Anexo 01

Versión 1.0.0

|  |
| --- |
| Filtro de baterías  SODIMAC |
| Arquitectura de Aplicación |

HISTORIAL DEL DOCUMENTO

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 09/04/2014 | 1.0.0 |  | Renzo Telenta |

# Historial de Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
|  |  |  |  |

# Apoyo Funcional - SODIMAC

|  |  |  |
| --- | --- | --- |
| **Área** | **Nombre** | **Rol** |
|  |  |  |

INDICE DE CONTENIDO

Historial de Versiones 2

Historial de Revisiones 2

Apoyo Funcional - SODIMAC 2

1.1 Objetivo 4

1.2 Alcance 4

2.1 Atributos de Calidad 5

2.1.1 Disponibilidad 5

2.1.2 Interoperabilidad 5

2.1.3 Rendimiento 5

2.1.4 Confiabilidad 6

2.1.5 Escalabilidad 6

2.1.6 Seguridad 6

2.1.7 Usabilidad 7

3.1 Introducción 8

3.1.1 Consideraciones de la Arquitectura y Diseño 8

3.1.2 Arquetipos de Solución 9

3.1.3 Uso de Arquetipo – Aplicaciones Orientada a Servicios 10

3.2 Arquitectura Lógica de Aplicación (Capas) 11

3.2.1 Implementación de Arquitectura Lógica de Aplicación 12

3.2.2 Capa de Presentación 13

3.2.3 Capa de Negocio 18

3.2.4 Capa de Datos 19

3.2.5 Capa Transversal 20

3.2.6 Tecnologías a Utilizar 20

3.3 Arquitectura Física de Aplicación (Niveles) 21

3.3.1 Diagrama de Despliegue 21

3.3.2 Especificación de Servidores 22

4. GLOSARIO 23

4.1 Terminología 23

1. **GENERALIDADES**

## Objetivo

El presente documento tiene por objetivo especificar la arquitectura de aplicación, con respecto a la implementación de los proyectos informáticos para SODIMAC, lo cual permitirá el manejo de una arquitectura estándar para próximos proyectos.

## Alcance

1. **REQUERIMIENTOS**

## Atributos de Calidad

### Disponibilidad

|  |  |  |  |
| --- | --- | --- | --- |
| **Código** | **DEFINICIÓN** | **Descripción** | **Criterios de Aceptación** |
| REQNF01 | El sistema no deberá bloquear o congestionar el ancho de banda en sus conexiones y extracción de información con otros sistemas. |  | No congestionar el ancho de banda a niveles de bloqueo funcional de otras aplicaciones. |

### 

### Interoperabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **DEFINICIÓN** | **DESCRIPCIÓN** | **CRITERIOS DE ACEPTACIÓN** |
| REQNF02 | El sistema deberá conectarse con una base de datos Mysql |  | Lectura de tablas de base de datos. |

### 

### Rendimiento

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **DEFINICIÓN** | **DESCRIPCIÓN** | **CRITERIOS DE ACEPTACIÓN** |
| REQNF04 | El sistema deberá tener una capacidad de respuesta para consultas externas (conexiones) de x segundos como tolerancia en entornos de cantidad de datos extensos (extracción de datos), en un entorno de baja demanda. | Se deberá de gestionar las posibilidades de cache a nivel de BD y de las diversas zonas HTML. | Tolerancia de XXX segundos. |

### Confiabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **DEFINICIÓN** | **DESCRIPCIÓN** | **CRITERIOS DE ACEPTACIÓN** |
| REQNF07 | El sistema deberá contar con un registro de los eventos relacionados a fallas. | Deberá ser fácilmente gestionada para identificar la falla y su solución. | Registro de logs de eventos en las operaciones principales de flujo de datos. |

### Escalabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **DEFINICIÓN** | **DESCRIPCIÓN** | **CRITERIOS DE ACEPTACIÓN** |
| REQNF08 | El sistema deberá ser diseñado y desarrollado sobre una plataforma orientada a servicios. | Deberá de definir y utilizar capas, orientándola a capas de datos, lógica, servicios e interfaz. | Arquetipo de solución basada en servicios. |
| REQNF09 | El sistema deberá estar basado en Tecnología Web. | Utilizando el protocolo HTTP para la aplicación web. | Ejecución a través de los navegadores web y protocolo http. |

### Seguridad

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **DEFINICIÓN** | **DESCRIPCIÓN** | **CRITERIOS DE ACEPTACIÓN** |
| REQNF12 | El sistema deberá de contar con mecanismos de autorización (accesos) al nivel de página por parte de los usuarios. | Deberá contar con una gestión de Roles, Usuarios y Perfil (acciones) las cuales estarán asociadas a páginas web de la aplicación. | El nivel de acceso debe ser el mismo que la Intranet (Usuario – Rol – Páginas). |

### Usabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **DEFINICIÓN** | **DESCRIPCIÓN** | **CRITERIOS DE ACEPTACIÓN** |
| REQNF13 | El sistema podrá ser usado sin inconvenientes como mínimo en los navegadores IE, Firefox, Chrome, Safari y Opera. |  | IE8+  Mozilla Firefox 7+  Chrome 11+  Safari 5+ |
| REQNF14 | El sistema considera una resolución mínima de pantalla de 320px de ancho, *responsive*. |  | Ancho de página establecido como mínimo para resolución de 1024x768 px. |
| REQNF15 | El sistema deberá contar con un diseño común para todas las páginas y asociada a la imagen empresarial de SODIMAC. | Deberá de contar con una página maestra que permita el diseño común y la rápida modificación general de datos (como datos de conexión visibles, menú general, logo, etc.) | De acuerdo a la imagen empresarial de SODIMAC, y el Layout (definición de plantilla general). |

1. **ARQUITECTURA DE APLICACIÓN**

## Introducción

La Arquitectura de la Aplicación, define la estructura de una aplicación en función de los requerimientos técnicos y funcionales, aquí se hace uso de los atributos de calidad como rendimiento, seguridad y escalabilidad. También involucra una serie de decisiones basadas en tecnología e implementación, ya que pueden tener un impacto considerable en la calidad, capacidad de mantenimiento y el éxito global de la aplicación. Por ejemplo se puede definir que una aplicación use manejo de sesiones en una base de datos para mejorar la escalabilidad de dicha aplicación o definir una arquitectura flexible al despliegue en capas que soporte entornos complejos que usen DMZ (Zona Desmilitarizada).

Una de las metas de la arquitectura es identificar los requerimientos que impactan la estructura de la aplicación, para realizar un diseño que sea suficientemente flexible para ser capaz de encargarse de las variaciones que ocurrirán a través del tiempo en tecnología de hardware y software. Se debe tener en mente que la arquitectura debería:

* Exponer la estructura de sistema pero ocultar los detalles de la implementación.
* Encargarse de los requerimientos funcionales y de calidad.

### Consideraciones de la Arquitectura y Diseño

Para el desarrollo de la aplicación, se define un marco arquitectónico con elementos claves, que lo forman, y a partir de ellos se desarrolla un enfoque de arquitectura adecuado. La línea base de la arquitectura para SODIMAC estará formada por las siguientes consideraciones:



Figura 3.1.1 – Consideraciones de Arquitectura y Diseño

Los requerimientos pueden ser funcionales y no funcionales, entre ellos tenemos características de la capa de presentación, cantidad de usuarios a consumir la aplicación (concurrencia), modelos de despliegue, interoperabilidad, entre otros.

Los temas transversales representan áreas de diseño que serán implementadas y consideradas en todas las aplicaciones y están asociadas a más de una capa o varias capas dentro de una aplicación.

La siguiente lista describe los temas transversales que se consideran en la aplicación:

* **Autenticación.** Determinar cómo autenticar los usuarios y pasar identidades a través de las capas.
* **Autorización.** Asegurar una autorización con granularidad apropiada en cada capa, y a través de límites de confianza.
* **Almacenamiento en caché.** Identificar que debe ser almacenado en caché, y donde almacenar el caché, para el mejor desempeño y responsabilidad de la aplicación.
* **Comunicación.** Elegir protocolos apropiados, reducir llamadas a través de la red, y proteger información sensitiva a través de la red
* **Gestión de excepción.** Capturar excepciones y ocultar la información sensible. No revelar información sensitiva a los usuarios finales.
* **Instrumentación y logging.** Personalizar todos los eventos críticos del sistema, y registrar detalles suficientes para recrear eventos en el sistema. No registrar información sensitiva.

### Arquetipos de Solución

En la siguiente tabla visualizamos los tipos de aplicación, sus ventajas y las consideraciones al momento de elegir una solución de acuerdo al escenario.

|  |  |  |
| --- | --- | --- |
| Tipo de Aplicación | Ventajas | Consideraciones |
| Aplicaciones para  dispositivos móviles | Sirven en escenarios sin conexión o con conexión limitada.  Se pueden llevar en dispositivos de mano.  Ofrecen alta disponibilidad y fácil acceso a los usuarios fuera de su entorno habitual. | Limitaciones a la hora de interactuar con la aplicación.  Tamaño de la pantalla reducido. |
| Aplicaciones de  escritorio | Aprovechan mejor los recursos de los clientes.  Ofrecen la mejor respuesta a la interacción, una interfaz más potente y mejor experiencia de usuario.  Proporcionan una interacción muy dinámica.  Soportan escenarios desconectados o con conexión limitada. | Despliegue complejo.  Versionado complicado.  Poca interoperabilidad. |
| RIA (Rich Internet  Applications) | Proporcionan la misma potencia gráfica que las aplicaciones de escritorio.  Ofrecen soporte para visualizar contenido multimedia.  Despliegue y distribución simples. | Algo más pesadas que las aplicaciones web.  Aprovechan menos los recursos que las aplicaciones de escritorio.  Requieren tener instalado un plugin para funcionar. |
| Aplicaciones  orientadas a servicios | Proporcionan una interfaz muy desacoplada entre cliente y servidor.  Pueden ser consumidas por varias aplicaciones sin relación.  Son altamente interoperables. | No tienen interfaz gráfica.  Necesitan conexión a internet. |
| Aplicaciones web | Llegan a todo tipo de usuarios y tienen una interfaz de usuario estándar y multiplataforma.  Son fáciles de desplegar y de actualizar. | Dependen de la conectividad a red.  No pueden ofrecer interfaces de usuario complejas. |

Tabla 3.1.1 – Arquetipos de Aplicación

### Uso de Arquetipo – Aplicaciones Orientada a Servicios

El Arquetipo seleccionado para SODIMAC es “Aplicaciones Orientadas a Servicios”, donde el diseño de la solución estará basado en una Arquitectura Orientada a Servicios (SOA) donde se expondrá la funcionalidad de la aplicación como un conjunto de servicios. Los servicios se exponen en base a contratos e interfaces. Los servicios deben ser invocados, publicados, y descubiertos. Los servicios SOA brindan un esquema de interacción basada en mensajes. Los principales beneficios son:

* **Creación de un modelo de dominio**. La definición de un conjunto de entidades en los contratos a nivel de negocio permiten reusar la definición del negocio entre las aplicaciones.
* **Abstracción**. Los servicios son autónomos y accedidos a través de un contrato formal, con él se logra acoplamiento y abstracción.
* **Capacidad de descubrimiento**. Los servicios pueden exponer metadata que permiten a otras aplicaciones y servicios localizarlos y consumirlos.

Se considera el uso de este estilo, pues:

* Se quiere construir aplicaciones que compongan una variedad de servicios.
* Se requiere implementar más adelante aplicaciones de tipo BPM para que sean accedidos por este tipo de aplicaciones.
* Se quiere soportar interoperabilidad e integración entre diferentes aplicaciones.
* Brinda una capa de negocio el cual se puede integrar fácilmente con una capa de presentación (Aplicación Web).

## Arquitectura Lógica de Aplicación (Capas)

Para realizar la implementación se debe tener en cuenta que el sistema debe poseer características de sostenibilidad, eficiencia y usabilidad para cualquier usuario. Así mismo deberá estar alineado con los objetivos de una arquitectura basada en capas y presentar características claras de modularidad, como se muestra a continuación:

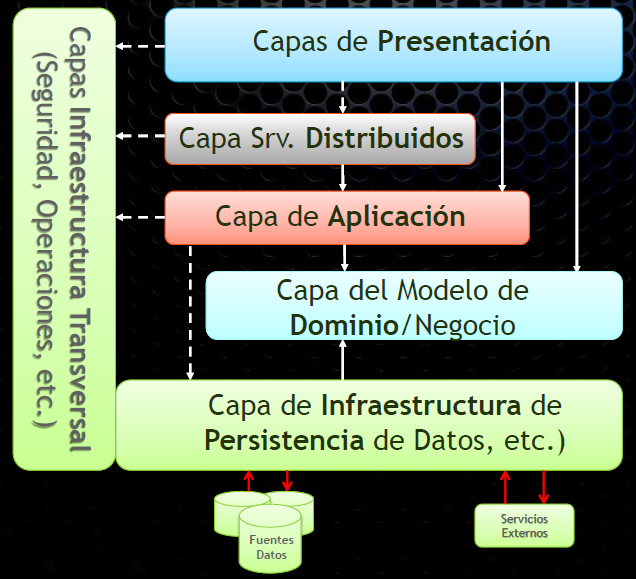


Figura 3.2.1 – Vista de Arquitectura Lógica

El diseño de cada capa será cohesivo, pero delimitando claramente las diferentes capas entre ellas, aplicando patrones estándar de Arquitectura para que dichas dependencias sean en muchas ocasiones basadas en abstracciones y no referenciando una capa directamente a la otra.

Cada capa de la aplicación contendrá una serie de componentes que implementan la funcionalidad de dicha capa. Estos componentes serán cohesivos internamente (dentro de la misma capa de primer nivel), pero algunas capas (como las capas de Infraestructura/Tecnología) estarán débilmente acopladas con el resto de capas para poder potenciar las pruebas unitarias, mocking, la reutilización y finalmente que impacte menos al mantenimiento.

Por ende la vista de Arquitectura Lógica es un modelo de alto nivel de la relación existente entre las capas. A partir de este modelo se presenta el modelo de implementación de arquitectura de aplicación. La arquitectura extendida puede verse de la siguiente manera:



Figura 3.2.2 – Arquitectura Lógica de Aplicación

### Implementación de Arquitectura Lógica de Aplicación

La implementación de la Arquitectura Lógica de Aplicación debe estar basada en las siguientes capas:

* Capa de Presentación
* Capa de Negocio
* Capa de Datos
* Capa Transversal

### Capa de Presentación

La capa de presentación contiene los componentes que implementan la pantalla del usuario interfaz y gestión de la interacción del usuario. Esta capa incluye los controles de entrada del usuario y pantalla, además de los componentes que organizan la interacción del usuario. La figura muestra cómo la capa de presentación se inscribe en una arquitectura de aplicación común.



Figura 3.2.3 – Arquitectura de Aplicación – Capa de Presentación

La capa de presentación tendrá componentes, como se detalla a continuación:

* **ASP.Net MVC4**: Este patrón ha sido uno de los más importantes e influyentes de la historia de la programación, tanto que hoy en día sigue siendo un patrón actual y ampliamente utilizado. Este patrón pertenece a un conjunto de patrones agrupados en el estilo arquitectural de presentación separada.

El objetivo que buscamos al utilizar este patrón es separar el código responsable de la representación de los datos en pantalla, del código encargado de la ejecución de la lógica de negocio. Para ello el patrón divide la capa de presentación en tres tipos de objetos básicos: modelos, vistas y controladores. Para mayor detalle se describe el flujo de comunicación entre estos tres tipos de objetos, como muestra en esta imagen:

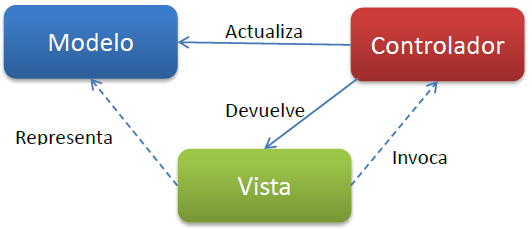


Figura 3.2.4 – Patrón MVC

* **Vistas IU**: Las vistas son las encargadas de representar gráficamente el modelo y de ofrecer las acciones de los controladores para que el usuario pueda interactuar con este. Hay que tener muy claro que una vista no debe invocar ningún método que provoque un cambio de estado en el modelo o llamar a algún método que requiera parámetros, es decir, debe limitarse a acceder a propiedades simples y métodos de consulta del objeto que no tengan parámetros.
* **Controladores**: El controlador orquesta la interacción entre las vistas y el modelo. Recibe las peticiones del usuario, interactúa con el modelo realizando consultas y modificaciones a este, decide que vista se muestra en respuesta y le proporciona los datos requeridos para su renderizado, o delega la respuesta a otra acción de otro controlador.
* **Helpers**: Pequeños métodos que facilitan la escritura de HTML y simplifican la creación de las vistas.
* **AJAX**: acrónimo de Asynchronous Java Script And XML (Java Script asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en un segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.
* **Jquery**: Es una biblioteca de Java Script rápida y concisa que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
* **Controladores de Servicio**: Sobre este componente se implementaran las rutinas correspondientes a las llamadas de los servicios de la aplicación.
* **Proxies Servicio**: Sobre este componente reposan todos los Proxies (Referencia a un programa o dispositivo que realiza una acción en representación de otro) que serán obtenidos a través de los servicios expuestos por **SODIMAC**.

Por una parte la capa de presentación será consumida por un tipo de Cliente:

* **Clientes Web:** Estaciones de trabajo (Terminales) que tendrán acceso a la Aplicación **SODIMAC** mediante una URL definida.



Figura 3.2.5 – Arquitectura de Aplicación - Clientes Finales

Por otra parte la capa de presentación consumirá los siguientes servicios de la capa de servicios.

* **Servicios de Reportes:** Dentro de la solución de SODIMAC se utilizara “SQL Server Reporting Services” que es un componente de Microsoft SQL Server utilizado para la generación de reportes. Permitiendo que el autor del reporte defina los datos y la manera de presentación de estos. En esta etapa normalmente hay que definir conexiones a los distintos orígenes de datos para ver de dónde obtener los resultados que debe reflejar el reporte.



Figura 3.2.6 – Arquitectura de Aplicación – Servicios de Reportes

* **Servicios de Integración**: Dentro de la solución de SODIMAC se utilizara “SQL Server Integration Services (SSIS)” que es un componente de Microsoft SQL Server utilizado para migración de datos. SSIS es una plataforma para la integración de datos y sus de flujos de trabajo. Permite mover datos de origen a destino sin modificar los datos. Se pueden importar datos de fuentes diferentes a SQL Server.



Figura 3.2.7 – Arquitectura de Aplicación – Servicios de Integración

Los servicios de integración también proporcionan un amplio conjunto de tareas, contenedores, transformaciones y adaptadores de datos integrados que permiten desarrollar aplicaciones de negocios. Sin escribir una sola línea de código, se puede crear soluciones de SSIS para resolver problemas de negocios complejos mediante ETL y Business Intelligence, administrar bases de datos de SQL Server y copiar objetos de SQL Server entre instancias de SQL Server.

En los siguientes escenarios se describen usos típicos de los paquetes SSIS.

* Combinar datos de almacenes de datos heterogéneos
* Llenar almacenamiento de datos y puestos de datos

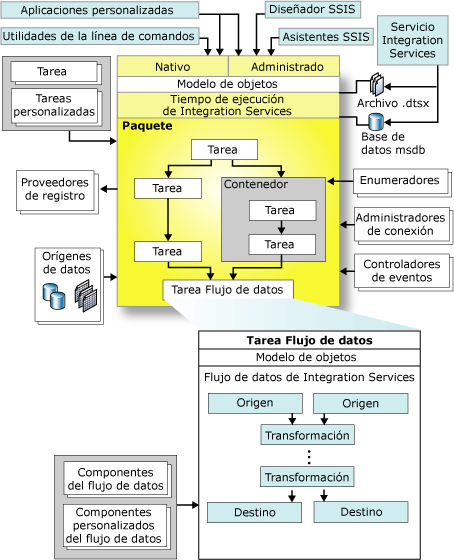


Figura 3.2.8 – Arquitectura de Aplicación – SSIS

* **Servicios de Seguridad:** En este punto se utilizara la seguridad de ASP.NET en conjunción con la seguridad de Microsoft Internet Information Services (IIS) ya que incluye servicios de autenticación y autorización para implementar el modelo de seguridad en la solución de SODIMAC. A continuación se detalla los mecanismos de seguridad a utilizar:
  + **Autenticación:** Para el manejo de autenticación de un usuario en la aplicación web ASP.NET MVC de SODIMAC, se deberá utilizar el proveedor **SodimacMembershipProvider** que permite validar el registro de la información de usuario contra el repositorio de credenciales que se encuentra en un gestor de base de datos (Por ejemplo el método ValidateUser).
  + **Autorización:** Para el manejo de autorización que se debe realizar para validar el acceso a los recursos que tiene un determinado usuario en la aplicación web ASP.NET MVC de SODIMAC, se deberá utilizar el proveedor **SodimacRoleProvider**.

Para el manejo de autorización basado en un esquema de acceso a páginas mediante un mapa de sitio, se deberá utilizar el proveedor **SodimacSiteMapProvider**. El proveedor obtendrá la información de opciones menú desde la Base de Datos y así construir el mapa de sitio, permitiendo administrar el acceso al mapa de sitio de un determinado usuario en la aplicación.



Figura 3.2.9 – Arquitectura de Aplicación – Servicios de Seguridad

### Capa de Negocio

La capa de negocio es el “puente” entre un usuario y los servicios de datos. Responden a peticiones del usuario (u otros servicios de negocios) para ejecutar una tarea de este tipo. Cumplen con esto, aplicando procedimientos formales y reglas de negocio a los datos relevantes. Cuando los datos necesarios residen en un servidor de bases de datos, garantizan los servicios de datos indispensables para cumplir con la tarea de negocios o aplicar su regla. Esto aísla al usuario de la interacción directa con la base de datos.

La definición de lógica de negocio se implementara bajo un arquetipo Orientado a Servicios. Bajo estas consideraciones es necesario implementar un patrón “Service Interface” que defina la creación inicial de contratos a manera de entidades y que estas sean las que viajen a lo largo de toda la interacción entre los clientes y el servicio.



Figura 3.2.14 – Arquitectura de Aplicación – Capa de Negocios

La capa de negocios tendrá componentes, como se detalla a continuación:

**Servicios WCF:** Esta capa define las acciones previstas por el servicio, los mensajes necesarios para interactuar con cada operación, y los patrones por los que estos mensajes deben interactuar, estos patrones se conocen como los patrones de intercambio de mensajes.

* **Service Contract:** Une varias operaciones relacionadas en una unidad funcional única. El contrato puede definir ajustes del nivel de servicio, como el espacio de nombres del servicio, un contrato de devolución de llamadas correspondiente y otros ajustes de este tipo. En la mayoría de los casos, el contrato se define mediante la creación de una interfaz en el lenguaje de programación que elija y la aplicación del atributo ServiceContractAttribute a la interfaz. El código de servicio real resulta mediante la implementación de la interfaz.
* **Message Contract:** Describe el formato de un mensaje. Por ejemplo, declara si los elementos del mensaje deberían ir en encabezados en lugar de en el cuerpo, qué nivel de seguridad debería aplicarse a qué elementos del mensaje, etc.
* **Service Implementation:** Implementación del servicio, representando la ejecución del service contract.
* **Fault Contract:** Puede estar asociado a una operación de servicio para denotar errores que se pueden devolver al autor de la llamada. Una operación puede tener cero o más errores asociados a ella. Estos errores son errores de SOAP que se modelan como excepciones en el modelo de programación.

**Componentes de Negocio:**

* **Core SODIMAC:** Capa que implementa todos los requerimientos a nivel de lógica teniendo en cuenta cada proceso especificado.
* **Validaciones Core:** Capa de validación que permite aplicar las reglas de negocio contempladas en el Core **SODIMAC**.
* **Manejo de Excepciones:** Control de excepciones centralizada que permite administrar de manera más eficiente los errores suministrados por aplicaciones externas como internas.
* **Recursos de Validación:** Permite almacenar principalmente los mensajes de validación a nivel de Core **SODIMAC** como a nivel de interfaces.
* **Utilitarios:** Métodos que exponen funcionalidad común en toda la Capa de Negocio.

### Capa de Datos

Esta capa posee toda la lógica de acceso e interacción con las diferentes fuentes de datos involucradas en la solución del sistema.



Figura 3.2.15 – Arquitectura de Aplicación – Capa de Negocios

Estos componentes proveen el acceso a los datos que está ubicado dentro de los límites del sistema y datos expuestos por otros subsistemas. Los siguientes componentes que comúnmente se encuentran en esta capa se describen a continuación:

**Servicio de Datos:** Estos componentes manejan la lógica requerida para acceder a los datos almacenados. De esta manera la funcionalidad de acceso a datos se centraliza y permite fácil configuración.

* **Componentes de Acceso a Datos:** Es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, en este caso SQL Server.
* **Lenguaje Integrado de Consulta (LINQ):** Define operadores de consulta estándar que permiten filtrar, enumerar y crear proyecciones de varios tipos de colecciones usando la misma sintaxis. Tales colecciones pueden incluir vectores (arrays), clases enumerables, XML, conjuntos de datos desde bases de datos relacionales y orígenes de datos de terceros.

Adicionalmente se utilizara el gestor de base de datos SQL Server 2008



Figura 3.2.16 – Arquitectura de Aplicación – Capa de Negocios

### Capa Transversal

En esta capa se pueden apreciar componentes globales que se caracterizan porque pueden ser usados a través de todas las capas de la arquitectura.

* **Enterprise Library 5.0:** Es una colección de componentes de software reutilizables (bloques de aplicación) diseñados para ayudar a los desarrolladores de software empresariales con el desarrollo de cuestiones transversales comunes como el registro, validación, acceso a datos, manejo de excepciones, y muchos otros.

### Tecnologías a Utilizar

Las tecnologías que se deben de considerar son:

* Plataforma y entorno de desarrollo
  + .NET Framework 4.0 y Visual Studio 2010 SP1
  + Lenguaje de Programación Visual Basic .NET
* Capa de Presentación
  + ASP .NET MVC 4.0
* Capa de Negocio
  + Windows Communication Foundation
* Capa de Datos
  + SQL Server 2008 R2 Estándar
    - SQL Server Reporting Services 2008 R2
    - SQL Server Integration Services 2008 R2
* Capa Transversal
  + Enterprise Library 5.0

## Arquitectura Física de Aplicación (Niveles)

### Diagrama de Despliegue

El siguiente grafico representa el diagrama de despliegue de los servidores de SODIMAC relacionados a los proyectos de desarrollo por parte de 3Dev, para un ambiente de producción:

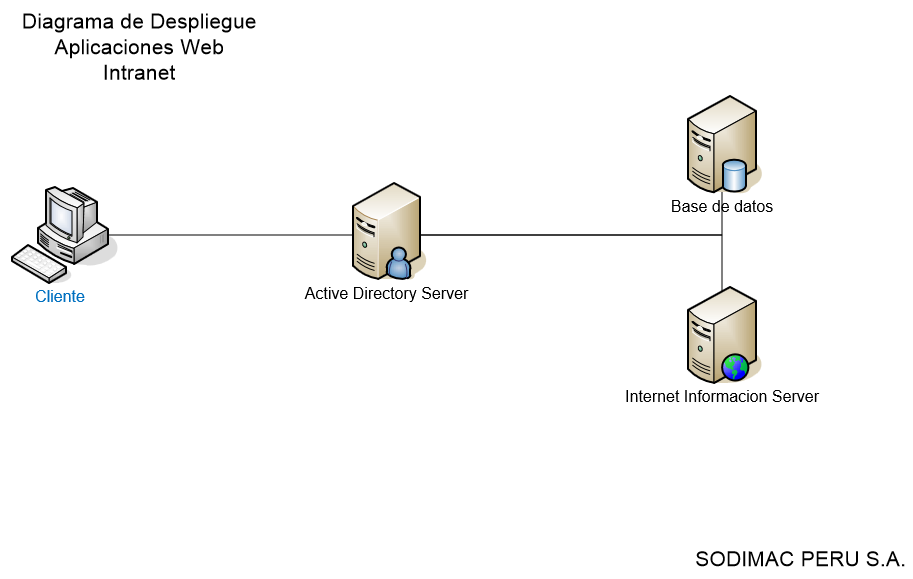


Figura 3.3.1 – Arquitectura Física de Aplicación – Diagrama de Despliegue

# GLOSARIO

## Terminología

* **Aplicación Web:** Es un sitio Web que implementa la lógica de algún negocio. La diferencia entre una aplicación Web y un sitio Web es que la aplicación Web afecta al estado del negocio.
* **MVC (Modelo Vista Controlador)**: Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.
* **Sitio Web**: Es un conjunto de páginas con contenido sin determinar parcialmente o en su totalidad. El contenido final de éstas se determina sólo cuando un visitante solicita una página al servidor Web. El contenido final de la página varía de una petición a otra en función de las acciones del visitante.
* **TCP/IP**: Protocolo de control de transmisión basado en IP. Se trata de un protocolo de internet que proporciona la entrega fiable de datos de un sistema a otro.
* **Usuario**: Persona a la que se le otorga una identificación o cuenta de usuario y una contraseña (password o clave de acceso) que le permitirán acceder a información.