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 de componentes y conectores es utilizado para inferir atributos de calidad del sistema en tiempo de ejecución, tales como rendimiento, confiabilidad y disponibilidad.

Este tipo de vista permite responder a preguntas como:

¿Cuáles son los principales componentes de ejecución del sistema y como se relacionan?

¿Cuáles son los principales repositorios de datos?

¿Cuál es el flujo que siguen los datos en el sistema a medida que éste ejecuta?

¿Qué protocolos de comunicación son utilizados por las diferentes entidades?

Por otro lado, este tipo de vista no es adecuado para representar elementos de diseño que no tienen presencia en tiempo de ejecución.

Los elementos (componentes) y sus responsabilidad coinciden con los módulos análogos presentados anteriormente. Es importante notar que en esta vista se hace énfasis en las relaciones entre dichos componentes (conectores). Dichos conectores y sus propiedades son presentados en la figura 6.

A continuación, se describen las justificaciones relevantes a tener en cuenta para las decisiones de diseño arquitectónico:

**Presentación:**

* En función de favorecer la mantenibilidad, los componentes de presentación se constituyen como una única aplicación web.
* Dos aplicaciones web separadas agregar complejidad innecesaria frente a atributos de calidad como mantenibilidad, escalabilidad y seguridad, todos atributos relevantes para el SilesWeb.
* Cache a nivel de presentación: Las decisiones tomadas en este punto se apoyan en las recomendaciones de las guias del grupo de practicas y patrones de Microsoft. En particular se hace uso extensivo de “Fragment Caching” (controles de usuario y master pages) y “Output Caching”

**Negocio:**

* Siguiendo el principio de segmentación de interfaces, que indica que no se debe exponer componentes clientes, servicios o métodos que no necesita conocer; se opta porque cada macroproceso cuente con su propia Interface Patters para los servicios y Facade Pattern para los componentes de negocio.

**Servicio de Opciones del Sistema:**

* Se implementa en los macroprocesos de seguridad (SEG) y administración (ADM).
* Servicios de Cátalogos: Se optp por la implementación decentralizada de cada uno de los catálogos para facilitar la implementación de diversos Macroprocesos y obedeciendo el orden consolidado de casos de uso definidos para el proyecto.

A su vez, si bien existen muchos puntos en común entre las diferentes instancias de los catálogos, también existe diferencias importantes entre ellos lo que hace válida una implementación caso a caso con algunas restricciones.

Cabe notar que la implementación de cada uno de los catálogos mantiene la uniformidad y consistencia y esto se asegura a partir de siguientes practicas:

* Desarrollo de componentes de interfaz de usuario comunes (controles) para cada uno de los catálogos.
* Diseño de clases respetando una jerarquía predefinida que garantice la mantenibilidad y consistencia
* Diseño de esquema de base de datos uniforme para cada uno de los catálogos.
* Servicios Transaccionales: Las responsabilidades asociadas a los servicios transaccionales (administración de transacciones) son implementadas completamente deltro del macroproceso ADM.

Los requerimientos funcionales para el macroproceso de ADM dirigen su implementación, asociándole responsabilidades de:

* Definición y administración de transacciones
* Creación de categorías de transacciones
* Configuración de transacciones: políticas de restricción, disponibilidad y horario, características de seguridad, manejo de datos administrativos y consecutivos.

Por su parte, el documento Siles abstrae unidades de información creadas y modificadas dentro de distintas transacciones (Documentos) y provee información metadatos y capacidades para la obtención de esas estructuras. Estas estructuras son utilizadas constantemente en el proceso de contabilidad. A su vez, conceptos propios de los servicios transaccionales como datos administrativos y consecutivos son también implementados dentro del macroproceso de ADM.

* 1. Vistas de Asignación (Implementación de módulos)

El objetivo de esta vista es mostrar la infraestructura de desarrollo de la aplicación en relación con los módulos vistos anteriormente. Se mostraran también los distintos proyectos de Visual Studio seleccionados para la implementación de cada uno de esos módulos y sus relaciones.

La tabla 3, presenta la información de la vista de modo no muy extenso, pare ello se utiliza la denominación [Macroproceso] que debería ser reemplazada por cada uno de los módulos funcionales del sistema a saber, ADM , ADQ, APR, CNT, EPG, GPR, ING, MHCREP, PAC, PAG,PRG, REC y SEG.

**Tabla 3. Infraestructura de desarrollo de la aplicación arquitectura CORE**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Responsabilidades** | **Detalles de implementación** |
| SilesWeb.Datos.Servicios.  [Macroprocesos] | Interfaces: contiene las interfaces de C# para los distintos objetivos de acceso a datos  AccesoDatos: Implementaci las inbterfaces del proyecto anterior (interfaces) y provee acceso a datos SilesWeb | Biblioteca de clases |
| SilesWeb.Comunes | Contiene clases bases y de configuración comunes a los distintos componentes de lógica de negocio.  Contiene clases comunes a todos los componentes de acceso a datos como ser clases bases, fábricas o clases para la comunicación con el Enterprise Library. Implementa Servicio de Acceso a Fuentes de Datos  Contiene entidades de negocio que son comunes a distintos macroprocesos. El propósito aquí es evitar referencias circulares entre ellos.  Este proyecto tiene un conjunto de clases (DTO) cuyo objetivo es agrupar información(no necesariamente relacionada en la realidad) para servir a la capa de presentación de modo de evitar round-trips. | Se implementa como una biblioteca de clases que es referenciada por los distintos componentes. |
| SilesWeb.Negocio.Comun.  Servicios | Este proyecto contiene clases de configuración y activacón (patrón Factory) que permite obtener referencias a cada una de las fachadas de los servicios.  Centraliza la creación de estas fachadas de los servicios.  Centraliza la creación de estas fachadas. | Biblioteca de clases  Las clases aquí presentes toman de un archivo de configuración el tipo de la fachada de un determinado servicio y luego utilizan “Reflection” para crear estos tipos. Eso evita las referencias circulares entre los proyectos |
| SilesWeb.Negocio.Comun.  Negocio | ReglasNegocio: En este proyecto se implementan las reglas de negocio comunes al sistema.  LogicaNegocio: Aquí se implementa la lógica de negocio común al sistema  Proceso de Negocio: Implementa otra de las capas lógicas identificada. En este caso la capa lógica de procesos de negocio común al sistema. | Biblioteca de clases.  Referencia al proyecto común de reglas de negocio |
| SilesWeb.Negocio.Comun.  Servicios | Interfaces: Contiene las interfaces que implementan cada una de las fachadas de los servicios.  Este proyecto contiene clases de configuración y activación (patrón Factory) que permiten obtener referencias a cada una de las fachadas de los servicios.  Centraliza la creación de estas fachadas. | Biblioteca de clases |
| SilesWeb.Negocio.Comun.  Servicios.FuentesExternas | Este proyecto contiene las clases que actúan como proxy a las distintas fuentes externas a las que el Siles Accede.  Se comunica con el servicio de integración para finalmente interactuar con la fuente externa. | Biblioteca de clases |
| SilesWeb.Negocio.  [Macroproceso].  ReglasNegocio | En este proyecto se implementan las reglas de negocio particulares del macroproceso | Biblioteca de clases.  Referencia a las entidades de negocio, componentes de acceso a datos y reglas de negocio. |
| SilesWeb.Negocio.  [Macroproceso].  LogicaNegocio | Aqui se implementa la lógica de negocio para un macroproceso dado. | Biblioteca de clases.  Referencia a la lógica de negocio, componentes de acceso a datos y reglas de negocio. |
| SilesWeb.Negocio.  [Macroproceso].  ProcesosNegocio | Implementa otra de las capas lógicas identificada. En este caso la capa lógica de procesos de negocio para un macroproceso dado | Biblioteca de clases  Referencia al proyecto de lógica de negocio. |
| SilesWeb. Negocio.  [Macroproceso].  FachadaNegocio | Como en el caso anterior este proyecto implementa la siguiente capa lógica de un macroproceso.  Contiene una clase única que sirve como punto de entrada al macroproceso | Biblioteca de clases |
| SilesWeb.Negocio.Servicios  Web | Representa la interfaz de servicio de la capa de negocio del sistema.  Contiene servicios web para cada macroproceso.  Ademas es un proyecto web en cuyo contexto de ejecución se ejecutan todos los proyectos mencionados anteriormente de modo de ser un punto central en cuanto a configuración de parámetros, seguridad y ejecución. | Proyecto de servicios web ASP.NET |
| SilesWeb. Negocio.WS.  Utils | Presenta algunas clases de utilidad o apoyo a los servicios web mencionados. | Biblioteca de clases |
| SilesWeb.Presentacion.  Comun.Controladores | Contiene los “controladores” que son clases que abstraen el acceso a la lógica de negocio.  Permite que los distintos servicios de presentación tengan un mecanismo uniforme para el acceso a la capa de negocio. | Biblioteca de clases  Contiene las referencias a los servicios web de la capa de negocio. |
| SilesWeb.WebApplication | Este proyecto web implementa los servicios de presentación y de la aplicación | Proyecto de sitio web de ASP.NET |
| SilesWeb. Logging | Contiene una serie de clases que exienden a las del “Loging Aplication Block” de modo de tener un formato estándar para la aplicación en cuanto al registro. | Biblioteca de clases |
| SilesWeb.Cache | Contiene las clases que permiten configurar el Servicio de Caching. | Biblioteca de clases |
| SilesWeb.Cache.  BaseDatos | Implementa las clases que extiende el Application Block de acceso a datos de modo de agregarles el aspecto de caching | Biblioteca de clases |
| SilesWeb.Cache.Nodo | Contiene las clases que abstraen el acceso al Application Block de caching | Biblioteca de clases |
| SilesWeb.Cache.Util | Clases de utilidad para servicio de Caching | Biblioteca de clases |

**Relaciones y sus propiedades**

El tipo de relación que se presenta en esta vista es el de “contiene” o “esta asignado a” que se da entre las distintas carpetas, proyectos, ítems de configuración de software y el sistema de archivos de sistema operativo.

En la figura 7 presenta nuevamente la organización de la solución, proyectos de Visual Studio y namespaces y muestra de forma esquemática la relación entre estos proyectos (fechas).

A nivel de capa de presentación, se puede ver una única aplicación web ASP.NET que integra cada uno de los servicios y que internamente se subdivide en los servicios de macroproceos, administración, seguridad. También contiene una serie de controles web comunes a los módulos mencionados anteriormente y que son considerados dentro de los servicios generales de negocio, como es el caso de los servicios de datos administrativos. Por ultimo, dentro de presentación existe un proyecto que contiene los controladores que actúan como proxy a los servicios de negocio. Esto beneficia la portabilidad del aplicativo, de modo que un cambio por ejemplo en los protocolo de comunicación entre las capas físicas de presentación y negocio afecta solamente este proyecto.

Por otro lado, permite que otros clientes .NET accedan a los servicios de negocio al actuar como proxy a los mismos.

A nivel de capa de negocio, se observan tres grandes bloques, por un lado, la implementación de la interfaz de servicios, dentro de la carpeta Servicios Web, expone las funcionalidades de negocio como servicios web XML. Por otro lado, dentro de la carpeta común, residen una serie de bibliotecas de clases (DLLs) cuyo propósito es proveer funcionalidades comunes a cada uno de los proyectos de negocio, de modo que implementan muchos de los servicios generales del negocios y servicios de apoyo al negocio como ser el servicio de reglas de negocio, el módulo de información de transacciones, el documento Siles y el servicio de datos administrativos. Finalmente, dentro de la carpeta de servicios, se encuentran los módulos funcionales de negocio, los servicios de macroprocesos, el servicio de administración del sistema y el módulo de configuración de usuarios, perfiles y privilegios; cada uno de los cuales presentan una descomposición uniforme en capas.

Por último, en la capa de datos, se implementan las clases de base representan los servicios de acceso a fuentes de datos. También incluye una carpeta de servicios, en donde reside la lógica de acceso a los datos que se corresponde con los servicios de la capa de negocio.

A nivel general, se pueden visualizar las relaciones permitidas entre los proyectos y la dirección de las mismas mediante las fechas. Las mismas se implementan simplemente como referencias entre los proyectos de visual Studio 2022 en adelante.

**Figuras 7 Infraestructura de desarrollo de la aplicación Arquitectura CORE**

1. Capitulo 2 –Reportes SilesWeb

En este capítulo se presenta los reportes de SilesWeb, refinándolos a su vez en submódulos con el objetivo de identificarlos, clasificarlos y definir responsabilidades y características de cada uno.

Basados en los requerimientos no funcionales del sistema, el sitio web de reportes fue diseñado para permitir al usuario ejecutar todos los reportes definidos en SilesWeb por medio de formularios de filtros que deben estar acoplados a silesWeb, por lo que es necesario alinearse a la arquitectura SilesWeb para su creación. Este lineamiento debe ser a nivel modular y transversal.

Por otro lado, SQL server reporting Services nos permite crear, publicar y administrar subscripciones e informes con datos extraidos de diferentes orígenes, estos informes se pueden visualizar y administrar mediante una conexión web, además puede ser exportados a diversis formatos (ver tabla 4); nosotros hacemos uso de muchas de las características de reporting Services para lograr la integración de los reportes al sistema SilesWeb (ver figura 8), como es especificado en los requerimientos no funcionales.

Algunas de las caracteristcas de reporting services nos permite solucionar requerimientos como la administración, disponibilidad y portabilidad, necesidad de extender o personalizar sus componentes, a continuación, listamos los componentes principales de SQL Server Reporting Services:

* Un conjunto completo de herramientas que se pueden utilizar para vrear, administrar y ver informes
* Un componente servidor de informes que aloja y procesa reportes en diversos formatos
* Componentes para administrar la seguridad, recepción de datos de diferentes orígenes de datos, exportación a diferentes formatos y programación y envio de reportes.
* Componentes para integrar y extender características de reporting services como seguridad, recepción de data, formatos de renderización y programación de envió de reportes

**Tabla 4. Opciones de exportar reportes en Reporting Services**

|  |  |
| --- | --- |
| **Opción de Exportación** | **Descripción** |
| PDF | PDF representa un mapa del documento como el papel de marcadores. Todos los elementos del mapa del documento se enumeran uno detrás de otro en el panel. Incluye una jerarquía de vínculos. Si se especifica un intervalo de páginas, solo aparecen en la jerarquía los marcadores representados. |
| Microsoft Excel | Excel representa un mapa del documento como la primera hoja de cálculo del libro. Incluye una jerarquía de vínculos. Cuando se hace clic en el vínculo el mapa de documento, se abre la celda de destino adecuada en la hoja de cálculo correspondiente. |
| Microsoft Word | Word representa el mapa del documento como etiquetas de la tabla de contenido. |
| Otros (por ejemplo, TIFF, XML y CSV) | No disponible en MHTL, XML, CSV ni image |

Figura 8 Diagrama de contexto interacción SilesWeb – Reporting Services

En silesWeb, con el creciente aumento de almacenamiento de información, se hace necesario definir implementar algún tipo de escalabilidad (horizontal o vertical).La decisión de arquitectura es utilizar escalamiento horizontal(scale out, Ver figura 9) por que aprovecha el ahorro que supone distribuir la carga de procesamiento en más de un servidor. Aunque el escalado en horizontal se logra utilizando muchos equipos, la colección funciona esencialmente como un único equipo. Al dedicar varios equipos a una tarea común, mejora la tolerancia de errores de la aplicación. Por supuesto, desde el punto de vista del administrador, escalar en horizontal presenta un desafio mayor de administración debido al mayor numero de equipos

**Figura 9 Escalamiento Horizontal (Scale Out)**

Llevando lo definido anteriormente a Repórting Services, la figura 10, nos muestra la configuracion definida en modo de escalamiento Scale POut, en la cual varios servidores de informes comparten una sola base de datos del servidor de informes. La base de datos del servidor de informes se debe instalar en una instancia de SQL Server.

**Figura 10 Arquitectura con escalamiento Scale Out para reporting services**

**Vistas:**

Las vistas son una representación de un conjunto de elementos del sistema y de relaciones asociadas entre ellas. Al conjunto de vistas se le denomino ViewPoint, que son puntos de vista sobre intereses específicos de los stakeholders del sistema. Los estilos Arquitectonicos se utiliza para lograr determinados atributos de calidad, estos estilos reflejan soluciones recurrentes (patrones).

Existen una serie de View, ViewPoint y Styles que podemos utilizar para documentar nuestra arquitectura. Podemos clasificarlas las vistas de acuerdo a la información que trasmiten.

* 1. Vistas de Módulos – Descomposición de Reportes – SilesWeb

En la figura 11, se muestra como se utiliza el estilo “Descomposición”,que representa una petición lógica del código. Este estilo permite identificar la funcionalidad e importancia de los módulos dentro del sistema, por lo que es de mucho interés para mantenimiento del sistema.

**Figura 11 Vista de Módulo – Descomposicion de reportes – SilesWeb**

La tabla 5, describe cada uno de los elementos de esta vista (Figura 11) y sus respectivas propiedades:

**Tabla 5 Elementos propiedades vistas de modulo – descomposición de reportes**

|  |  |
| --- | --- |
| **Componente** | **Descripción** |
| Presentación | Contiene los componentes de preentacion de SilesWeb, entre los cuales están los formularios que contienen los filtros para ejecución de los reportes |
| Negocio | Contiene los componentes de lógica de negocio de SilesWeb, entre los cuales está la lógica usada por los formularios que contienen los filtros de los reportes |
| Datos | Contiene los componentes de acceso de datos de SilesWeb, entre los cuales están los componentes de acceso a datos usado por los formularios que contienen los filtros de los reportes |
| Reportes SilesWeb | Son los reportes de SilesWeb alojados en el servidor , estos serán organizados en carpetas por macroprocesos. |
| Seguridad | Extiende la seguridad de reporting services, usando la que se maneja en todo el SilesWeb, autenticación a través del macroproceso de SEG |
| Report Viewer | Componente utilizando por el cliente web para visualizar los reportes |
| Report Server | Contiene los componentes propíos del Report Server, los cuales sirven para administrar, publicar y enviar los reportes. |

* 1. **Vistas y componentes y conectores**

En esta vista ( ver figura 12), se usa el estilo “Cliente – Servidor”, este estilo es apropiado para mostrar sistemas distribuidos, por que enfatiza el desacoplamiento entre el cliente y los servicios que consume, también muestra la diferencia entre el cliente y servidores desde una perspectiva funcional, se caracteriza por tener un patrón de comunicación request – response, donde un cliente inicia la comunicación, enviando un requerimiento a un servidor.

**Figura 12 Vista de componentes y conectores**

La tabla 6, descripción cada uno de los elementos de esta vista y sus respectivas propiedades:

**Tabla 6 Elementos y propiedades vista de componentes y conectores**

|  |  |
| --- | --- |
| Componentes | Descripción |
| Report Viewer | Componente utilizado por el cliente web para visualizar los reportes |
| Presentación | Contiene los compoenentes de presentación de SilesWeb, entre los cuales están formularios que contienen los filtros de reporte, en este componentes se manejara las excepciones de los reportes a nivel de conexión. |
| Controlador | Provee al acceso a los servicios de negocio a través de los proxies de los servicios web, abstrae detalles técnicos en temas de comunicación, protocolos y formato de mensajes |
| Web Services | Procee una capa de abstracción de asuntos técnicos como, comunicación, protoclolos, formato de mensajes y seguridad |
| Fachada | Ofrece un único punto de acceso a los servicios funcionales de todos los macroprocesos, abstrayendo la complejidad de su implementación |
| DTO | Implementa el DTO Enterprice Pattern, que es usado como un contenedor de data para soportar intercambio heterogéneo de data, rediciendo las llamadas remotas |
| Negocio | Contiene los componentes de lógica de negocio de SilesWe, entre los cuales está la lógica usada por los formularios que contienen los filtros del reporte. |
| Datos | Contiene los componentes de acceso a datos de SilesWeb, entre los cuales están los componentes de acceso a datos usado por los formularios que contienen los filtros del reporte. |
| Report Server | Contiene los componentes propios del Report Server, los cuales sirven para administrar, publicar y enviar los reportes. |
| Seguridad | Extiende la seguridad de Reporting Services, usando la seguridad que se maneja en todo SilesWeb, autenticación a través del macroproceso de seguridad, adicionalmente se debe garantizar la autorización y la vinculación de perfiles de acuerdo con el Siles |
| Reportes SilesWeb | Contiene los reportes de Siles, almacenados como documentos XML |
| Datos SilesWeb | Contiene la data de Siles |

Al integrar el servicio de reportes con silesWeb, es imprescindible para el cliente que los criterios de búsqueda se hagan a través de la capa de presentaicon definida para los reportes en el sitio web: SilesWeb.WebApplicaction\_Reportes.

En la figura 13, se muestra esta iteración entre el usuario SilesWeb con reporting Services y Siles.

**Figura 13 Diagrama de secuencia de reporting Services – Siles Web**

* 1. **vista de asignación / Ubicación Reportes – SilesWeb**

El estilo seleccionado para esta vista es la de “Despliegue”. La figura 14, muestra la distribución física del hardware y el procedimiento a través de un conjunto de nodos en el sistema, tambien muestra los protocolos de comunicación que usan los nodos.

**Figura 14 Distribucion física de hardware y procesamiento Reporting Services – SilesWeb**

* 1. **Configuración inicial**

Para la carga inicial de reportes se gestionara el manejo de dos servidores de reporting Services, direccionados a una sola base de metadatos (Report Server)

El esquema de autenticación con Reporting Services, será sin manejo de sesión.Net del control ReportViewer, esto se realiza a través de la generación de un ensamblado que evita la comprobación de credenciales cada vez que hay peticiones del control a Reporting Services.

Los reportes son transacciones de SilesWeb, por lo que sobre ellos aplican todas las restricciones que está definidas para las transacciones del sistema. (Ej. Restricciones de horario, Perfil de usuario, Etc)

Como se especificó anteriormente, es necesario alinearse con la arquitectura de Siles, y sobre esta arquitectura agregar los componentes necesarios para integrar los reportes.

Se deben invocar los reportes desde SilesWeb utilizacndo el control de ReportViewer que permite embeber los reportes e integrarlos con aplicaciones Web desarrolladas en Microsoft. Net.

Si los formularios de filtro, los requiere se puede crear el camino de acceso hacia la base de datos, pasado a través de las capas físicas de SilesWeb y siguientdo los estándares definidos para la aplicación.

Finalmente, la ejecución de la transacción de reporte utiliza la URL del reporte para obtenerlo de Reporting Services y desplegarlo.

* 1. Decisiones de Arquitectura

Las excepciones de conexión del reporte se controlan a nivel de presentación, haciendo uso de las sentencias de Try adn catch.

Algunos reportes son consultados en tiempo real, por lo que es necesario que SilesWeb y los servicios de reportes compartan la misma base de datos.

Procedimientos almacenados (operaciones CRUD): Los procedimientos almacenados son una útil herramienta ya que proporcionan un mecanismo eficiente para el manejo (alta , baja , modificación , obtención) de la información de una base de datos. Sin embargo, la incorporación de excesiva lógica centro de los procedimeintos almacenados perjudica su mantenimiento y portabilidad del sistema. La implementación de lógica y/o reglas de negocio del sistema deberia ser implementado con una estrategia oriendata a objetos, pero no dentro de los procedimientos almacenados. A partir de esto, se definen los siguiente criterios para el uso y creación de procedimientos almacenados en el sistema:

Los procedimientos almacenados pueden incluir:

* Consultas complejas, incluyendo tablas multiples utilizando operadores JOIN y UNION, entre otros
* Multiples operaciones de consulta (SELECT) las cuales son pre compiladas por el motor de base de datos
* Uso de funciones predefinidas de SQL o T-SQL para simplificar la extracción o manipulación de datos
* Los procedimientos almacenados deben además cumplir con los estándares de nomenclatura

A lo anterior recomienda que se cuenten con las siguientes características:

* Index SEEK
* Cover Index
* Indexed view
* Join sobre columnas indexadas
* Índices filtrados
* Índices con Fill factor en 100 cuando los campos son de longitud fija
* Índices con ONLINE = ON
* Tablas con más de 20% de cambios en un dia: Avisar al DBA
* Verificar que cada índice tenga sus estadísticas
* No debe existir estadísticas sin índices
* Cada tabla con CLUSTER INDEX
* Todas las consultas sobre una tabla deben estar alineadas con el o los índices de la tabla
* Una tabla no debe tener índices que no se usan, o índices duplicados, o índices redundantes
* Usar sp\_executesql con los parámetros correctos
* Si se va a utilizar SQL DINAMICO y SP\_EXECUTESQL como mecanismo técnico, tenga en cuenta que SP\_EXECUTESQL tenga los siguientes argumentos:

EXECUTE sp\_executesql

@SQLString

[@ParmDefinition](mailto:.@ParmDefinition)

@CustomerID=@IntVariable

@SalerOrderOUT= SalesORderNumber OUTPUT;

Los procedimientos almacenados deben limitar el uso de:

* Manipulación de variables o cadenas de texto: Los procedimientos almacenados no deberían manipular explícitamente variables o campos obtenidos de la base de datos; por ejemplo, no debería extraer de un campo del tipo fecha, el día o año para tomar acciones en base a ello ya que eso lo debería hacer la capa de su presentación en una interfaz de usuario.
* Uso de (IF-THEN-ELSE) para el chequeo de condiciones multiples: Este tipo de sentencia deberia ser minimo; no se deberían tomar decisiones de negocio en los procedimientos almacenados en la base a estos chequeos
* Uso de cursores y operadores recursivos (WHILE): No deberían ser usados en general. Su uso debería ser limitado y las excepciones deben estar correctamente justificadas y documentadas.
* Uso de tablas temporales: Su uso deberia ser minimo. No se deben utilizar para implementar lógica de negocio o para iterar entre sus filas. Pueden ser utilizadas en escenarios de consultas complejas que no pueden ser implementadas con la sentencia única.
* Sentencias CASE: Estas sentencias se consideran al igual que los IF-THEN-ELSE; pueden ser utilizadas para implementar lógica de acceso a datos condicional.
* Sentencias dinámicas: por motivos de rendimiento deberían ser mínimas. Pueden ser usadas como último recurso en situaciones especiales que no pueden implementarse con sentencias normales como por ejemplo la implementación de búsquedas con comodines que dependen fuertemente de los parámetros de entrada.
* Uso del lenguaje T-SQL: En los casos en que SQL no provea las herramientas técnicas para una funcionalidad de acceso a datos en particular se podrá utilizar operaciones propias de T-SQL. En los casos en que SQL cubra los requerimientos necesarios se priorizara frente al uso de T-SQL. Se evitara el uso de T-SQL si esto implica perjuicios en el rendimiento del sistema.
* Uso de SQLCLR: Se limitara el uso de los procedimientos almacenados manejados por motivos de rendimiento. Sin embargo, esta es una herramienta muy poderosa que permite realizar tareas que serían imposibles o extremadamente complejas con T-SQL o SQL (manipulación de XML, por ejemplo). En esos casos se justifica el uso de SQLCLR.
* “Hard-coding” dentro de los procedimientos almacenados: No está permitido incluir código “quemado” dentro de los procedimientos almacenados. Ese tipo de información deberá ser pasada como parámetros a partir de valores de enumerados o constantes de C# desde las capas de negocio o acceso a datos.
* Calculos dentro de los procedimientos almacenados: Deberían ser minimos. Los cálculos de carácter financiedo deberían estar implementados como reglas de negocio. Cálculos simples utilizando funciones de agregación ( calculo de totales, por ejemplo) pueden implementarse en los procedimientos almacenados.
* Evitar uso de UNION en vez de UNION ALL en situaciones en que las consultas involucradas no retornan resultados en común y por lo tanto no se requiere el trabajo adicional que debe ejecutar SQL Server para destacar los duplicados.
* Definicion inadecuada de parámetros y variables en cuanto al tipo de tipos de datos. El tipo de datos de parámetros y variables utilizados en clausula Where no corresponden al tipo de datos de las columnas de las tablas por lo que los índices existen son subutilizados o ignorados por completo.
* Uso de UNION para unir resultados sobre objetos similares, causando que SQL realice IO y procesamiento redundante: Ejemplo:

SELECT \*

FROM T1

WHERE COL1=@var1

UNION

SELECT \*

FROM T1

WHERE COL 1=@var2

En vez de

SELECT \*

FROM T1

WHERE COL1=@var1 or COL1=@var2

* No realizar DROP TABLE de tablas temporales una vez dejan de ser utilizadas en un procedimiento
* Es posible que sea necesario crear índices en algunas tablas para mejorar el desempeño de algunos Stored Prodecures, sin embargo, la causa de los problemas se debe a código TSQL deficiente.
* Table Scan: Verifique y ajuste el COVER INDEX. Aplique las técnicas vistas y documentadas.
* Key LookUp:Restringir columnas en el SELECT y adicionar las restantes al COVER INDEZ existentes.
* Missing Index: Utilizar en la consulta los índices existentes.

Todos estos puntos citados representan los lineamientos de carácter general a ser seguidos por los desarrolladores para la creaciuon de procedimientos almacenados, los cuales garentizan buena portabilidad y mantenibilidad de la lógica de acceso a datos. Sin embargo, atributos tales como rendimiento pueden en algunos casos impedir que se cumplan cabalmente algunos de esos puntos; esos casos se justificaran y se documentaran la manera acorde indicando cuál de estas reglas no se estarían cumpliendo y porqué.

Aplicando los conceptos anteriormente expuestos. La figura 15 muestra el diseño general de la arquitectura conceptual para los reportes definidos por el SILES.

**Figura 15 Arquitectura Cenceptual Reportes SilesWeb**

3 CAPITULO 3 – SITIO WEB CARGA MASIVA

En este capitulo se presentan cargas de archivos de manera masiva para SilesWeb, en la tabla 7 se indican los tipos de archivo que son permitidos para realizar caras masivas en SilesWeb

Basados en los requerimientos no funcionales del sistema, el sitio web de cargas masivas fue diseñado para permitir al usuario ejecutar todas las cargas masivas definidas en SilesWeb por medio de formularios que deben estar acondicionados al SilesWeb para su creación.

Este lineamiento deber ser a nivel modular y transversal.

**Tabla 7 Tipos de archivos permitidos para cargas masivas**

|  |  |
| --- | --- |
| Opcion de Carga | Descripción |
| Pipe(|) | Son archivos texto, cuyos campos van separados por el símbolo (|) que se utilizan para cargar masivamente información de una transacción |
| Microsoft Excel | Hojas de cálculo de Microsoft Excel, utilizados únicamente para cargar grillas de presentación, utilizada únicamente en Nueva Arquitectura |
| XML | Archivos XML, que se utilizan para cargar masivamente información de una transacción |

3.1 Cargas masivas arquitectura Core

Las siguientes son las características principales que se manejan en las cargas masivas de la arquitectura Core

* Solo permite archivos texto separados por pipe (|)
* Solo recibe peticiones del transaccional
* Usa los mismos servicios web del transaccional
* Usa un control para permitir la carga de archivos
* Valida estructura usando archivos XSD
* Valida firmas
* Solo se puede manejar archivos con extensión .zip

La figura 16, muestra la arquitectura core de las cargas masivas de SilesWeb

**Figura 16. Arquitectura SilesWeb Core – Carga masiva**

Al integrar el servicio de cargas masivas con SilesWeb, es impresindible para el cliente que las cargas se hagan a travez de la capa de presentación definica en el sitio web : SilesWeb.WebApplication\_CargaMasiva

La figura 17, se muestra esta iteración entre el Usuario SilesWeb con las cargas masivas

**Figura 17. Diagrama de secuencia de Cargas Masivas – SilesWeb**

La tabla 8, describe cada uno de los elementos de este diagrama y sus respectivas propiedades:

**Tabla 8 Descripción componentes carga masiva arquitectura Core**

|  |  |
| --- | --- |
| Componente | Descripción |
| Presentación Carga Masiva | Contiene los componentes para ejecución de las cargas masivas, entre los cuales están los formularios que contienen los archivos a seleccionar, la validación de la firma digital de los archivos, el almacenamiento de la evidencia digital generada después de ejecutar el proceso de carga. |
| Negocio | Contiene los componentes de lógica de negocio de SilesWeb, entre los cuales esta la lógica usada por los formularios que contiene el sitio web de carga masiva. |
| Datos | Contiene los componentes de acceso a datos SilesWeb, entre los cuales esta los componentes de acces a datos usado por los formularios que contiene el sitio web de carga masiva |
| Seguridad y administración | Extiende la seguridad y administración de SilesWeb a la carga masiva, es decir permisos, uso de las transacciones de carga, se configuran en Siles |

3.2 Cargas masivas Nueva Arquitectura

Al ir evolucionando las herramientas de desarrollo. La arquitectura debe sufrir cambios que permitan utilizar y aprovechar de la mejor manera estas evoluciones, es el caso de las cargas masivas, que inicialmente fueron diseñadas para procesar, múltiples registros, pero de manera secuencial, ahora son las nuevas versiones tanto de herramientas de desarrollo como de motor de bases de datos, se hace posible enviar de manera masiva y en un solo bloque la totalidad de los registros, dando como resultado la optimización del procesamiento de los archivos, mejoras en el tiempo de respuesta al usuario, mejor uso de los artefactos de hardware.

El diseño presentado da la facilidad de reusar toda la lógica de negocio que ha sido implementada y probada sobre el sitio web. Como menciono en capítulos anteriores la Arquitectura SIlesWeb esta enfocada a servicios, lo que nos otorga mayor escalabilidad y flexiilidad en términos de configuracion e implementación del software sobre la infraestructura que lo soporte.

La figura 18, muestra la nueva arquitectura de las cargas masivas en SilesWeb, que a diferencia de la definida en la arquitectura Core, incluye las siguientes ventajas.

* Valida en un solo paso
* Genera Mensajes
* Realiza operación masiva de datos
* Genera respuesta via email

Figura 18 Nueva Arquitectura SilesWeb – Carga Masiva

La figura 19 muestra claramente el flujo ordenado de acciones que describe el comportamiento de la creación de una carga masiva bajo el nuevo esquema.

**Figura 19. Flujo creación carga masiva Nueva Arquitectura**

La tabla 9, describe cada uno de los elementos del flujo y sus respectivas propiedades:

**Tabla 9. Descripción elementos creación carga masiva en Nueva Arquitectura**

|  |  |
| --- | --- |
| **Nombre de Módulo** | **Responsabilidades** |
| Conocimiento del negocio | Revisar cuales transacciones involucradas hay, que restricciones, las validaciones, las tablas a afectar, los campos requeridos, etc. |
| Definición de requerimientos de la carga | Definir el formato de la carga (archivo plano o archivo en formato Excel), la cantidad de archivos y los campos necesarios para que la transacción funcione correctamente |
| Creación de esquemas (xsd) | Archivo que define el formato del archivo a cargar, se debe verificar si existe, se crea por transacción y se referencia otros archivos .xsd en caso de ser requeridos |
| Definición de creación de Types tipo tablas en la base de datos | Los campos de los .xsd y los types deben llamarse igual y ser del mismo tipo  Un type tipo tabla por archivo de carga |
| Implementación y utilización de servicios de carga masiva | Generar clases de carga EW\_\_\_\_.cs con los nombre de los SP´s(Inserción y Validación) y las “cajasde texto” para los archivos (presentación)  Incluir la carga en la fabrica |
| Definición de SP de validación | Devuelve tablas de parámetros de validación |
| Definición de esquemas de validación | Modificar el archivo xsd en la transacción correspondiente |
| Definición de SP de inserción | Devuelve resultados en tablas |
| Habilitación de la transacción | Modificar archivo configuracion MenuSistema.xml enn SilesWeb  Modificar archivo de configuración Web.Config en SilesWeb, definiendo como transacción Generics  En el sitio MVC, modificar el archivo de configuracion RouteSiteMap |

De acuerdo al diagrama de flujo expuesto, la secuencia de creación de una carga masiva utilizando el nuevo esquema es el siguiente:

* Creación de Archivos XSD Comunes: imagen
* Definición archivos xsd de la transacción imagen
* Definición elementos tipo Types: imagen
* Implementación y utilización de los servicios de la carga masiva:

Creacion de la clase EXCargaMasivaXYZ(XYZ nombre de la carga): imagen

Implementación en la fabrica:Imagen

Definición Store Procedure de Validación:Imagen

Definición de Store Procedure de Inserción: Imagen

Habilitación de la transacción:Imagen

**3.3 Decisiones de Arquitectura**

* La configuración inicial del sitio de presentación de carga masiva, será gestionado por una granja de servidores.
* Las excepciones a nivel de presentación, harán uso de las sentencias try and catch
* Las cargas masivas son ejecutadas en tiempo real, por lo que es necesario que SilesWeb y las cargas masivas compartan la misma base de datos.
* Las cargas masivas son transacciones de SilesWeb por lo que sobre ellas aplican todas las restricciones que están definidas para las transacciones del sistema(Ej.Restricciones de horario, perfil de usuario, etc)
* Por el volumen de información que se puede procesar, las cargas masivas deben tener restricciones de Horario
* Como se especifico anteriormente, es necesario alinearse con la arquitectura de SilesWeb, y sobre esta arquitectura agregar los componentes necesarios para integrar las cargas masivas
* No esta permitido utilizar componentes de terceros para invocar cargas masivas se debe hacer los desarrollos sobre Microsoft.Net
* Todas las cargas masivas deben estar implementadas en el sitio web de cargas asivas, no es valido que se encuentren en otro lugar (Ej, Sitio de presentación SilesWeb)
* Todos los archivos deben ir comprimidos y firmados digitalmente (minimo una firma).
* El usuario que carga el archivo debe tambien haber firmado el (los) archivo(s)
* Todos los archivos de carga deben ser almacenadois en el sitio de evidencia digital como expedientes digitales.(Ver arquitectura evidencia digital)
* La carga masiva debe funcionar de igual manera cómo funciona la transacción que el usuario utiliza cuando registra uno a uno
* El acceso al sitio de carga masiva se hace a través de SilesWeb, las validaciones de seguridad se realizan por cada carga masiva la cual se maneja como una transacción más del sistema.

Aplicando los conceptos anteriormente expuestos, la figura 20 muestra el diseño general del la arquitectura conceptual para cargas masivas definidas para SilesWeb.

Figura 20.ArquitecturaConceptual Cargas Masivas SilesWeb imagen

1. Capitulo 4 – Nueva Arquitectura SilesWeb

En este capítulo se presenta la nueva arquitectura para cada nuevo módulo de SilesWeb refinándolos a su vez en sub módulos con el objetivo 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.

* 1. Definición MVC

El patrón de arquitectura Modelo-vista-controlador (MVC) separa la aplicación en tres componentes principales: El modelo, la vista y el controlador (ver figura 21) . El marco ASP.NET MVC proporciona una alternativa al modelo de formularios web ASP.NET para crear aplicaciones Web, El marco ASP.NET MVC es un marco de presentación de peso ligero, altamente comprobable de que (al igual que con las aplicaciones basadas en Web Forms) se integra con las características de ASP.NET existentes, tales como las paginas maestras y autenticación basada en membresía. El marco MVC se define en conjuntoSystem.Web.Mvc.

**Figura 21 Patrón MVC**



MVC es un patrón de diseño estándar que muchos desarrolladores están familiarizados. Algunos tipos de aplicaciones Web se beneficiarán del marco MVC. Otros seguirán utilizando el patrón tradicional de aplicación ASp.NET que se basa en las solicitudes web y las devoluciones de datos. Otros tipos de aplicaciones Web se combinaran los dos enfoques; ni el planteamiento excluye a la otra.

El marco MVC incluye los siguientes componentes:

Modelo: los objetos del modelo son las partes de la aplicación que implementan la lógica de dominio de datos de la aplicación. A menudo, los objetos del modelo y recuperar el estado tienda de modelo en una base de datos. Por ejemplo, un objeto del producto podría recuperar información de una base de datos, operar en él, y luego escribir la información actualizada en la tabla productos en una base de datos SQL Server.

En aplicaciones pequeñas, el modelo es a menudo una separación conceptual en lugar de una física. Por ejemplo, si la aplicación solo lee un conjunto de datos y lo envía a la vista, la aplicación no tiene una capa del modelo físico y clases asociadas. En ese caso, el conjunto de datos asume el papel de un objeto modelo.

Vista. Las vistas son los componentes que muestran la interfaz de usuario de la aplicación (interfaz de usuario). Por lo general, esta interfaz de usuario se crea a partir de los datos del modelo. Un ejemplo sería una vista de edición de una tabla que muestra los productos cuadros de texto, listas desplegables y casillas de verificación basadas en el estado actual de un objeto o producto.

Controlador: Los controladores son los componentes que manejan la interacción con el usuario, el trabajo con el modelo, y finalmente seleccionar una vista para hacer que muestra la interfaz de usuario. En una aplicación MVC, la vista solo muestra la información; el controlador maneja y responde a la entrada del usuario y la interacción. Por ejemplo, el controlador trata los valores de cadena de consulta, y pasa estos valores en el modelo, que a su vez podría utilizar estos valores para consultar la base de datos.

El patron MVC ayuda a crear aplicaciones que separan los diferentes aspectos de la aplicación (lógica de entrada, la lógica de negocio, la lógica de interfaz de usuario), mientras que proporciona un acoplamiento débil entre estos elementos. El patrón especifica que cada tipo de lógica se debe colocar en la solicitud. La lógica de la interfaz de usuario pertenece en la vista. Lógica de entrada pertenece en el controlador. La lógica de negocio pertenece en el modelo. Esta separación ayuda a gestionar la complejidad cuando se genera una aplicación, ya que le permite centrarse en un aspecto de la aplicación a la vez.Por ejemplo, usted puede centrarse en la vista sin depender de la lógica de negocio.

El acoplamiento débil entre los tres componentes principales de una aplicación MVC también promueve el desarrollo paralelo. Por ejemplo, un desarrollador puede trabajar en vista, un segundo desarrollador puede trabajar en la lógica del controlador, y un tercero desarrollador puede concentrarse en la lógica de negocio en el modelo.

4..2 Ventajas de ASP.NET MVC

La ventaja que primero salta a la vista de ASP.NET es la facilidad con la que se generam URL semánticas, es decir URL que tengan la form <http://servidor/ver/productos/cafeteras> en lugar de <http://servidor/rpoductos/ver.aspx?code=cafeteras>. LasURLs semánticas se indexan mejor en los buscadores y son una practica SEO habitual. No es que en webforms no se puedan hacer, es que en ASP.NET MVC vienen de serie.

Otras ventajas, as a nivel técnico, son que con ASP.NET MVC se facilita mucho el probar nuestra aplicación (especialmente usando pruebas unitarias) y que el uso correcto del patrn MVC facilita la reutilización de código de manera mucho mas efectiva que en web forms.

Es posible tener diferentes vistas para un mismo modelo(ej. Representación de un conjunto de datos como una tabla o como un diagrama de barras)

Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.

Proporciona un mecanismo de configuración a componentes complejos mucho mas tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

Por supuesto, todo esto tiene un precio: la curva de aprendizaje de ASP.NET MVC puede ser mas alta que la de web forms, especialmente si nunca has desarrollado para web. A diferencia de web forms, que te abstrae de HTTP y HTML, ASP.Net MVC esta “mucho más cerca de la web”, lo que hace necesario conocer HTTP, HTML y Javascript para trabajar con él. De ltodos modos, eso no deberia echarte para atrás: si quieres crear aplicaciones web es normal que debas conocer los protocolos y lenguajes en los que se asentan la we, ¿no?

* 1. Flujo en una implementación general de MVC

Aunque se puede encontrar diferentes implementaciones de MVC, el flujo que sigue el control (ver figura 22) generalmente es el siguiente:

El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)

El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. EL controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra).

MVC tiene por objeto separar la lógica del negocio de las consideraciones de la interfaz de usuario para que los desarrolladores puedan modificar cada parte mas fácilmente sin afectar a la otra. En MVC el modelo representa la información (los datos) y las reglas de negocio: la vista contiene elementos de la interfaz de usuario como textos, formularios de entrada; y el controlador administra la comunicación entre la vista y el modelo.

Más allá del MVC, Allí también introduce un front-controller llamado aplicación el cual representa el contexto de ejecución del procesamiento del pedido. La aplicación resuelve el pedido del usuario y la dispara al controlador apropiado para tratamiento futuro.

Al final, la vista es quien lleva la responsabilidad de estableces la comunicación entre los elementos de nuestro patrón MVC.Cuando la vista recibe un mensaje que concierne al modelo o al controlador, lo deja registrado como el modelo con el cual se comunicara y apunta con la variable controller al controlador asignado, enviándole al mismo su identificación para que el controlador establezca en su variable view el identificador de la vista y asi quedan operar conjuntamente. EL responsable de deshacer estas conexiones, seguirá siendo la vista, quitándose a si misma como dependiente del modelo y liberando al controlador.

Figura 22: Flujo implementación general de MVC



* 1. Implementación en SilesWeb

En silesWeb los nuevos desarrollos utilizaran la nueva arquitectura, para tal fin se estableció un estándar que facilita la interacción entre dominios de negocio.

Los Dominios de Negocio (Macroprocesos) son módulos que integran la solución, están organizados mediante carpetas que hacen uso de las capas que forman el patrón MVC.

En la figura 23, el usuario realiza un Request (Solicitud), bajo protocolo HTTP, dicha solicitud se retoma mediante un POST o un GET al controlador dedicado para atender la transacción, el cual interactua con el modelo particular que tiene la vista(Cargado el Modelo), una vez cargado dicho modelo se retorna la solicitud hacia la vista mediante un response (Vista Modelo).

**Figura 23 Proceso Request bajo protocolo http**



La arquitectura Core de SilesWeb defina una plantilla o página maestra, sin embargo, para esta arquitectura se define un loyout (Ver figura 24), el cual presenta un problema en los archivos de cabecera y de pie que consiste en no tener código HTML valido, sin embargo, haciendo uso del patrón decorator resuelve el problema mencionado.



Por defecto todas las aplicaciones tiene es archivo llamado layout, en este directorio se guardan todas las plantillas globales de la aplicación.

4.5. Vista de Módulos (Distribución en capas lógicas)

Esta vista tiene como propósitos presentar una separación del software en unidades más pequeñas. En este caso las unidades son las 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 25 muestra la descomposición en capas de algunos de los modulos detallados anteriormente. Esta descomposición es común y genérica.

Los recuadros representan una capa de un módulo partícula. Los borde adyacente de los recuadros representan la relación “tiene permito utilizar” entre las capas correspondientes en la dirección desde abajo hacia arriba.

Es preciso aclarar además que las relaciones aquí mostradas son válidas dentro de un único modulo 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 25 Descomposición en capas lógicas – Nueva Arquitectura SilesWeb



4.6 Vista de asignación (Implementación de Módulos).

El objetivo de esta vista es mostrar la infraestructura de desarrollo de la aplicación en relación con los módulos vistos anteriormente. Se mostraran también los distintos proyectos de Visual Studio seleccionados para la implementación de cada uno de esos módulos y sus relaciones.

La tabla 10, presenta la información de la vista de modo no muy extenso, para ello utiliza la denominación [Macroproceso] que debería ser reemplazada por cada uno de los módulos funcionales del sistema.

**Relaciones y sus propiedades**

El tipo de relación que se presenta en esta vista es el de “contiene” o “está asignado a” que se da entre las distintas carpetas, proyectos, ítem de configuración de software y el sistema de archivos de sistema operativo.

En la figura 26 presenta nuevamente la organización de la solución, proyectos de visual Studio y namespaces y muestra de forma esquemática la relación entre estos proyectos (fecha).

A nivel general, se pueden visualizar las relaciones permitidas entre los proyectos y la dirección de las mismas mediante las flechas. Las mismas se implementan simplemente como referencia entre los proyectos de Visual Studio 2010 en adelante.

Tabla 10. Infraestructura de desarrollo de la aplicación en Nueva Arquitectura

|  |  |  |
| --- | --- | --- |
| Nombre | Responsabilidades | Detalles de implementación |
| SilesWeb.Datos.Arquitectura | Provee acceso a datos de la base de datos del SilesWeb para componentes transversales | Biblioteca de clases |
| SilesWeb.Negocio.Arquitectura | Contiene clases bases y de configuración transversales a los distintos compnentes de lógica de negocio.  Contiene clases comunes a todos los componentes de acceso a datos como ser clases bases y fabricas.  Contiene entidades de negocio que son trasversales al aplicativo.  Este proyecto tiene un conjunto de clases (DTO) cuyo objetico es agrupar información (no necesariamente relacionada en la realidad) para servir a la capa de presentación de modo de evitar round-trips | Se implementa como una biblioteca de clases que es referenciada por los distintos componentes |
| SilesWeb.Negocio.Arquitectura.Generics | Contiene clases bases y de configuración comunes a los distintos componentes de lógica de negocio para la ejecución del componente generics  Contiene clases comunes a todos los componentes de acceso a datos como ser clases bases y fabricas para la ejecución del componente generics  Contiene entidades de negocio para el componente generics | Biblioteca de clases |
| SilesWeb.Negocio.Arquitectura.Model | Contiene clases abstractas para los distintos componentes de lógica de negocio | Biblioteca de clases |
| SilesWeb.Negocio.Arquitectura.Security | Contiene clases de negocio trasversales para el manejo del contexto del aplicativo | Biblioteca de clases |
| SilesWeb.Presentacion.Arquitectura | Contiene clases comunes y trasversales a todos los componentes de presentación | Biblioteca de clases |
| SilesWeb.Presentacion.Arquite  ctura.Generics | Contiene clases comunes y trasversales para el componente generics | Biblioteca de clases |
| SilesWeb.Datos.[Macroprocesos  ].AccesoDatos | Provee acceso a datos de la base de datos de SilesWeb para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb.Datos.[Macroprocesos  ].Interfaces | Alberga las interfaxes de C# para los distintos objetos de acceso a datos para el para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb.Datos.[Macroprocesos  ].Modelos | Contiene el modelo de Entity Framework para realizar el acceso a datos de la base de datos del SilesWeb para los macroprocesos implementados nueva arquitectura |  |
| SilesWeb.Negocio.[Macro  proceso].DTO | Este proyecto tiene un conjunto de clases (DTO) cuyo objetivo es agrupar información (no necesariamente relacionada en la realidad) para servir a la capa de presentación de modo de evitar round-trips | Biblioteca de clases |
| SilesWeb.Negocio.[Macro  procesos].EntidadesNegocio | Alberga las clases que representan las entidades de negocio C# para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb.Negocio.[Macro  Proceso].Fachada | Contiene una clase única que sirve como punto de entrada para los macros procesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb.Negocio.[Macro  Proceso].Interface | Alberga las interfaces de C# para los distintos objetos de lógica de negocio para los macroprocesos implementados en la nueva arquitectura | Biblioteca de clases |
| SilesWeb.Negocio.[Macro  Proceso].ProcesosNegocio | Alberga las clases que representan los procesos de negocio C# para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb. Negocio.[ Macro-proceso ].ServiciosWeb.Fachada | Contiene una clase única que sirve como punto de entrada al servicio WCF para los macroprocesos omplementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb. Negocio.[Macro-proceso ].ServiciosWeb.Interface | Alberga las interfaces que implementan la funcionalidad disponibles en el servicio WCF para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb. Presentacion.[ Ma-croproceso] .Web.Controladores | Contiene las clases que implementan los contrladores MVC para los macroprocesos implementados en la nueva arquitectura | Biblioteca de clases |
| SilesWeb.Presentacion.[Macroproceso].Web.EntidadesWeb | Contiene las clases que implementan los modelos MVC para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb.Datos.Comun | Contiene clases comunes y trasversales a todos los componentes de base de datos como enumeradores y cosntantes | Biblioteca de Clases |
| SilesWeb.Datos.ServicioBD | Contiene el modelo de entity Framework para realizar el acceso a datos de la base de datos del silesWeb para los componentes comunes del aplicativo y un servicio WCF (WebDataServices) para el acceso desde la capa de negocio | Protecto de servicios WCF |
| SilesWeb. Servicios.[ Macroproceso ].Modelos | Contiene el modelo Entity Framework para realizar acceso a datos a la base de datos silesWeb para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| DatosComun.Modelo | Contiene el modelo entity framework para realizar el acceso a datos de la base de datos del SilesWeb para compnentes comunes a los diferentes macroprocesos | Biblioteca de clases |
| Datos[Macroproceso].Modelo | Contiene el modelo de Entity Framework para realizar el acceso a datos d ela basde de datos de SilesWeb para los macroprocesos implementados en nueva arquitectura | Biblioteca de clases |
| SilesWeb.Negocio.NComun | Albega las clases que representan las entidades de negocio C# comunes a todos los macroprocesos | Biblioteca de clases |
| Negocio.Servicios.[Macroproceso].Dominio | Alberga las claes que representan las entidades de negocio C# propias del [Macroproceso] | Biblioteca de clases |
| Negocio.Servicios. [Macroproce-so].Excepciones | Alberga las clases que implementan las excepciones personalizadas propias del [Macroproceso] | Biblioteca de clases |
| Negocio.Servicios. [Macroproce-so].Fachada | Contiene las clases que sirven como punto de entrada al [Macroproceso] | Biblioteca de clases |
| Negocio.Servicios. [Macroproce-so].Interfaz | Alberga las interfaces de C# para los distintos objetos de lógica de negocio para el [Macroproceso] | Biblioteca de clases |
| Negocio.Servicios. [Macroproce-so].Inyeccion | Implementación del patrón injection para los métodos brindados por la lógica de negocio del [Macroproceso] | Biblioteca de clases |
| Negocio.Servicios. [Macroproce-so].Logica | Contiene las clases que implementan la lógica de negocio del [Macroproceso] | Biblioteca de clases |
| Negocio.Servicios. [Macroproce-so].Repositorio | Contiene las clases que brindan el acceso a los modelos de Entity Framework por medio de instrucciones LINQ para la manipulación de la información almacenada en la base de datos de SilesWeb para el [Macroproceso] | Biblioteca de clases |
| Negocio.Servicios. [Macroproce-so].WebServices | Representa la interfaz de servicio de la capa de negocio del sistema  Contiene servicios web para cada macroproceso | Proyecto de servicios WCF |
| SilesWeb. ServiciosWeb.NFabrica | Contiene la implementación de emulación para instanciemiento del servicio externo de la DIAN para permitir referenciarlos desde la capa de presentación del sistema | Proyecto de servicios WCF |
| SilesWeb.Negocio.ServiciosWeb | Represnta la interfaz de servicio de la capa de negocio del sistema  Contiene servicios web para los componentes comunes y transversales, generics y seguridad | Proyecto de servicios WCF |
| SilesWeb. Negocio.ServiciosWeb.NFabrica.Config | Contiene la implementación dela configuración para el servicio externo de la DIAN para permitir referenciarlos desde la capa de presentación del sistema | Proyecto de servicios WCF |
| SilesWeb. Negocio.CUN.ServiciosWeb | Representa al interfaz de servicio de la capa de negocio del sistema.  Contiene servicios web para los componentes a los diferentes macroprocesos | Proyecto de servicios WCF |
| SilesWeb. Negocio.CUN.ServiciosWeb.Fachada | Contiene una clase única que sirve como punto de entrada al servicio WCF del componente anterior | Biblioteca de clases |
| SilesWeb. Negocio.PRM.ServiciosWeb | Representa la interfaz de servicio de la capa de negocio del sistema  Contiene servicios web para el componente de programador de tareas |  |
| SilesWeb.Presentacion.Web | Este proyecto web implementa los servicios de presentación y de la aplicación | Proyecto de aplicación web ASP.NET MVC |
| SilesWeb. Presentacion.CAM.Web.Controladores | Contiene las clases que implementan los controladores MVC para el componente de carga masiva | Biblioteca de clases |
| SilesWeb. Presentacion.CAM.Web.EntidadesWeb | Contiene las clases que implementan los modelos MVC para el componente carga masiva | Biblioteca de clases |
| SilesWeb. Presentacion.Comun.Web.Controladores | Contiene las clases que implementan los controladores MVC para los componentes comunes del aplicativo | Biblioteca de clases |
| SilesWen.Presentacion.Comun.Web.Modelos | Contiene las clases que implementan los modelos MVC para los componentes comunes del aplicativo | Biblioteca de clases |
| SilesWeb.Presentacion.PRM.Web.Controladores | Contiene las clases que implementan los controladores MVC para el componente de programador de tareas | Biblioteca de clases |
| SilesWeb. Presentacion.PRM.Web.EntidadesWeb | Contiene las clases que implementan los modelos MVC para el componente de programador de tarea | Biblioteca de clases |

Figura 26 Infraestructura de desarrollo de la apliacion en nueva Arquitectura

Imagen

4.7 Decisiones de Arquitectura

Jerarquía Proyecto MVC:

La figura 27,muestra la interfaz y los mecanismos de interaccion con el usuario, para ello se utiliza lenguaje marcado, como XHTML, HTML , CSS y/o JavaScript, acompañado con bloques de código de lado del servidor.

Siles resuelve su renderizacion de vista mediante tecnología AJAX para enviar y obtener información desde el servidor, siempre mediante llamados a acciones definidas en el controlador, que permitan crear transacciones mas dinámicas, sin saltos visuales al usuario y actualizaciones de segmentos HTML, sin recargar la pagina, haciendo el proceso de renderización mas limpio.

Figura 27. Jerarquía protecto MVC



En nueva arquitectura no se utilizan controles de servidor (label,Buton,Dropdowns,etc), son sustituidos por elementos HTML equivalentes, lo que implica perder la simplicidad de los Web Forms.

<for each="var name in names">

<test if="name == 'Jose'">

<p>Yo mismo</p>

<else/>

<p>Amigo: ${name}

</test>

</for>

Sin embargo, se utilizan otras ventajas de la arquitectura ASP.NET MVC, como por ejemplo el uso de motores de vistas distintos al estanda. Existen diferentes motores ligeros(NHaml,Spark,Brail,NVelocity, etc), cada uno con su propio lenguaje de marcas y convenciones, que permiten la definición de vistas a partir de plantillas.

Siles utiliza MVC con técnicas que permiten mejorar la calidad del software y extender el Framework, hace manejo de inyección de dependencias, que es una técnica bastante desacoplada entre si, lo que flexibiliza el diseño y facilita la realización de pruebas unitarias. Siles hace uso de elementos de cliente como Jquery, Json y Ajax con el fin de optimizar los comportamientos asíncronos al servidor.

Enrutamiento

Cada dominio de negocio, comprende un macroproceso y subdominio, esto esta representado mediante un controlador particular para respnder a un conjunto de Vista de Subdominio.

Las URL no correspnden a archivos, debido a que las solicitudes son manejadas por los controladores, teniendo un mejor contrl de las URL, lo cual nos permitirá cualquier patrin de mapeo URL-Controlador.

El sistema de ruteo fue diseñado por ASP.NET MVC, dicho modulo se encuentra en System.Web.Routing.

En Siles, el Request o Solicitud inicial ingresa a un Controlador principal definido como Homecontroller, la información del contexto se ejecuta en este controlador, garantizando el acceso seguro al sistema de enrutamiento.

El HomeController redirecciona la solicitud a la acción principal de cada controlador a nivel transaccional y de Subdominio para que la transacción previa sea inicializada.

El Homecontroller carga esta información de un archivo definido como Route.sitemap(ver figura 28), donde se alojan las transacciones a ser lanzadas en tiempo de ejecución.

Figura 28. Route.Sitemap



Cada controlador cuenta con una acción principal que interactua entre modelo y controlador para renderizar la UI mediante el Motor de Vistas RAZOR de MVC, plantilla que se utiliza para SILES.

*Contexto de Información:*

Se define para Siles como alojar un conjunto de datos de información sincronizados y equiparados con el actor que esta en sesión, que se encuentran disponibles en cada apertura de Usuario.

Ejemplo:

* El usuario crojasp pertenece a la Entidad SENA.
* La identificación de Usuario crojasp es XXXX
* El Nombre del usuario es Claudia Rojas Piñera

Información que además es altamente clasificada para el sistema, y será utilizada es la medida que se requiera en tiempo de ejecución, que será el estado de la aplicación en la cual esta información se debe generar, almacenar en sesión y compartir a cada uno de los controladores que manejan la transaccionabilidad.

La clase contexto de sesión esta inicializada por primera vez en handler navigation y carga propiedades que están creadas en la clase BaseController que a su vez hereda de la clase Controller.

* HomeController lee las propiedades del request



* HomeController crea un contexto de información seguro a partir de la solicitud HTTP del sitio de presentación de Siles instanciando y cargando las propiedades de sesión de la clase BaseController