ARQUITECTURA SILESWEB

1. LISTA DE ABREVIATURAS

**CSV:** (En inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

**C#:** Es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework.

**DTO:** (En inglés Data Transfer Object), es un objeto que transporta datos entre procesos.

**HTML:** (En inglés HiperText Markup Language), Es un lenguaje es se utiliza común-mente para establecer la estructura y contenido de un sitio web, tanto de texto, objetos e imágenes.

**HTTP:** (En inglés Hypertext Transfer Protocol), que se utiliza en algunas direcciones de internet

**JSON:** (En Inglés JavaScript Object Notation), es un formato de texto ligero para el intercambio de datos

**.NET:** Es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

**MVC:** El Modelo Vista Controlador, es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

**ORM:** (En inglés Object-Relational mapping), o lo que es lo mismo, mapeo de objeto relacional, es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador.

**PDF:** (En Inglés Portable Document Format), es un formato de almacenamiento para documentos digitales independiente de plataformas de software o hardware.

**UI:** (En inglés User interface) se refiere a la creación de la interfaz, ya sea gráfica o desarrollada con diversas tecnologías web.

**UML:** (En inglés Unified Modeling Language), que es el lenguaje de modelado de sis-temas de software más conocido y utilizado en la actualidad.

**URL:** (En inglés Uniform Resource Locator), es una secuencia de caracteres que se utiliza para nombrar y localizar recursos, documentos e imágenes en Internet.

**VPN:** acrónimo de red privada virtual.

**W3C:** Son las siglas en inglés de World Wide Web Consortium, un consorcio fundado en 1994 para dirigir a la Web hacia su pleno potencial mediante el desarrollo de protocolos comunes que promuevan su evolución y aseguren su interoperabilidad.

**WCF:** (En inglés Windows Communication Foundation), es un marco de trabajo para la creación de aplicaciones orientadas a servicios.

**WSDL:** (En inglés Web Services Description Language), un formato XML que se utiliza para describir servicios Web.

**XML:** (En inglés eXtensible Markup Language, es un lenguaje de marcas desarrollado por el W3C.

1. DEFINICIONES

**ADO.NET:** Es un conjunto de clases que exponen servicios de acceso a datos para programadores de .NET Framework.

**AJAX:** Es una técnica que permite la comunicación asíncrona entre un servidor y un navegador en formato XML mediante programas escritos en Javascript.

**Aplicación Cliente:** Programa que intercambia mensajes con uno o varios extremos. La aplicación de cliente comienza creando una instancia de un cliente de WCF y llamando métodos del cliente de WCF. Es importante tener en cuenta que una aplicación individual puede ser tanto un cliente como un servicio.

**Arquitectura Conceptual:** Describe la estructura básica de la solución, explicando claramente los conceptos utilizados. Identifica por un lado los Requerimientos Funcionales de Arquitectura (los del negocio) basado en los Casos de Uso y los Requerimiento No Funcionales de Arquitectura (Seguridad, Comunicaciones, Administración, Flexibilidad, etc.). Incluye en general las llamadas en la literatura Vista Conceptual y Vista de Casos de Uso.

**Arquitectura Lógica:** Describe los componentes lógicos del sistema, su estructura interna básica, y sus relaciones. Incluye los componentes lógicos tanto de negocio (funcionales) como de sistemas (no funcionales). Normalmente es independiente de la Plataforma tecnológica en la que se implementará el sistema. Incluye la llamada Vista Lógica. A veces incluye la Vista de Datos cuando se describe la persistencia de ciertas entidades de negocios.

**Arquitectura Física:** Describe las tecnologías que se deben usar de la Plataforma escogida y decisiones básicas sobre asociación de componentes lógicos y físicos. Describe decisiones sobre la Plataforma (en el caso de SIIF Nación explotación de tecnologías .NET, frameworks, librerías, patrones), las decisiones sobre Arquitectura de Hardware (tipos de servidores, tipos de clientes) y guías de diseño en .NET. Incluye tópicos de las llamadas Vista de Implementación y Vista de Deployment o Puesta en Marcha.

**ASP.NET:** Es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.

**Balanceador:** Es un dispositivo de hardware o software que se pone al frente de un conjunto de servidores que atienden una aplicación y, tal como su nombre lo indica, asigna o balancea las solicitudes que llegan de los clientes a los servidores usando algún algoritmo.

**Canal:** Implementación concreta de un elemento de enlace. El enlace representa la configuración, y el canal es la implementación asociada a esa configuración. Por consiguiente, hay un canal asociado a cada elemento de enlace. Los canales se apilan uno sobre otro para crear la implementación concreta del enlace: la pila de canales.

**Casos de Uso (CU):** Los Casos de Uso son narrativas estructuradas de los requerimientos funcionales de un sistema. Incluyen también algunos de los requerimientos no funcionales. Son la base fundamental para desarrollar una arquitectura. En SIIF Nación existen casos de uso que están descritos en narrativas y modelados en UML. Los Casos de Uso se describen en la narrativa a tres niveles de detalle: Negocio, Resumen y de Sistema.

**Capas:** El concepto de capas surge luego de la evolución de los sistemas cliente-servidor a sistemas distribuidos.

Es importante diferenciar las “capas lógicas” (layer) de las “capas físicas” (tier) de un sistema ya que son fácilmente confundibles. Las primeras representan una estrategia de implementación con el objetivo de beneficiar la portabilidad del software, las segundas están relacionadas con la distribución del software físicamente en los servidores o elementos de infraestructura con el objetivo de centralizar el procesamiento y lógica de negocio de un sistema.

**Certificado Digital:** Documento Firmado digitalmente, emitido por una Entidad Certificadora o un Prestador de Servicios de Identificación, que permite identificar un usuario o un sistema, garantizando la integridad del mensaje y el No repudio.

**Colección:** Una colección es una clase, de modo que antes de poder agregar elementos a una nueva colección, debe declararla.

**Componente:** Los componentes suelen mapearse con procesos o módulos que se implementan físicamente dentro del software. De ahí, que presentan atributos menos conceptuales que los que se presentan en los módulos.

**Comportamientos:** Componente que controla varios aspectos del tiempo de ejecución de un servicio, un extremo, una operación determinada o un cliente. Los comportamientos están agrupados en función del ámbito: los comportamientos comunes afectan globalmente a todos los extremos, los comportamientos de servicios sólo afectan a los aspectos relacionados con servicios, los comportamientos de extremos sólo afectan a las propiedades relacionadas con los extremos y los comportamientos de operaciones afectan a las operaciones determinadas. Por ejemplo, un comportamiento del servicio está limitando que especifica cómo un servicio reacciona cuando un exceso de mensajes amenaza agobiar sus funciones del control. Un comportamiento de extremos, por otro lado, solo controla los aspectos relacionados con los extremos, como, por ejemplo, cómo y dónde encontrar una credencial de seguridad.

**Contratos:** Definen varios aspectos del sistema de mensajes y las capacidades del servicio, existen tres tipos de datos, de mensaje y de servicio.

**DataSets:** Es una representación de datos residente en memoria que proporciona una modelo de programación relacional coherente independientemente del origen de datos que contiene.

**Datareader:** Se utilizar para recuperar una secuencia de datos de sólo avance y de sólo lectura desde una base de datos. Los resultados se devuelven cuando se ejecuta la consulta y se almacenan en el búfer de red del cliente hasta que el usuario los solicite.

**Delegados anónimos:** La creación de métodos anónimos es básicamente una forma de pasar un bloque de código como parámetro de delegado.

**Dirección:** Especifica la ubicación donde se reciben los mensajes. Se especifica como un identificador uniforme de recursos (URI). La parte del esquema URI nombra el mecanismo de transporte que se ha de utilizar para alcanzar la dirección, por ejemplo, HTTP y TCP. La parte jerárquica del URI contiene una ubicación única cuyo formato depende del mecanismo de transporte.

La dirección del extremo le permite crear direcciones únicas de extremos para cada extremo de un servicio o, bajo ciertas condiciones, compartir una dirección en los extremos.

**Enlace:** Define cómo se comunica un extremo con el mundo. Consta de un conjunto de componentes llamados elementos de enlace que se "apilan" uno sobre el otro para crear la infraestructura de comunicaciones. Como mínimo, un enlace define el transporte (como HTTP o TCP) y la codificación utilizada (por ejemplo, de texto o binaria). Un enlace puede contener elementos de enlace que especifican detalles, por ejemplo, los mecanismos de seguridad utilizados para proteger los mensajes o el patrón de mensaje utilizado por un extremo.

**Elemento de enlace:** Representa una parte determinada del enlace, por ejemplo, un transporte, una codificación, una implementación de un protocolo del nivel de infraestructura (como WS-Reliable Messaging), o cualquier otro componente de la pila de comunicaciones.

**Enterprise Library:** Es un conjunto de librerías que facilitan el desarrollo de aplicaciones empresariales en .NET.

**Entidad:** Para este documento las entidades que hacen parte del SilesWeb.

**Entity Framework:** Es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos.

**Entity Data Model:** Es la definición del mapeo que se crea entre la base de datos y el modelo conceptual o de entidades.

**Escalabilidad:** Es la capacidad de mejorar recursos para ofrecer una mejora (ideal-mente) lineal en la capacidad de servicio. La característica clave de una aplicación es que la carga adicional sólo requiere recursos adicionales en lugar de una modificación extensiva de la aplicación en sí.

**Firewall:** Es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas

**Framework:** Es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones como el soporte de programa, bibliotecas, plantillas y más

**Facade Pattern:** (Patrón Factory), Conoce qué clases del subsistema son responsables de una determinada petición, y delega esas peticiones de los clientes a los objetos apropiados del subsistema.

**Foreign Key (FK):** Es una columna o combinación de columnas que se utiliza para establecer y exigir un vínculo entre los datos de dos tablas.

**Generics:** Permite diseñar clases y métodos que aplazan la especificación de uno o más tipos hasta que el código de cliente declara y crea una instancia de la clase o del método.

**Interfaz:** Es la definición de un conjunto de métodos para los que no se da implementación, sino que se les define de manera similar a como se definen los métodos abstractos.

**Javascript:** Es un lenguaje de programación que se puede utilizar para construir sitios Web y para hacerlos más interactivos.

**Jquery:** es un framework de JavaScript para facilitar, entre otros, el acceso a los elementos del DOM, los efectos, interactuar con los documentos HTML

**Layout:** Permite definir una plantilla común para un sitio, y heredar su look and feel por todas las vistas/páginas de nuestro sitio.

**Look and Feel:** es una expresión inglesa que puede tener diferentes significados, de-pendiendo del contexto en que se utilice.

**Macroproceso:** Son el conjunto de procesos que contribuyen, en forma sistémica, a satisfacer los requerimientos de la comunidad y de la Entidad para lograr el cumplimiento y los fines propios del Estado.

**Mensajería:** es la capa donde se crea el canal, un canal es un componente que procesa el mensaje a nivel de transporte y de protocolo.

**Message Inspector:** Es un objeto de extensibilidad que se puede utilizar en el cliente del modelo del servicio en tiempo de ejecución y enviarse mediante programación en tiempo de ejecución o a través de configuración, y que puede inspeccionar y modificar los mensajes una vez recibidos o antes de enviarse.

**Metadatos:** Los metadatos expuestos por el servicio incluyen documentos de esquema XML, que definen el contrato de datos del servicio, y documentos WSDL, que des-criben los métodos del servicio.

Cuando se habilita, WCF genera automáticamente los metadatos para el servicio mediante la inspección del servicio y sus extremos. Para publicar metadatos desde un servicio, se ha de habilitar explícitamente el comportamiento de los metadatos.

**Microsoft Data Access Application Block:** Consiste de un ensamblado denominado "Microsoft.ApplicationBlocks.Data", el cual contiene toda la funcionalidad necesaria para realizar la mayoría de las funciones de acceso a datos sobre la base de datos Microsoft SQL Server.

**Módulo:** Es el artefacto principal de las vistas de “tipo de módulos”. Se utilizan con el objetivo de descomponer el software en estructuras y definir responsabilidades sobre ellas. Los módulos pueden representar conceptos teóricos que pueden o no ser volcados a estructuras físicas dentro del desarrollo.

**Namespace:** se utiliza para declarar un ámbito que contiene un conjunto de objetos relacionados. Puede utilizar un espacio de nombres para organizar elementos de código y crear tipos globales únicos.

**Nueva Arquitectura:** Describe la arquitectura con la que se implementan los nuevos módulos que se han ido desarrollando para SIIF Nación (Ej. CUN, DYC).

**Objeto:** Es una unidad dentro de un programa de computadora que consta de un estado y de un comportamiento.

**Página Maestra:** Permiten crear un diseño coherente de las páginas de la aplicación. Una sola página maestra define la apariencia y el comportamiento estándar que desea para todas las páginas (o un grupo de páginas) en su aplicación

**Patrón de Arquitectura de software:** Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados.

**Patrón Decorator:** Responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto nos permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se aso-cian a la primera.

**Proxy:** Programa o dispositivo que realiza una acción en representación de otro.

**Reporting Services:** Es una plataforma de informes basada en servidor que proporciona la funcionalidad completa de generación de informes para una gran variedad de orígenes de datos.

**Reflection:** El patrón de reflexión otorga la habilidad a un programa para inspeccionar su estructura interna y poder modificar a ésta misma en tiempo de ejecución y por tanto, su comportamiento.

**ReportViewer:** Es un control AJAX de ASP.NET que se utiliza para hospedar informes en proyectos de ASP.NET.

**Scale out:** Escalabilidad horizontal consiste en añadir más máquinas a la aplicación, aumentando su número, aunque no necesariamente su potencia.

**Service Interface Pattern:** Se utiliza cuando se quiere que una clase que hace uso de los servicios proporcionados por otras clases, permanezca independiente de estas.

**Servicios:** Los servicios se definen como componentes de software auto-contenidos y reusables independientes de las aplicaciones que los ejecutan. Los servicios tienen interfaces bien definidas y deben ser capaces de proveer un mapeo uno a uno (1:1) entre las tareas de negocio y elementos de software.

**Servicios Web:** Un servicio web (en inglés, web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

**Sistemas Transversales:** Son sistemas de obligatorio uso, administrados por una entidad gubernamental y las entidades deben registrar y/o reportar a esos sistemas.

**Sistemas Locales (Entidades):** Aplicativos propios de las entidades que hacen parte del PGN, en los que registran los procesos administrativos y/o misionales, que generan información para ser cargada en el SIIF Nación y generan reportes e información que requiere la Entidad en su operatividad.

**SQL Server:** Es un sistema de gestión de bases de datos relacionales (RDBMS) de Microsoft que está diseñado para el entorno empresarial.

**Stakeholder:** Es una palabra del inglés que, en el ámbito empresarial, significa 'interesado' o 'parte interesada', y que se refiere a todas aquellas personas u organizaciones afectadas por las actividades y las decisiones de una empresa.

**Store Procedure (procedimiento almacenado):** Es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos. La ventaja de utilizar un procedimiento almacenado en respuesta a una petición de usuario, es la ejecución directa en el motor de bases de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

**Tiempo de ejecución:** Instancia los nuevos servicios y su comportamiento.

**Tipos de vistas:** En el presente documento se pueden identificar tres tipos de vistas: Vistas orientadas a mostrar aspectos estáticos del software como su descomposición en estructuras (tipo de vista de módulos), vistas cuyo propósito es mostrar aspectos dinámicos de comunicación, procesamiento, concurrencia, ejecución, etc. (tipo de vista de componentes y conectores) y vistas orientadas a asignar las estructuras identifica-das a recursos físicos de hardware o de sistema operativo (tipo de vista de asignación).

**Transacciones:** Son unidades lógicas de trabajo. Son un conjunto de acciones de ne-gocio o de su administración con un sentido lógico para el usuario. En SIIF NACIÓN corresponde prácticamente uno a uno con los Casos de Uso de Sistema. SIIF NACIÓN está especificado de tal forma que parte de la funcionalidad gira alrededor de este con-cepto: una transacción es un elemento con propiedades claves de seguridad, una transacción es una un paso dentro de un proceso, una transacción tiene propiedades.

**Viewpoint:** Son puntos de vistas sobre intereses específicos de los Stakeholders del sistema.

**Vistas:** Es el concepto fundamental asociado a la documentación de software. La ar-quitectura de un sistema es una entidad demasiado compleja como para ser detallada en un formato de “dimensión única”.

Las vistas representan diferentes metas y usos de la documentación y exponen diferen-tes atributos de calidad a distintos niveles dentro del sistema.

**Vista de Módulo:** Describen como el sistema es estructurado en un conjunto de unida-des de código.

**Vista de Componentes y Conectores:** Describen como el sistema es estructurado como un conjunto de elementos que interactúan en tiempo de ejecución.

**Vista de Asignación / Ubicación:** Describe como el sistema se relaciona con elementos no-software en su entorno. Es decir, como los componentes son asignados a los recursos físicos.

**Visual Studio:** Es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles.

**Web Forms:** Son páginas que los usuarios solicitan a través de su navegador y que forman la interfaz de usuario (UI) que da su apariencia a sus aplicaciones web. Estas páginas están escritas usando una combinación de HTML, controles de servidor y código de servidor. Cuando los usuarios solicitan una página, se compila y se ejecuta en el servidor y, a continuación, genera el código HTML que el navegador puede mostrar.

**WS-Reliable Messaging:** Describe un protocolo que permite SOAP de mensajes que se entregan de manera fiable entre las aplicaciones distribuidas en la presencia del componente de software, sistema o fallos de red.

1. LISTA DE FIGURAS
2. LISTA DE TABLAS
3. INTRODUCCION

El objetivo de este documento es presentar la Arquitectura de SilesWeb basada en tres conceptos: Lógico, Físico y conceptual, los cuales representan las diferentes vistas de Arquitectura. En la arquitectura conceptual se describe la estructura básica de la solución con énfasis en los conceptos y requerimientos. En la arquitectura lógica se describen los componentes lógicos del sistema, su estructura interna y relaciones. En la arquitectura física se describen las tecnologías que se deben usar de la plataforma escogida.

La arquitectura Lógica para SilesWeb está basada en la arquitectura estándar para aplicaciones multinivel definida por Microsoft y su grupo de Patterns and Practices2.

En la figura 1 se muestra el rol de cada uno de los componentes dentro del sistema de información que sigue esta arquitectura:

**Figura 1. Arquitectura estándar para aplicaciones multinivel**



A continuación, se explica el rol de cada uno de los componentes de la figura 1:

*Componentes de interfaz de usuario:* La mayor parte de las soluciones necesitan ofrecer al usuario un modo de interactuar con la aplicación. Las interfaces de usuario se implementan utilizando formas web (Ej ASP.NET), controles u otro tipo de tecnología que permita procesar y dar formato a los datos de los usuarios, así como adquirir y validar los datos entrantes procedentes de éstos. Son responsables de la captura, validación simple, visualización de la información.

*Componentes de proceso de interfaz:* En la mayoría de casos, la interacción del usuario con el sistema se realiza de acuerdo a un proceso predecible. Por ejemplo, en una aplicación comercial, se podría implementar un procedimiento que permita ver los datos del producto. De este modo, el usuario puede seleccionar de una lista de categorías de productos disponibles y , a continuación, elegir uno de los productos de la categoría seleccionada para ver los detalles correspondientes. Del mismo modo, cuando el usuario realiza una compra, la interacción sigue un proceso predecible de recolección de datos por parte del usuario, por el cual éste en primer lugar proporciona los detalles de los productos que se desea adquirir, a continuación, los detalles de pago y, por último, la información para el envío. Para facilitar la sincronización de la organización de las interacciones con el usuario, resulta útil utilizar componentes de proceso de interfaz de usuario individuales. De este modo, el flujo del proceso y la lógica de administración de estado no se incluye en el código de los componentes de interfaz de usuario, por lo que varias interfaces (Web, Windows, Movil, Etc) podrán utilizar el mismo “motor” de interacción básica.

*Flujos de Negocio:* Una vez que el proceso de interfaz ha recopilado los datos necesarios, éstos se pueden utilizar para ejecutar un proceso de negocios. Por ejemplo, tras enviar los detalles del producto, el pago y preparación del envió. Gran parte de los procesos de negocio conllevan acabo en un orden determinado. Por ejemplo el sistema empresarial necesita calcular el valor total del pedido, validar la información de la tarjeta de crédito, procesar el pago de la misma y preparar el envío del producto. El tiempo que este proceso puede tardar en completarse es indeterminado, por lo que sería preciso administrar las tareas necesarias, así como los datos requeridos para llevarlas a cabo.

*Componentes de negocio:* Independientemente de si el proceso de negocios consta de un único paso o de un flujo de trabajo organizado, la aplicación requeriría probablemente el uso de componentes que implementen reglas de negocio y realicen tareas de negocio. Por ejemplo, en una aplicación comercial, se deberá implementar una funcionalidad que calcule el precio total del pedido y agregue el costo adicional correspondiente por el envio del mismo. Los componentes de negocio implementan la lógica de negocio de la aplicación.

*Agente de servicios:* Cuando un componente de negocio requiere el uso de la funcionalidad proporcionada por un servicio externo, tal vez sea necesario hacer uso de componentes que administren la semántica de la comunicación con dicho servicio. Por ejemplo, el componente de negocio de la aplicación comercial descrita anteriormente podría utilizar un agente de servicios para administrar la comunicación con el servicio de autorización de las tarjetas de crédito y utilizar un segundo agente de servicios para controlar las conversaciones con el servicio de mensajería. Los agentes de servicios permiten aislar las particularidades de las llamadas de varios servicios desde aplicación y pueden proporcionar servicios adicionales, como el mapeo del formato de los datos que expone el servicio al formato que quiere la aplicación.

*Interfaces de servicios*: Para exponer lógica de negocios como un servicio, es necesario crear interfaces de servicios que sopórtenlos contratos de comunicación (comunicación basada en mensajes, formatos, protocolos, seguridad y excepciones, entre otros) que requieren los clientes. Por ejemplo, el servicio de autorización de tarjetas de crédito debe exponer una interfaz de servicios que describa la funcionalidad que ofrece el servicio, así como la semántica de comunicación requerida para llamar al mismo.

*Componente de acceso a datos*: La mayoría de las aplicaciones y servicios necesitan obtener acceso al repositorio de datos en un momento determinado del proceso de negocios, Por ejemplo, la aplicación necesita recuperar los datos de los productos de una base de datos para mostrar al usuario los detalles de los mismos, así como insertar dicha información en la base de datos cuando un usuario realiza un pedido. Por lo tanto, es razonable abstraer la lógica necesaria para obtener acceso a los datos (y la estructura como están almacenados)en una capa independiente de componentes de acceso a datos, ya que de este modo se centraliza la funcionalidad de acceso a datos y se facilita la configuración y mantenimiento de la misma.

*Entidades de negocio:* La mayoría de las aplicaciones requieren el paso de datos entre distintos componentes. Por ejemplo, en una aplicación comercial es necesario pasar una lista de productos de los componentes de acceso a datos a los componentes de interfaz de usuario para que este pueda visualizar dicha lista. Los datos se utilizan para representar entidades de negocio del mundo real, como productos o pedidos. Las entidades de negocio que se utilizan de forma interna en la aplicación suelen ser estructuras de datos, DataSets de ADO.NET, DataReader o secuencias XML, aunque también se puede implementar utilizando clases personalizadas (patrón DTO) que representa entidades del mundo real necesarias para la aplicación, como productos, pedidos, o clientes.

*Verticales de seguridad, administración operacional y comunicaciones:* La aplicación probablemente utilice también componentes para realizar la administración de excepciones, autorizar a los usuarios a que realicen tareas determinadas y comunicarse con otros servicios y aplicaciones. Aunque por su naturaleza de vertical influyen en cada una de las capas de la arquitectura, en la vista lógica y física de la arquitectura, Existen tres componentes de la capa de negocio que facilitan la implementación de cada una de estas verticales en esta capa, que es donde mayor impacto y uso tienen.

Cada servicio funcional y no-funcional de los identificados en la arquitectura tiene su propia arquitectura de 3 capas. Cada uno en estas tres capas, pueden incluir (según sus requerimientos) algunos de los componentes estándares incluidos en el diagrama anterior. Teniendo en cuenta esto. La figura 2 muestra el diagrama general de la arquitectura para SilesWeb.

**Figura 2. Arquitectura general de SilesWeb**

1. OBJETIVO

Mostrar la arquitectura del SilesWeb en su división por vista para facilitar y mejorar el entendimiento del sistema de producción actual. A fin de que se puedan llevar seguimiento de órdenes de producción.

1. ALCANCE

Este documento junto con sus otras partes guía todo el diseño e implementación del sistema SilesWeb.

Está dirigido a las personas técnicas involucradas o que ingresen al proyecto para su entendimiento y fácil integración al mismo.

1. CAPITULO 1 – DESCOMPOSICIÓN DE SILESWEB

En este capítulo se presentan cada uno de los módulos de SILESWEB refinándolos a su vez en sub módulos con el objetivo de identificarlos, clasificarlos y definir responsabilidades y características de cada uno.

El estilo seleccionado para representar la vista de módulos es el de descomposición. Que permite representar las particiones del código en subsistemas, módulos, etc. Este estilo facilita las modificaciones puesto que la funcionalidad se especifica en lugares determinados.

Los criterios para descomponer un módulo en sub módulos dependen del propósito de la descomposición y está vinculado a la forma en la cual se alcanzan los atributos de calidad. Adicionalmente, en la sección que corresponde a las justificaciones de diseño se encuentra n documentadas, en detalle, las referencias a las decisiones tomadas.

Uso esperado del tipo de vista Módulo:

Construcción: Provee el bosquejo/referencia para el código fuente.

Análisis: Provee información para análisis de impacto

Comunicación: Explicar la funcionalidad del sistema a través de los diferentes niveles de granularidad empleados

No se empleará el tipo de vista seleccionada para:

* Realizar inferencias sobre comportamiento en tiempo de ejecución
* Realizar análisis de performance, confiabilidad, o cualquier otra característica propia de runtime(para ello emplear el tipo de vista componentes y conectores).

En la tabla 1 se observan las propiedades relevantes para cada uno de los módulos de SilesWeb (ver figura 3)

Tabla 1 Composición SilesWeb

|  |  |
| --- | --- |
| Nombre del Módulo | Responsabilidades |
| SilesWeb.Presentacion.Web | Representa la interfaz de usuarios de la nueva arquitectura |
| SilesWeb.Web\_XYZ | Representa la interfaz de usuario del o de los módulos que por decisión de arquitectura se desacoplen del SilesWeb |
| SilesWeb.ServicioWebExterno | Representa la interfaz de los servicios externos que ofrece SilesWeb |
| SilesWeb.Presentacion.Comun.Controladores | Contiene “proxies” a los servicios web de negocio. Provee acceso a los servicios de negocio desde los componentes de presentación e interfaz de usuario abstrayendo conceptos técnicos de comunicación, protocolos y formatos de mensajes |
| SilesWeb.Presentacion.Comun.Integrador | Contiene “proxies” a los servicios de negocio del macroproceso XYZ desacoplado que permiten la interacción con los servicios de negocio abstrayendo conceptos técnicos de comunicación protocolos y formatos de mensajes. |
| SilesWeb.Negocio.PREP.ServiciosWeb | Contiene “proxies” a los servicios web de negocio. Provee acceso a los servicios de negocio desde los componentes de presentación de interfaz de usuario abstrayendo conceptos técnicos de comunicación, protocolos y formatos de mensajes |
| SilesWeb.Negocio.ServiciosWeb | Abstrae aspectos técnicos de comunicación, protocolos, formatos, seguridad, entre otros separándolos de la implementación de los servicios funcionales.  Implementa el patrón “Service Interface”  Asignando un servicio web a cada macroproceso. |
| SilesWeb.Negocio.WS.Utils | Es parte de la capa de interfaz de servicios. Provee utilidades y extensiones a cada uno de los servicios web |
| SilesWeb.Negocio.REP.ServiciosWeb | Abstrae aspectos técnicos de comunicación, protocolos, formatos, seguridad, entre otros separándolos de la implementación de los servicios funcionales.  Implementa el patrón “Service Interface” asignando un único servicio web para operaciones de consulta |
| SilesWeb.Negocio.XYZ.Fachada | Proporciona un punto de acceso único a cada uno de los macroprocesos abstrayendo la complejidad de cada uno de los servicios de negocio (funcionalidades) |
| SilesWeb.Negocio.XYZ.ProcesosNegocio | Se encarga de orquestar y de manejar el flujo de actividades necesarias para llevar a cabo una funcionalidad. Ejecuta dichos flujos asegurando la transaccionalidad del proceso |
| SilesWeb.Negocio.XYZ.LogicaNegocio | Implementa la lógica de negocio, Sus responsabilidades incluyen validaciones funcionales y técnicas y la comunicación con los módulos de persistencia de datos y servicios externos. |
| SilesWeb.Negocio.XYZ.ReglasNegocio | Maneja determinadas validaciones de negocio de forma desacoplada de la lógica. Aportan flexibilidad para implementación de determinadas validaciones susceptibles de cambios. |
| SilesWeb.Comunes | Este módulo contiene los objetos de dominio compoartidos entre diferentes macroprocesos.  Proporciona un marco de base para la implementación de las reglas de negocio en cada uno de los macroprocesos.  Encapsula la lógica de activación de las instancias de las fachadas de negocio para cada macroproceso.  Prporciona un marco de base para los componentes de presistencia utilizando el “Microsoft Data Access Aplication Block”. |
| SilesWeb.Negocio.Comun.Negocio | Implementa la lógica de negocio común. Sus responsabilidades incluyen validaciones funcionales y técnicas y la comunicación con los módulos de presistencia de datos y servicios externos.  Se encarga de orquestar y de manejar el flujo común de las actividades necesarias para llevar a cabouna funcionalidad. Ejecuta dichos flujos asegurando la transaccionalidad del proceso.  Maneja determinadas validacxiones comunes al negocio de forma desacoplada de la lógica. Aportan flexibilidad para la implementación de determinadas validaciones susceptibles de cambios |
| SilesWeb.Negocio.Comun.Servicio | Interceptores: Representa el marco de base para la implementación de aspectos que interceptan las operaciones de negocio agregando responsabilidades a las mismas de forma dinámica  Incluye aspectos concretos de intercepción de operaciones  Incluye los atributos personalizados(“custom attributes”) que reciben los datos básicos de cada una de las operaciones de negocio a ser interceptadas.  Interfaces: Contiene las interfaces para cada uno de los macroprocesos. Estas interfaces son realizadas dentro del módulo.  SilesWeb.Negocio.XYZ.Fachada  Mapeadores: Encapsila la lógica de manejo de metadatos de transacciones.Provee la funcionalidad de mapeos entra los atributos de las transacciones y objetos de negocio de la aplicación  Seguridad: Implementa de forma centralizada aspectos de seguridad tales como autenticación y autorización a transacciones y sirve de apoyo para la validación de requerimientos funcionales como las políticas de restricción o la disponibilidad de las transacciones. |
| SilesWeb.Negocio.Comun.Servicios.  FuentesExternas | Abstrae aspectos técnicos de comunicación con servicios o fuentes externas al SilesWeb.Representa un “proxy” a estos. |
| SilesWeb.Datos.Servicis.XYZ | Interfaces: contiene las interfaces de los componentes de persistencia para un macroproceso dado ocultando estrategias concretas de implementación.  Acceso a Datos: Implementan interfaces definidas en modulo anterior utilizando una fuente de datos en particular. |

Figura 3 SilesWeb Módulos

* 1. Vista de Modelos (Distribución en capas lógicas)

Esta vista tiene como propósito presentar una separación del software en unidades más pequeñas. En este caso las unidades son capas, cada una de las cuales representan una máquina virtual. Cada una de estas máquinas virtuales proporciona un conjunto cohesivo de servicios beneficiando los atributos de portabilidad y de modificación a funcionalidades del sistema.

Esta vista pretende identificar esas capas para un conjunto de módulos del sistema definiendo a su vez las características y relaciones que las gobiernan.

La figura 4 muestra la descomposición en capas para un conjunto de módulos detallados anteriormente. Esta descomposición es común y genérica, y se aplica a los siguientes módulos.

* Módulos de servicio de macroprocesos
* Servicios Generales de Negocio
* Servicios de Administración del sistema
* Servicios de Configuración de usuarios, perfiles y privilegios

Los recuadros representan una capa de un módulo particular. Los bordes adyacentes de los recuadros representan la relación “tiene permitido utilizar” entre las capas correspondientes en la dirección desde abajo hacia arriba. Por ejemplo, la capa de fachada de negocio “tiene permitido utilizar” las capas de proceso de negocio, la de lógica de negocio y la de entidades de negocio. Sin embargo la capa lógica de negocio NO “tiene permitido utilizar” ni la capa de procesos de negocio – ya que la relación se lee desde bajo hacia arriba y no al revés- ni la de interfaz de servicio – ya que no tiene bordes adyacentes con dicha capa-.

En algunos módulos las capas de interfaz de usuario o procesos de interfaz de usuario puede que estén ausentes al no requerir servicios de presentación ya que son módulos o servicios de soporte a otros.

Es preciso aclarar además que las relaciones aquí mostradas son validas dentro de un único módulo fijo; o, dicho de otro modo, no se muestran aquí relaciones entre las distintas capas de dos o más módulos diferentes. Del mismo modo no se muestran tampoco las relaciones entre estas capas y otros módulos definidos como los servicios de soporte al negocio.

Figura 4 Descomposición de capas lógicas – SilesWeb

La tabla 2, provee información de identificación y responsabilidades de los elementos de la vista. La visibilidad de los mismos se determina a partir de las relaciones identificadas en el siguiente apartado.

|  |  |
| --- | --- |
| Nombre | Responsabilidades |
| Interfaz de usuario | Esta capa es la encargada de presentar la información al usuario final y proporcionar mecanismos para ingreso de información e interacción entre el sistema y el usuario haciendo uso de componentes gráficos, controles y formas. |
| Procesos de interfaz de usuario | Esta capa tiene como responsabilidad proporcionar servicios a la capa de interfaz de usuario abstrayendo una serie de aspectos comunes como es la navegabilidad, manejo de estado, configuración, acceso a servicios de negocio, entre otros con el objetivo de aumentar la reutilización, modificación y portabilidad de la capa física de presentación de la aplicación |
| Interfaz de servicio | La capa interfaz de servicio abstrae a su vez aspectos técnicos de comunicación, protocolos, formato de mensajes, latencia, entre otros con el objetivo de separar estos aspectos técnicos de la implementación de los servicios funcionales mejorando así la portabilidad de la aplicación.  La presencia de esta capa determina que un futuro cambio en alguno de los aspectos técnicos mencionados anteriormente tenga un impacto mínimo en los aspectos funcionales del sistema. |
| Fachada de negocio | Proporciona un único punto de acceso por macroproceso a un servicio determinado abstrayendo la complejidad de la lógica de negocio de dicho servicio. |
| Procesos de negocio | Esta capa se encarga de orquestar y de manejar el flujo de actividades necesarias para llevar a cabo una funcionalidad.  En muchos casos una determinada funcionalidad o proceso puede ser descompuesta/o en actividades de mayor granularidad. Estas actividades son controladas por un componente dentro de la capa de procesos de negocio garantizando la transaccionabilidad y consistencia del proceso.  Un ejemplo de esto puede ser la creación de un clasificador y sus niveles en el sistema. La creación de cada uno de los niveles en el sistema. La creación de cada uno de los niveles cuenta con su propia lógica y reglas de negocio y luego la creación del clasificador en si, cuenta también con su propia lógica y reglas de negocio. Un “Proceso de negocio” Orquesta y garantiza la transaccionabilidad de cada una de esas actividades. Existen otros casos como los terceros, que el tratarse de una entidad de negocio muy grande, se descompone la responsabilidad de la creación en multiples actividades (información básica, información tributaria, representantes legales, etc), y nuevamente existe un “Proeceso de Negocio” que organiza cada una de esas actividades. |
| Reglas de negocio | Las reglas de negocio del sistema se ubicaran o bien como parte de la lógica de negocio, o dentro de componentes aislados para este propósito.  Esta capa es responsable de manejar el ultimo escenario, brindando acceso a los componentes que ejecutan reglas de negocio de manera desacoplada con la lógica. |
| Entidades de negocio | Esta compuesto por las entidades de negocio del sistema. Además hace uso del patrón de diseño “Data Transfer Object”(DTO) representando entidades reales del negocio como clases de C#  Las entidades de negocio albergan los datos pasados entre los módulos o componentes del sistema evitando así el uso de parámetros dispersos que atentan contra la mantenibilidad del sistema. |
| Acceso a fuentes de datos | Esta capa se encarga de abstraer la lógica necesaria para el acceso a un repositorio especifico de datos o un sistema de información externo |

El documento “Designing Data Tier Components and passing data through Tiers del grupo de practicas y patrones de Microsoft muestra diversas formas de implementación para las entidades de negocio de un sistema con sus ventajas y desventajas.

De las opciones presentadas, el “DataSet – Tuped o no –“ y las “Clases propias” son generalmente las mas utilizadas y ambas presentan argumentos muy validos para la elección de una u otra estrategia por lo que típicamente la decisión se basa en las preferencias del desarrollador o arquitecto. En particular, los dataset son muy buenas herramientas para el manejo de colecciones de datos, vinculación y para representar con objetos un modelo relacional imitando el modelo de datos del sistema. Por otro lado, las clases propias presentan mejoras en cuanto al rendimiento de serializaccion y ventajas inhererentes de la orientación de objetos.

Sin embargo, el lenguaje Microsoft .Net C# incorpora una serie de mejoras y carqacteristicas nuevas que enriquece el espectro de las clases propias. En particular, características como los “generics”, “predicados” y “delegados anónimos” hacen que el manejo de colecciones (tipadas) de objetos y sus respectivas búsquedas sea sencillas y seguras.

A partir de todo esto, y teniendo en cuenta que atributos como rendimiento han sido privilegiados en esta arquitectura, se tomó la decisión de utilizar objetos de C# como entidades de negocio y no esquemas o datasets.

* 1. Vista de Módulos (Usos y dependencias entre módulos)

El objetivo de esta vista es presentar las relaciones de dependencias entre los módulos y capas lógicas mostradas anteriormente. Esta vista permite identificar precondiciones o requerimientos para el desarrollo de un modulo en particular, asi como el desarrollo incremental de los mismos.

Los elementos, sus propiedades, interfaces y comportamiento de estos elementos serán detallados en el tema: Descomposición del Servicio de Administración del Procesamiento.

El elemento ServicioXYZ puede ser reemplazado por los módulos de servicios de macroprocesos, Servicios de administración del sistema o servicios generales de negocio.

En cuanto a las relaciones, el propósito de esta vista (figura 5) es mostrar tipos de relaciones de dependencia, en este caso en particular el tipo de relación es la de “usa”.

Figura 5 Usos y dependencias entre módulos

* 1. Vista Componentes y conectores

El propósito de esta vista es presentar aquellos elementos del sistema que tengan presencia en tiempo de ejecución (por ejemplo: un cliente web). Adicionalmente se incluyen en los modelos los caminos de comunicación y sus detalles.

Los elementos de esta vista son los componentes y los conectores, Cada una de estos elementos tiene una manifestación en tiempo de ejecución, y contribuyente al comportamiento de ejecución de sistemas.

La figuro 6, muestra las relaciones que se presentan y asocian los componentes con conectores formando un grafo que representa una configuración del sistema en tiempo de ejecución.

Figura 6. Componentes y conectores SilesWeb

La vista de componentes y conectores se presenta el estilo “cliente-servidor”, la cual enfatiza el desacoplamiento entre los clientes de los servicios que estos consumen; adicionalmente enfatiza la división de funcionalidades entre clientes y servidores, lo cual habilita la asignación independiente hacia capas físicas dando soporte para la performance, escalabilidad y confiabilidad.

En este tipo de vista la comunicación se da típicamente de a pares e iniciada por un componente cliente que envía peticiones a otro componente servidor. Este tipo de interacciones son propias de una aplicación Web distribuida en aspectos de presentación, Lógica de negocio y acceso a datos tal como es el caso de Siles.

Uso esperado de la vista de componentes y Conectores

El tipo de vista la comunicación se da típicamente de a pares es iniciada por un componente cliente que envía petiiones a otro