se siguió la MSOAM que resultó adecuada para las necesidades del proyecto.

En el presente trabajo se definió e ilustró qué es un servicio, así como sus características, una entidad que encapsula una funcionalidad, con este desarrollo se espera quitar una idea clásica de que para hacer SOA hay que implementar servicios web.

Conforme a la metodología seguida en este trabajo se logró con éxito el desarrollo del Sistema de Consultorio Médico Virtual, el cual está conformado por servicios apegados a los principios de diseño. El SCMV está compuesto de:

- Servicios de entidad: 17 servicios desarrollados, especificados en el Anexo C
- Servicios de aplicación: 3
    - Servicio de mensajería
    - Servicio de documentos
    - Servicio de utilerías
- Servicios de negocio: 5 servicios desarrollados
    - Administración de usuarios
    - Administración de expedientes
    - Administración de citas

  - Administración de consultorios
  - Administración de catálogos
- Un procesamiento batch

Los desarrollos de los servicios desarrollados encargados de la mensajería y administración de documentos a los cuales el SCMV se integra, tienen la posibilidad de poder ser altamente reutilizados para algún otro sistema, que requiera proveer información puntual y/o personalizada a los usuarios del mismo.

La arquitectura del SCMV, separa claramente la capa de negocio de la capa de la vista o interfaz web, facilitando con ello el que pudiera ser utilizado por algún otro medio de acceso o cliente, tal como una aplicación móvil. Para ello, habría que desarrollar las interfaces que darán entrada a los servicios de negocio. Por otro lado, dicha separación también hace posible poder incrementar funcionalidad sin el riesgo de afectar negativamente lo que ya se tiene.

Derivado de lo anterior, se puede concluir y recomendar que para implementar una buena arquitectura SOA, primero se debe comprender el concepto de servicio y los principios del diseño de los mismos. Asimismo, se debe llevar a cabo el análisis y diseño de la solución con estricta orientación a servicios. Lo anterior, puede verse ejemplificado en el desarrollo del sistema SCMV, documentado en el presente trabajo.

# GLOSARIO

**Compose**: Traduciendo directamente al español es “componer”, en este trabajo se refiere a la acción de integrar distintos componentes para crear un ente más grande.

**CRUD**: Es el acrónimo en inglés para referirse a las cuatro operaciones que se pueden realizar sobre un conjunto de datos: Create, Read, Update, Delete, en español: crear, obtener, actualizar y eliminar.

**Estándar**: Un estándar es un conjunto de reglas que deben cumplir los productos, procedimientos o investigaciones que afirmen ser compatibles con el mismo producto. Los estándares ofrecen muchos beneficios, reduciendo las diferencias entre los productos y generando un ambiente de estabilidad, madurez y calidad en beneficio de consumidores e inversores.

**HPPT:** El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol), protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.

**Log**: es el registro o la bitácora de lo que va sucediendo dentro de la ejecución del software.

**Proceso batch**: es un software que está configurado para ejecutarse automáticamente en un momento futuro.

**Servicio web**: (*web service* en inglés), Son aplicaciones que se comunican sobre el protocolo HTTP, proporcionan un medio estándar de interoperabilidad entre aplicaciones de software que pueden estar ejecutándose en distintas plataformas y todas ellas logran entenderse ya que utilizan el XML para comunicarse, existe tanto cliente como servidor del servicio web.

**SOAP**: Es un protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido, está basado en XML que se compone de tres partes: una envoltura que define un marco para describir lo que contiene un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos por la aplicación, y una

convención para representar llamadas a procedimientos remotos y respuestas. SOAP se puede utilizar en combinación con una variedad de otros protocolos; por ejemplo en combinación con HTTP.

**Software**: Es el conjunto de programas, instrucciones para ejecutar ciertas tareas en una computadora, los cuales son acompañados de documentación y configuración de datos para que los programas funcionen correctamente.

**XML**: Son las siglas en inglés de eXtensible Markup Language, en español lenguaje de marcas extensible, es un lenguaje de etiquetas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Permite definir la gramática de lenguajes específicos. XML permite el intercambio de información estructurada entre diferentes plataformas.

# REFERENCIAS

[1] Kazman y Bass. (2003). *Software architecture in Practice 2nd Edition.* United States: Addison Wesley.

[2] Erl, Thomas . (2005). *Service-Oriented Architecture: Concepts, Technology, and Design.* United States: Prentice Hall.

[3] Erl, Thomas. (2005). *SOA Principles of service design.* United States: Prentice Hall.

[4] Oracle. (15 de noviembre de 2013). *JavaServer Faces Technology Overview*. Obtenido de http://www.oracle.com/technetwork/java/javaee/overview-140548.html

[5] Software Engineering Institute, CMU. (5 de octubre de 2013). *Software architecture*. Obtenido de Software Architecture: http://www.sei.cmu.edu/architecture/

[6] W3C. (5 de diciembre de 2013). *Guía Breve sobre Estándares Web*. Obtenido de http://www.w3c.es/Divulgacion/GuiasBreves/Estandares

# ANEXO A. SERVICIO DE MENSAJERÍA Y DE DOCUMENTOS

Estos servicios también fueron desarrollados como parte de este trabajo de tesis con el fin de ilustrar una integración con servicios externos.

## Servicio de documentos

El siguiente diagrama muestra cómo está conformado el servicio de documentos.

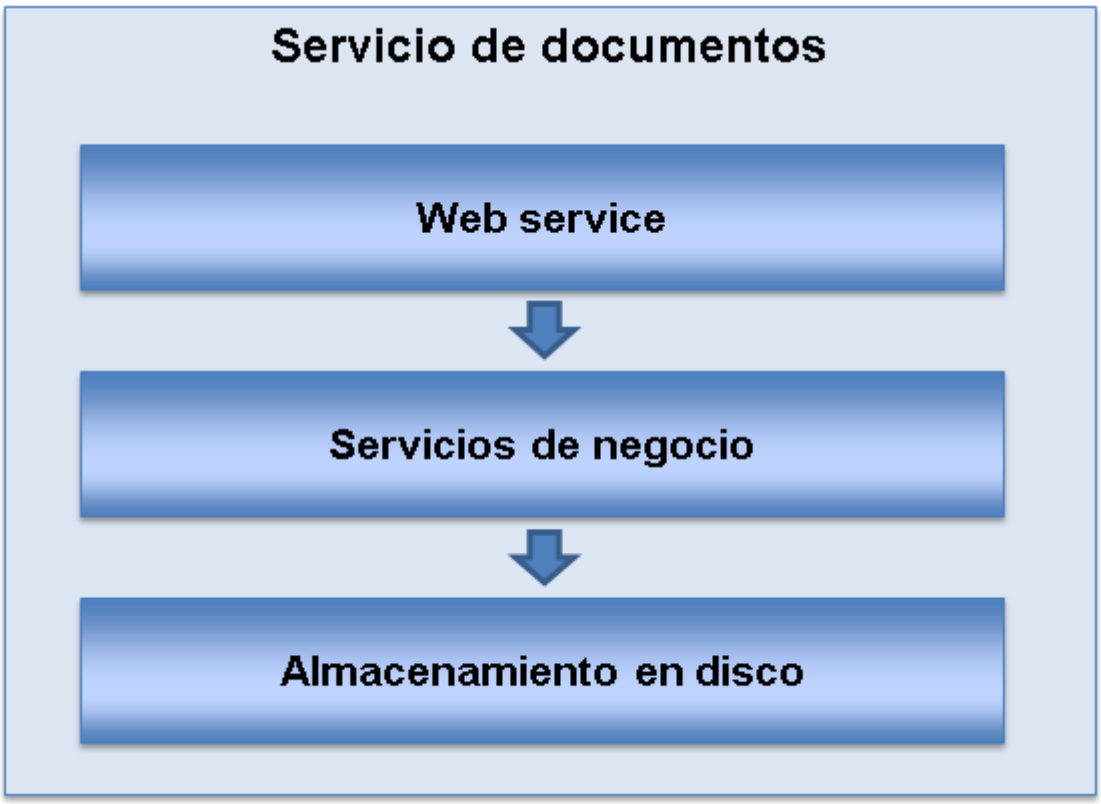


**Ilustración A-1 Diagrama de bloques de servicio de documentos.**

A continuación se muestra el WSDL del servicio de documentos.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="DocumentosService"
        targetNamespace="http://documentos.scmv.egc.mx/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
```

```
        xmlns:tns="http://documentos.scmv.egc.mx/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

        <!-- Importacion de esquemas -->
        <wsdl:types>
                <schema xmlns="http://www.w3.org/2001/XMLSchema">
                        <import namespace="http://documentos.scmv.egc.mx/"
                                schemaLocation="DocumentosServiceSchema.xsd" />
                </schema>
        </wsdl:types>

        <!-- Mensajes -->
        <wsdl:message name="enviarArchivoResponse">
                <wsdl:part name="parameters" element="tns:enviarArchivoResponse">
                </wsdl:part>
        </wsdl:message>
        <wsdl:message name="enviarArchivoRequest">
                <wsdl:part name="parameters" element="tns:enviarArchivoRequest">
                </wsdl:part>
        </wsdl:message>

        <wsdl:message name="recuperarArchivoResponse">
                <wsdl:part name="parameters" element="tns:recuperarArchivoResponse">
                </wsdl:part>
        </wsdl:message>
        <wsdl:message name="recuperarArchivoRequest">
                <wsdl:part name="parameters" element="tns:recuperarArchivoRequest">
                </wsdl:part>
        </wsdl:message>

        <!-- Port type -->

        <wsdl:portType name="DocumentosService">
                <wsdl:operation name="enviarArchivo">
                        <wsdl:input                                     name="enviarArchivoRequest"
message="tns:enviarArchivoRequest">
```

```
                </wsdl:input>
                <wsdl:output name="enviarArchivoResponse"
message="tns:enviarArchivoResponse">
                </wsdl:output>
            </wsdl:operation>

            <wsdl:operation name="recuperarArchivo">
                <wsdl:input name="recuperarArchivoRequest"
message="tns:recuperarArchivoRequest">
                </wsdl:input>
                <wsdl:output name="recuperarArchivoResponse"
message="tns:recuperarArchivoResponse">
                </wsdl:output>
            </wsdl:operation>

        </wsdl:portType>

        <wsdl:binding name="DocumentosServiceSoapBinding" type="tns:DocumentosService">

            <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http" />

            <wsdl:operation name="enviarArchivo">
                <soap:operation soapAction="" style="document" />
                <wsdl:input name="enviarArchivoRequest">
                    <soap:body use="literal" />
                </wsdl:input>
                <wsdl:output name="enviarArchivoResponse">
                    <soap:body use="literal" />
                </wsdl:output>
            </wsdl:operation>


            <wsdl:operation name="recuperarArchivo">
                <soap:operation soapAction="" style="document" />
                <wsdl:input name="recuperarArchivoRequest">
                    <soap:body use="literal" />
                </wsdl:input>
```

```
                        <wsdl:output name="recuperarArchivoResponse">
                                <soap:body use="literal" />
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>

        <wsdl:service name="DocumentosService">
                <wsdl:port                                              name="DocumentosServicePort"
binding="tns:DocumentosServiceSoapBinding">
                        <soap:address

        location="http://localhost:8080/scmv/services/DocumentosServicePort" />
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://documentos.scmv.egc.mx/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
        targetNamespace="http://documentos.scmv.egc.mx/" version="1.0">

        <xs:element name="enviarArchivoResponse" type="tns:enviarArchivoResponse" />
        <xs:element name="enviarArchivoRequest" type="tns:enviarArchivoRequest" />
        <xs:element name="recuperarArchivoResponse" type="tns:recuperarArchivoResponse" />
        <xs:element name="recuperarArchivoRequest" type="tns:recuperarArchivoRequest" />

        <xs:complexType name="enviarArchivoRequest">
                <xs:sequence>
                        <xs:element minOccurs="1" name="ruta" type="xs:string" />
                        <xs:element minOccurs="1" name="nombre" type="xs:string" />
                        <xs:element name="archivo" type="xs:base64Binary"
                                minOccurs="0" maxOccurs="1" />
                </xs:sequence>
        </xs:complexType>
```

```
        <xs:complexType name="enviarArchivoResponse">
                <xs:sequence>
                        <xs:element minOccurs="1" name="respuesta" type="xs:int" />
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="recuperarArchivoRequest">
                <xs:sequence>
                        <xs:element minOccurs="1" name="ruta" type="xs:string" />
                        <xs:element minOccurs="1" name="nombre" type="xs:string" />
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="recuperarArchivoResponse">
                <xs:sequence>
                        <xs:element name="archivo" minOccurs="0" type="xs:base64Binary"
                                maxOccurs="1" />
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

## Servicio de mensajería

El siguiente diagrama muestra cómo está conformado el servicio de mensajería:

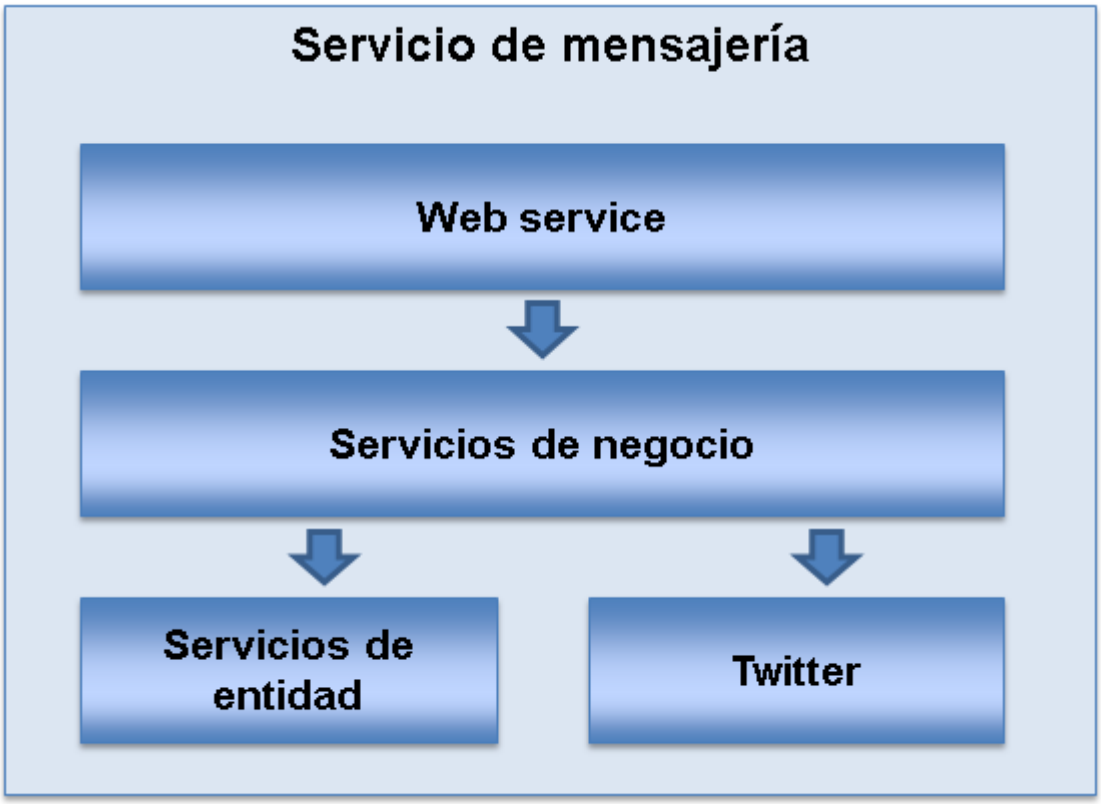


**Ilustración A-2 Diagrama de bloques de servicio de mensajería.**

A continuación se muestra el WSDL del servicio de mensajería:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="MensajeriaService"
        targetNamespace="http://mensajeria.scmv.egc.mx/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:tns="http://mensajeria.scmv.egc.mx/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

        <!-- Importacion de esquemas -->
        <wsdl:types>
                <schema xmlns="http://www.w3.org/2001/XMLSchema">
```

```
                <import namespace="http://mensajeria.scmv.egc.mx/"
                        schemaLocation="MensajeriaServiceSchema.xsd" />
        </schema>
</wsdl:types>

<!-- Mensajes -->
<wsdl:message name="enviarSmsResponse">
        <wsdl:part name="parameters" element="tns:enviarSmsResponse">
        </wsdl:part>
</wsdl:message>
<wsdl:message name="enviarSmsRequest">
        <wsdl:part name="parameters" element="tns:enviarSmsRequest">
        </wsdl:part>
</wsdl:message>

<wsdl:message name="enviarTwittResponse">
        <wsdl:part name="parameters" element="tns:enviarTwittResponse">
        </wsdl:part>
</wsdl:message>
<wsdl:message name="enviarTwittRequest">
        <wsdl:part name="parameters" element="tns:enviarTwittRequest">
        </wsdl:part>
</wsdl:message>

<wsdl:message name="enviarCorreoElectronicoResponse">
        <wsdl:part name="parameters" element="tns:enviarCorreoElectronicoResponse">
        </wsdl:part>
</wsdl:message>
<wsdl:message name="enviarCorreoElectronicoRequest">
        <wsdl:part name="parameters" element="tns:enviarCorreoElectronicoRequest">
        </wsdl:part>
</wsdl:message>

<!-- Port type -->
<wsdl:portType name="MensajeriaService">
        <wsdl:operation name="enviarSms">
                <wsdl:input                                    name="enviarSmsRequest"
```

```
message="tns:enviarSmsRequest">
                    </wsdl:input>
                    <wsdl:output name="enviarSmsResponse"
message="tns:enviarSmsResponse">
                    </wsdl:output>
                </wsdl:operation>

                <wsdl:operation name="enviarTwitt">
                    <wsdl:input name="enviarTwittRequest"
message="tns:enviarTwittRequest">
                    </wsdl:input>
                    <wsdl:output name="enviarTwittResponse"
message="tns:enviarTwittResponse">
                    </wsdl:output>
                </wsdl:operation>

                <wsdl:operation name="enviarCorreoElectronico">
                    <wsdl:input name="enviarCorreoElectronicoRequest"
message="tns:enviarCorreoElectronicoRequest">
                    </wsdl:input>
                    <wsdl:output name="enviarCorreoElectronicoResponse"
message="tns:enviarCorreoElectronicoResponse">
                    </wsdl:output>
                </wsdl:operation>
            </wsdl:portType>

            <wsdl:binding name="MensajeriaServiceSoapBinding" type="tns:MensajeriaService">

                <soap:binding  style="document"  transport="http://schemas.xmlsoap.org/soap/http"
/>

                <wsdl:operation name="enviarSms">
                    <soap:operation soapAction="" style="document" />
                    <wsdl:input name="enviarSmsRequest">
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output name="enviarSmsResponse">
                        <soap:body use="literal" />
                    </wsdl:output>
```

```
                </wsdl:operation>

                <wsdl:operation name="enviarTwitt">
                        <soap:operation soapAction="" style="document" />
                        <wsdl:input name="enviarTwittRequest">
                                <soap:body use="literal" />
                        </wsdl:input>
                        <wsdl:output name="enviarTwittResponse">
                                <soap:body use="literal" />
                        </wsdl:output>
                </wsdl:operation>

                <wsdl:operation name="enviarCorreoElectronico">
                        <soap:operation soapAction="" style="document" />
                        <wsdl:input name="enviarCorreoElectronicoRequest">
                                <soap:body use="literal" />
                        </wsdl:input>
                        <wsdl:output name="enviarCorreoElectronicoResponse">
                                <soap:body use="literal" />
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>

        <wsdl:service name="MensajeriaService">
                <wsdl:port                                      name="MensajeriaServicePort"
binding="tns:MensajeriaServiceSoapBinding">
                        <soap:address
location="http://localhost:8080/agatoWS/services/MensajeriaServicePort" />
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://mensajeria.scmv.egc.mx/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
```

```
        targetNamespace="http://mensajeria.scmv.egc.mx/" version="1.0">

        <xs:element name="enviarSmsResponse" type="tns:enviarSmsResponse" />
        <xs:element name="enviarSmsRequest" type="tns:enviarSmsRequest" />
        <xs:element name="enviarTwittRequest" type="tns:enviarTwittRequest" />
        <xs:element name="enviarTwittResponse" type="tns:enviarTwittResponse" />
        <xs:element name="enviarCorreoElectronicoRequest"
type="tns:enviarCorreoElectronicoRequest" />
        <xs:element name="enviarCorreoElectronicoResponse"
type="tns:enviarCorreoElectronicoResponse" />

        <xs:complexType name="enviarSmsRequest">
                <xs:sequence>
                        <xs:element minOccurs="1" name="usuario" type="xs:string" />
                        <xs:element minOccurs="1" name="mensaje" type="xs:string" />
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="enviarSmsResponse">
                <xs:sequence>
                        <xs:element minOccurs="1" name="folio" type="xs:long" />
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="enviarTwittRequest">
                <xs:sequence>
                        <xs:element minOccurs="1" name="usuario" type="xs:string" />
                        <xs:element minOccurs="1" name="twitterDestino" type="xs:string" />
                        <xs:element minOccurs="1" name="mensaje" type="xs:string" />
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="enviarTwittResponse">
                <xs:sequence>
                        <xs:element minOccurs="1" name="estatus" type="xs:string" />
                </xs:sequence>
        </xs:complexType>
```

```
        <xs:complexType name="enviarCorreoElectronicoRequest">
                <xs:sequence>
                        <xs:element minOccurs="1" name="usuario" type="xs:string" />
                        <xs:element minOccurs="1" name="destinatario" type="xs:string" />
                        <xs:element minOccurs="1" name="asunto" type="xs:string" />
                        <xs:element minOccurs="1" name="mensaje" type="xs:string" />
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="enviarCorreoElectronicoResponse">
                <xs:sequence>
                        <xs:element minOccurs="1" name="saldo" type="xs:long" />
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

### Servicio de mensajes SMS

El servicio de mensajes SMS se realiza a través de un proveedor que expone su servicio a través de un “http request” la cual recibe los parámetros por una invocación del tipo GET.

URL del servicio:

- http://servicio.smsmasivos.com.ar/enviar_SMS.asp?api=1

| Parámetros de entrada para el servicio contratado de SMS. | | |
|---|---|---|
| **Parámetro** | **Descripción** | **Obligatorio** |
| API | variable fija de valor “1” para indicar que las respuestas regresarán como texto plano | Sí |
| USUARIO | Nombre de usuario. | Sí |
| CLAVE | Contraseña del usuario | Sí |
| TOS | Número al que se desea enviar el SMS (con código de área) | Sí |
| TEXTO | Mensaje que se desea enviar (máximo 160 caracteres). | Sí |

| | | |
|---|---|---|
| | Caracteres permitidos:<br>ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!?#$%()*+,<br>-./:;=@ y el espacio | |
| TEST | Indica si se desea que se validen los datos del mensaje pero que no se envíe.<br>valor aceptado: “1” | No |

**Tabla A-1 Parámetros de entrada para el servicio contratado de SMS.**

Ejemplo de llamado.

- http://servicio.smsmasivos.com.ar/enviar_SMS.asp?api=1&usuario=DEMO500&clave=DEMO500&tos=1161883339&texto=Mensaje

Para mayor información consultar:

- www.smsmasivos.com.ar
- soporte@smsmasivos.com.ar

# ANEXO B. ENTIDADES

| Auxiliar | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idConsultorio | Consultorio | No |

**Tabla B-1 Entidad Auxiliar.**

| CatEnfermedad | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idCatEnfermedad | Integer | No |
| Enfermedad | String | No |
| Cuidados | String | No |
| Estatus | Boolean | No |
| catMedicamentoCollection | Collection<CatMedicamento> | No |

**Tabla B-2 Entidad Catálogo enfermedad.**

| CatMedicamento | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idCatMedicamento | Integer | No |
| nombreMedicamento | String | No |
| catEnfermedadCollection | Collecion<CatEnfermedad> | No |
| Estatus | Boolean | No |

**Tabla B-3 Entidad catálogo medicamento.**

| CatTipoEstudio | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idTipoEstudio | Integer | No |
| nombreEstudio | String | No |
| Estatus | Boolean | No |

**Tabla B-4 Entidad catálogo tipo de estudio.**

| CatTipoSangre | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idCatTipoSangre | String | No |
| Estatus | Boolean | No |

**Tabla B-5 Entidad catálogo tipo sangre.**

| Cita | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idCita | Integer | No |
| Estatus | Boolean | No |
| Fecha | Date | No |
| Nota | String | No |
| Paciente | Paciente | Si |
| Medico | Medico | Si |
| Consultorio | Consultorio | No |

**Tabla B-6 Entidad Cita.**

| Consulta | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idConsulta | Integer | No |
| Fecha | Date | No |
| Talla | String | Si |
| Temperatura | String | Si |
| presionArterial | String | Si |
| Observaciones | String | No |
| frecuenciaCardiaca | String | Si |
| frecuenciaRespiratoria | String | Si |
| Estatus | Boolean | No |
| Enviado | Boolean | No |
| numeroArchivos | int | No |
| idPaciente | Paciente | No |
| idMedico | Medico | No |

**Tabla B-7 Entidad Consulta.**

| Consultorio | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idConsultorio | Integer | No |
| nombreConsultorio | String | No |
| Estatus | Boolean | No |
| idDireccion | Direccion | No |

**Tabla B-8 Entidad Consultorio.**

| DatosContacto | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idDatosContacto | Integer | No |
| Nombre | String | Si |
| correoElectronico | String | Si |
| numeroCelular | String | Si |
| Twitter | String | Si |
| Estatus | Boolean | No |
| idCita | Cita | No |

**Tabla B-9 Entidad Datos de contacto.**

| Direccion | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idDireccion | Integer | No |
| Calle | String | No |
| numeroExterior | String | No |
| numeroInterior | String | Si |
| codigoPostal | String | No |
| Colonia | String | No |
| Estatus | Boolean | No |

**Tabla B-10 Entidad Dirección.**

| Expediente | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idExpediente | Integer | No |
| CorreoElectronico | String | Si |
| Twitter | String | Si |

| | | |
|---|---|---|
| numeroCelular | String | Si |
| fechaCreacion | Date | No |
| Nota | String | Si |
| usuarioApp | UsuarioApp | No |
| catTipoSangre | CatTipoSangre | No |

Tabla B-11 Entidad Expediente.

| Medico | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| Cedula | String | No |
| Consultorio | Consultorio | No |

Tabla B-12 Entidad Médico.

| Paciente | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| Medico | Medico | No |

Tabla B-13 Entidad Paciente.

| Prescripcion | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| prescripcionPK | PrescripcionPK | No |
| intervaloHoras | String | No |
| duracionDias | String | No |
| Cantidad | String | No |
| Consulta | Consulta | No |
| catMedicamento | CatMedicamento | No |
| catEnfermedad | CatEnfermedad | No |

Tabla B-14 Entidad Prescripción.

| PrescripcionPK | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idConsulta | Int | No |
| idCatMedicamento | Int | No |
| idCatEnfermedad | Int | No |

**Tabla B-15 Entidad Prescripción Pk.**

| Rol | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idRol | Integer | No |
| nombreRol | String | No |

**Tabla B-16 Entidad Rol.**

| RSolicitudEstudios | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| rSolicitudEstudiosPK | RSolicitudEstudiosPK | No |
| Estatus | Boolean | No |
| catTipoEstudio | CatTipoEstudio | No |
| solicitudEstudio | SolicitudEstudio | No |

**Tabla B-17 Entidad Relación solicitud estudios.**

| RSolicitudEstudiosPK | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idSolicitudEstudio | Int | No |
| idTipoEstudio | Int | No |

**Tabla B-18 Entidad Relación solicitud estudios Pk.**

| SolicitudEstudio | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idSolicitudEstudio | Integer | No |
| Nota | String | Si |
| Indicaciones | String | Si |
| Estatus | Boolean | No |
| idConsulta | Consulta | No |

**Tabla B-19 Entidad Solicitud estudio.**

| UsuarioApp | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idUsuarioApp | Integer | No |
| Usuario | String | No |
| Password | String | No |
| Nombre | String | No |
| apellidoPaterno | String | No |
| apellidoMaterno | String | Si |
| fechaNacimiento | Date | No |
| Género | Integer | No |
| lugarNacimiento | String | No |
| Estatus | Boolean | No |
| idDireccion | Direccion | No |

**Tabla B-20 Entidad Usuario app.**

| UsuarioRol | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| usuarioRolPK | UsuarioRolPK | No |

| | | |
|---|---|---|
| usuarioApp | UsuarioApp | No |
| Rol | Rol | No |

Tabla B-21 Entidad Usuario Rol.

| UsuarioRolPK | | |
|---|---|---|
| **Atributo** | **Tipo de dato** | **Opcional** |
| idRol | Int | No |
| idUsuarioApp | Int | No |

Tabla B-22 Entidad Usuario rol Pk.

# ANEXO C. SERVICIOS DE ENTIDAD

| Servicio entidad usuario app | | | |
|---|---|---|---|
| **Nombre del servicio** | UsuarioAppDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao. | | |
| | | | |
| **Operación** | **buscarUsuario** | | |
| **Descripción** | busca un usuario (administrador del sistema, médico, auxiliar o paciente) de acuerdo al nombre de usuario y el password en MD5 | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | usuario | String | No |
| | password | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | UsuarioApp | | |
| | | | |
| **Operación** | **buscarUsuario** | | |
| **Descripción** | busca un usuario (administrador del sistema, médico, auxiliar o paciente ) por el nombre de usuario | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | usuario | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | UsuarioApp | | |

**Tabla C-1 Servicio entidad usuario app.**

| Servicio entidad relación usuario y rol | | | |
|---|---|---|---|
| **Nombre del servicio** | UsuarioRolDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao. | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda la relación entre un usuario y los roles que tiene | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | listUsuarioRol | List<UsuarioRol> | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-2 Servicio entidad relación usuario y rol.**

| Servicio Entidad Auxiliar | | | |
|---|---|---|---|
| **Nombre del servicio** | AuxiliarDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.AuxiliarDAO | | |
| | | | |
| **Operación** | **obtenerAuxiliares** | | |
| **Descripción** | Obtiene el personal auxiliar asociado a un consultorio dado | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsultorio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Auxiliar> | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda o actualiza la información de un personal Auxiliar | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | auxiliar | Auxiliar | No |

| Mensaje de salida | Tipo de dato |
|---|---|
| | N/A |

Tabla C-3 Servicio Entidad Auxiliar.

| Servicio entidad catálogo enfermedad | | | |
|---|---|---|---|
| **Nombre del servicio** | CatEnfermedadDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.CatEnfermedadDAO | | |
| | | | |
| **Operación** | **obtenerEnfermedad** | | |
| **Descripción** | Obtiene del catálogo de enfermedades la enfermedad indicada por el ID | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idEnfermedad | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | CatEnfermedad | | |
| | | | |
| **Operación** | **obtenerEnfermedades** | | |
| **Descripción** | Obtiene del catálogo de enfermedades todas las enfermedades con estatus activo, el primer campo indica la propiedad por la que serán ordenadas y el segundo si es verdadero (true) indica si el orden es ascendente, de lo contrario será descendente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Propiedad | String | No |
| | ascendente | Boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatEnfermedad> | | |

Tabla C-4 Servicio entidad catálogo enfermedad.

| Servicio entidad catálogo medicamento | | | |
|---|---|---|---|
| **Nombre del servicio** | CatMedicamentoDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao. CatMedicamentoDAO | | |
| | | | |
| **Operación** | **obtenerMedicamento** | | |
| **Descripción** | Obtiene del catálogo de medicamentos el medicamento indicado por el ID | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idMedicamento | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | CatMedicamento | | |
| | | | |
| **Operación** | **obtenerMedicamentos** | | |
| **Descripción** | Obtiene del catálogo de enfermedades todas las enfermedades con estatus activo, el primer campo indica la propiedad por la que serán ordenadas y el segundo si es verdadero (true) indica si el orden es ascendente, de lo contrario será descendente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Propiedad | String | No |
| | Ascendente | Boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatMedicamento> | | |

**Tabla C-5 Servicio entidad catálogo medicamento.**

| Servicio entidad catálogo tipo sangre | |
|---|---|
| **Nombre del servicio** | CatTipoSangreDAO |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao. CatTipoSangreDAO |
| | |

| Operación | obtenerTipoSangre | | |
|---|---|---|---|
| **Descripción** | Obtiene del catálogo de Tipo Sangre el tipo indicado por el Tipo sangre | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | tipoSangre | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | CatTipoSangre | | |
| | | | |
| **Operación** | **obtenerTiposSangre** | | |
| **Descripción** | Obtiene del catálogo de Tipo Sangre todos los tipos de sangre con estatus activo, el primer campo indica la propiedad por la que serán ordenadas y el segundo si es verdadero (true) indica si el orden es ascendente, de lo contrario será descendente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | propiedad | String | No |
| | Ascendente | Boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<CatTipoSangre> | | |

**Tabla C-6 Servicio entidad catálogo tipo sangre.**

| Servicio entidad Cita | | | |
|---|---|---|---|
| **Nombre del servicio** | CitaDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.CitaDAO | | |
| | | | |
| **Operación** | **obtenerCitasPaciente** | | |
| **Descripción** | Recupera todas las citas con estatus activo del paciente indicado | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idPaciente | Integer | No |

| Mensaje de salida | Tipo de dato | | |
|---|---|---|---|
| | List<Cita> | | |
| | | | |
| **Operación** | **obtenerCitasMedico** | | |
| **Descripción** | Obtiene todas las citas con estatus activo asignadas al médico indicado | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idMedico | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Cita> | | |
| | | | |
| **Operación** | **buscarCitas** | | |
| **Descripción** | Obtiene las citas activas que están en un rango de fechas | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | fechaInicio | Date | No |
| | fechaFin | Date | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Cita> | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda o actualiza una cita. | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | cita | Cita | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **Eliminar** | | |

| **Descripción** | Elimina la cita indicada por su ID | | |
|---|---|---|---|
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idCita | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-7 Servicio entidad Cita.**

| **Servicio entidad consultorio** | | | |
|---|---|---|---|
| **Nombre del servicio** | ConsultorioDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.ConsultorioDAO | | |
| | | | |
| **Operación** | **obtenerActivos** | | |
| **Descripción** | Obtiene los consultorios con estatus activo, el primer campo indica la propiedad por la que serán ordenadas y el segundo si es verdadero (true) indica si el orden es ascendente, de lo contrario será descendente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | propiedad | String | |
| | ascendente | boolean | |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Consultorio> | | |
| | | | |
| **Operación** | **obtenerPorId** | | |
| **Descripción** | Obtiene el consultorio indicado por el ID | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsultorio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | Consultorio | | |

| | | | |
|---|---|---|---|
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda o actualiza la información de un consultorio | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | consultorio | consultorio | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **Eliminar** | | |
| **Descripción** | Da de baja el consultorio indicado por su ID | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsultorio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | Consulta | | |

**Tabla C-8 Servicio entidad consultorio.**

| **Servicio entidad datos de contacto** | | | |
|---|---|---|---|
| **Nombre del servicio** | DatosContactoDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.DatosContactoDAO | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda o actualiza la información de contacto asociada a una cita cuando el paciente no está registrado en el sistema previamente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | datosContacto | DatosContacto | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-9 Servicio entidad datos de contacto.**

| Servicio entidad dirección | | | |
|---|---|---|---|
| **Nombre del servicio** | DireccionDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.DireccionDAO | | |
| | | | |
| **Operación** | **guardarDireccion** | | |
| **Descripción** | Guarda una dirección postal | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | direccion | Direccion | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-10 Servicio entidad dirección.**

| Servicio entidad expediente | | | |
|---|---|---|---|
| **Nombre del servicio** | ExpedienteDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.ExpedienteDAO | | |
| | | | |
| **Operación** | **buscarExpediente** | | |
| **Descripción** | Busca el expediente asociado a un usuario, ya sea médico, auxiliar o paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idUsuario | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda o actualiza la información de un expediente asociado a un | | |

| | usuario | | |
|---|---|---|---|
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | expediente | Expediente | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-11 Servicio entidad expediente.**

| **Servicio entidad médico** | | | |
|---|---|---|---|
| **Nombre del servicio** | MedicoDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.MedicoDAO | | |
| | | | |
| **Operación** | **obtenerMedicos** | | |
| **Descripción** | Obtiene los médicos activos asociados a un consultorio | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsultorio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Medico> | | |
| | | | |
| **Operación** | **obtenerActivos** | | |
| **Descripción** | Obtiene los médicos con estatus activo, ordenados conforme al nombre del primer parámetro indicado, el segundo indica que serán ordenados de forma ascendente en caso de ser verdadero, de lo contrario los ordena de forma descendente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | propiedad | String | No |
| | ascendente | boolean | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Medico> | | |

| | | | |
|---|---|---|---|
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda o actualiza la información de un médico | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | medico | Medico | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-12 Servicio entidad médico.**

| | | | |
|---|---|---|---|
| **Servicio entidad paciente** | | | |
| **Nombre del servicio** | PacienteDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.PacienteDAO | | |
| | | | |
| **Operación** | **obtenerPacientes** | | |
| **Descripción** | Devuelve los pacientes asociados a un médico dado, de acuerdo al número de cédula | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | Cedula | String | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Paciente> | | |
| | | | |
| **Operación** | **obtenerPacientes** | | |
| **Descripción** | Devuelve los pacientes asociados al consultorio indicado | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsultorio | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |

| | | | |
|---|---|---|---|
| | List<Paciente> | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda o actualiza la información de un paciente | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | paciente | Paciente | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-13 Servicio entidad paciente.**

| **Servicio entidad prescripción** | | | |
|---|---|---|---|
| **Nombre del servicio** | PrescripcionDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.PrescripcionDAO | | |
| | | | |
| **Operación** | **obtenerPrescripcion** | | |
| **Descripción** | Obtiene las prescripciones asociadas a una consulta | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsulta | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Prescripcion> | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | guarda o actualiza un conjunto de prescripciones | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | listaPrescripciones | List<Prescripcion> | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-14 Servicio entidad prescripción.**

| Servicio entidad rol | | | |
|---|---|---|---|
| **Nombre del servicio** | RolDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.RolDAO | | |
| | | | |
| **Operación** | **obtenerRol** | | |
| **Descripción** | Obtiene los roles indicados por la descripción | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | rol | String[] | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | List<Rol> | | |

**Tabla C-15 Servicio entidad rol.**

| Servicio entidad relación solicitud de estudios | | | |
|---|---|---|---|
| **Nombre del servicio** | RSolicitudEstudiosDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.RSolicitudEstudiosDAO | | |
| | | | |
| **Operación** | **guardarActualizar** | | |
| **Descripción** | Guarda la relación entre una solicitud de estudios y el nombre de los estudios necesarios | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | listaRSolicitud | List<RSolicitudEstudios> | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-16 Servicio entidad relación solicitud de estudios.**

| Servicio entidad solicitud de estudios | | | |
|---|---|---|---|
| **Nombre del servicio** | SolicitudEstudioDAO | | |
| **Ubicación Interfaz** | mx.egc.scmv.app.dao.SolicitudEstudioDAO | | |
| | | | |
| **Operación** | **obtenerSolicitudEstudio** | | |
| **Descripción** | Recupera la solicitud indicada de acuerdo a la consulta indicada. | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | idConsulta | Integer | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | solicitudEstudio | | |
| | | | |
| **Operación** | **guardarSolicitudEstudio** | | |
| **Descripción** | Guarda la información de una solicitud de estudios | | |
| **Mensaje de entrada** | **Nombre del atributo** | **Tipo de dato** | **Opcional** |
| | consulta | Consulta | No |
| | solicitudEstudio | SolicitudEstudio | No |
| | listaTipoEstudio | List<CatTipoEstudio> | No |
| **Mensaje de salida** | **Tipo de dato** | | |
| | N/A | | |

**Tabla C-17 Servicio entidad solicitud estudios.**

# ANEXO D. CONVENCIONES PARA ELECCIÓN DE NOMBRES

**Respecto a la aplicación**

Nombrado de servicios de entidad

- <Nombre de la entidad> + DAO

Nombrado de servicios de negocio

- <Nombre del servicio> + Negocio

Nombrado de servicios de aplicación

- <Nombre del servicio> + Integracion

Nombrado de operaciones. (Camel case)

- <Verbo infinitivo> + <sustantivo>

Verbos permitidos para el nombrado de las operaciones:

- Obtener
- Guardar
- Eliminar
- Reiniciar
- Cambiar
- Enviar
- Asignar

**Respecto a la base de datos**

Nombrado de las tablas:

Letras mayúsculas, palabras separadas por guion bajo

<NOMBRE_DE_LA_TABLA>

Nombrado de los campos:

Letras minúsculas separadas por guion bajo

<nombre_del_campo_de_tablan>

ANEXO D. Convenciones para la elección de nombre.

Nombrado de las llaves primarias

Se antepone “id” seguido del nombra de la tabla, cada palabra es separada por un guion bajo.

Id_nombre_tabla

Nombrado de las llaves foráneas.

Deben tener exactamente el mismo nombre de su tabla origen