Fwk

Fwk Framework Libraries

Componentes / Bloques Principales del Framework

Versión 1.0.0

¡Error! No se encuentran entradas de índice.

Objetivo principal

Tiene como objetivo principal, definir un marco de trabajo común para la factoría de software de todos los proyectos de la compañía con bloques de componentes probados, para lograr soluciones basadas en una tecnología que nace de prácticas exitosas ante problemas arraigados con el transcurso del tiempo.

Básicamente el Framework posee dos grandes ventajas:

Reducción de los costos de un proyecto cuando el profesional de sistemas utiliza los conocimientos de experiencias pasadas puestas en un marco de trabajo para ahorrar tiempo y dinero en sus aplicaciones, obteniendo, así, óptimos resultados a un riesgo reducido.

Obtención de soluciones confiables apoyadas en guía segura, código de calidad y recomendaciones de profesionales IT

Que no es el Framework ?

El Framework no es una solución definitiva o final, por un lado debido a que debemos adaptarnos continuamente al medio tecnológico en continuo crecimiento y es necesario actualizar los diferentes bloques modificando o agregándoles las funcionalidades necesarias para estar siempre a la vanguardia.

Por otra parte el Framework se vera afectado por cambios realizados por los propios profesionales de la empresa, que a través de su experiencia y por consenso deciden que se deberá realizar mejoras en alguno de los diferentes componentes que ofrece el Framework.

Secciones del Framework

El Framework de Fwk esta dividido en cuatro secciones con el objetivo de separar complejidad y funcionalidades comunes en ámbitos distintos.

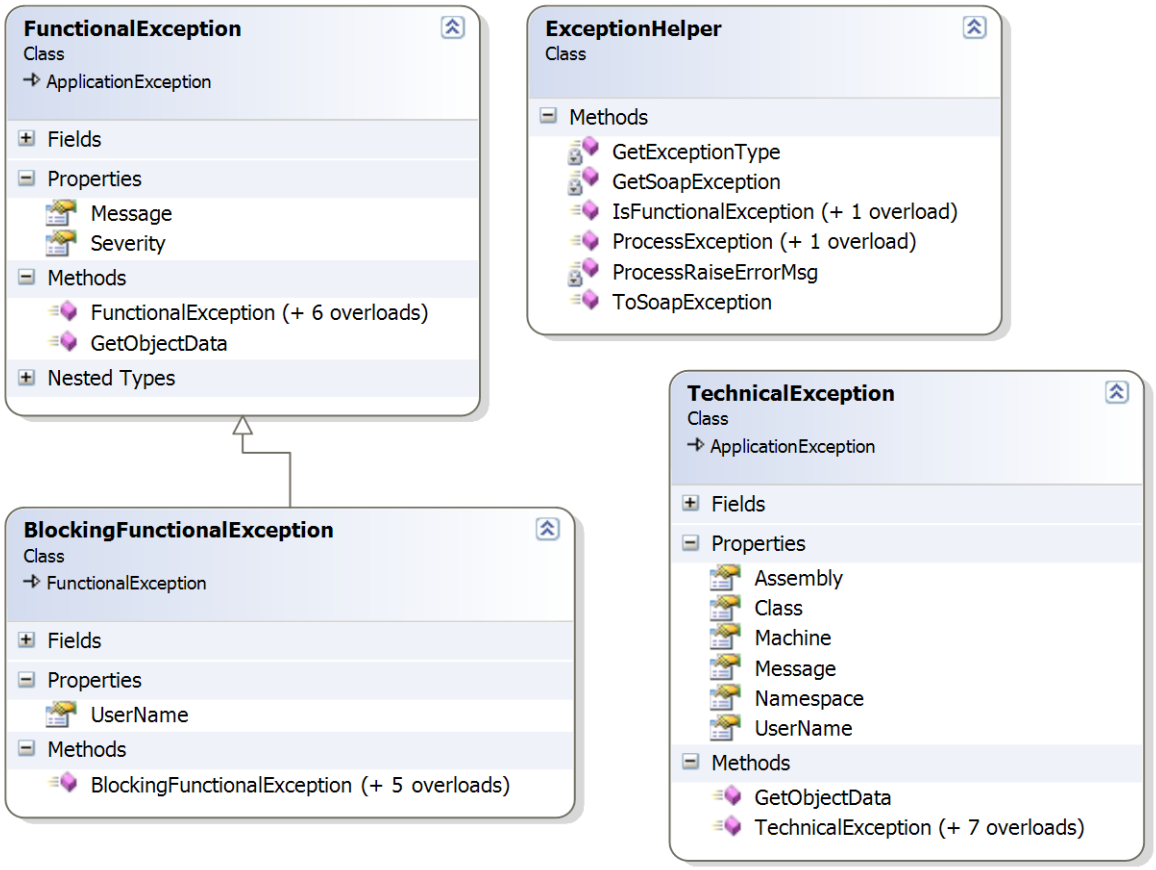
Consta de las siguientes secciones:

- **Sección Útils**

Formado por 4 bloques:

Fwk.Exceptions

Permite a los desarrolladores crear una estrategia consistente para procesar las excepciones o errores que surgen a través de las capas de arquitectura de las aplicaciones empresariales.



Fwk.Exceptions.Viewer

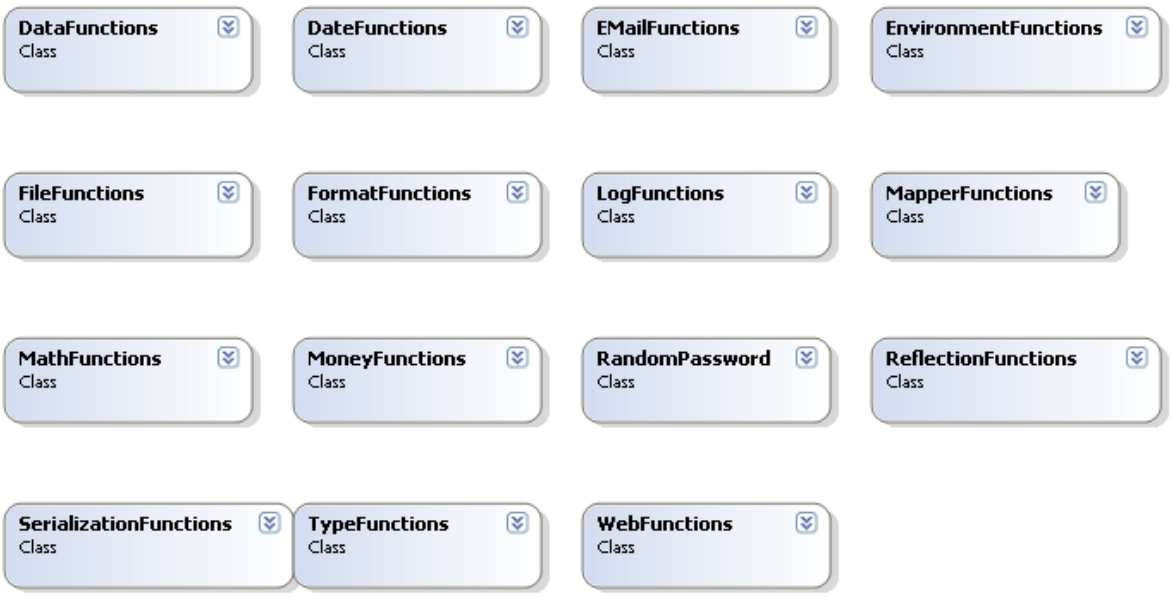
Ofrece facilidades para mostrar excepciones funcionales y técnicas del Framework.

Fwk.HelperFunctions

Son helpers que abstraen problemáticas comunes que enfrenta el desarrollador el transcurso del tiempo; como por ejemplo lectura de archivos, serializacion, manejo de objetos ADO, etc.-

El bloque HelperFunctions dispone de varias clases especializadas que ofrecen métodos estáticos para cada problemática diferente.-

A continuación se muestra el listado de todos las clases de forma grafica.-



La siguiente tabla describe brevemente el significado de cada una de estas clases:

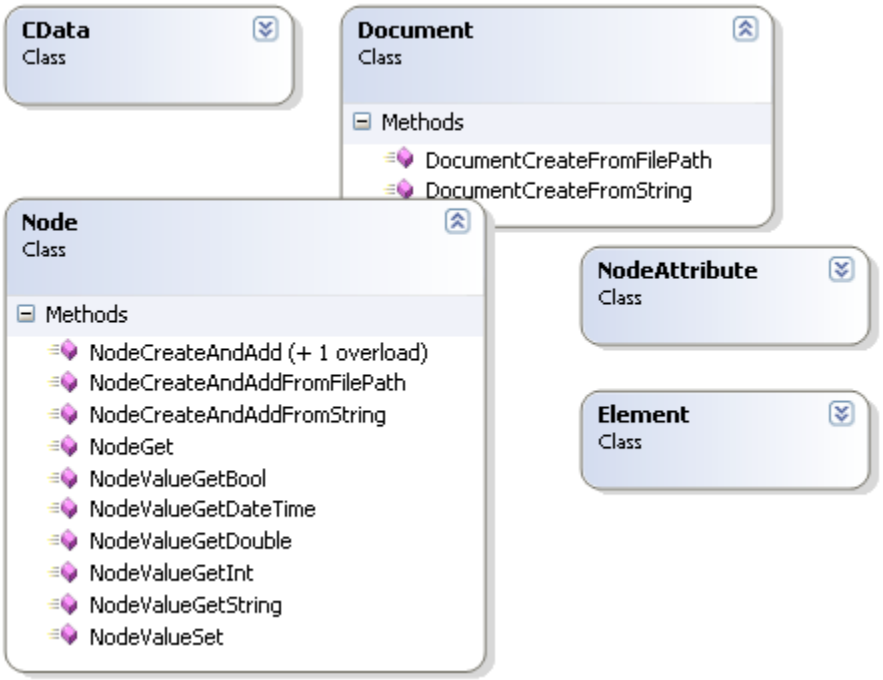
Class	Description
DataFunctions	Funciones de datos.
DateFunctions	Utilidades para fechas.
EMailFunctions	Funciones para envío de correo electrónico.
EnvironmentFunctions	Funciones que obtienen datos del Sistema Operativo.
FileFunctions	Funciones para manejo de archivos.
FormatFunctions	Funciones para realizar conversiones.
LogFunctions	Provee funcionalidades para el manejo de logs de nuestras aplicaciones.-
MathFunctions	Funciones matemáticas.
MoneyFunctions	Funciones de valores monetarios.
SerializationFunctions	Esta clase ayuda con los problemas que tienen que ver con la Serializacion de objetos.

TypeFunctions	Funciones de Tipos.
WebFunctions	Funciones para paginas Web.

Fwk.Xml

Es una extensión de las HelperFunctions que ofrece diferentes facilidades para la manipulación de objetos xml como por ejemplo XmlDocuments, XmlNodes, etc.-

Vista grafica:



Clases

Class	Description
CData	Manejo de CData.
Document	Manejo de XmlDocument.

Element	Manejo de XmlElement.

Node	Manejo de XmlNode.
NodeAttribute	Manejo de Attribute.

Fwk.Transaction

Este bloque dispone de una clase para encapsular manejo de transacciones:

TransactionScopeHandler

Encapsula la instanciación de los tipos necesarios para llevar a cabo una transacción, independizando el resto de la aplicación de la tecnología utilizada a su efecto.

Este componente ofrese los siguientes metodos para poder administrar las transacciones.

ConfigureScope

Configura los parámetros de inicialización para el alcance de la transacción.

```
public void ConfigureScope(
    TransactionalBehaviour pTransactionalBehaviour,
    IsolationLevel pIsolationLevel,
    TimeSpan pTimeOut
)
```

Donde el significado de los parametros son:

TransactionalBehaviour

Define Comportamiento del ámbito transaccional y puede ser cualquiera de los comportamientos que se detallan en la tabla a continuación:

Nombre de miembro	Descripción
Support	Si hay una transacción en curso, se hace uso de dicho ámbito. En caso contrario no transacciona.
Required	El servicio transacciona. Si hay un ámbito transaccional ya abierto, utiliza el existente, en caso contrario crea una nueva transaccion.

RequiresNew	Siempre se crea una nueva transacción.
Suppres	No transacciona, todas las operaciones se hacen sin estar en un ámbito transaccional.
None	No hace nada respecto del modo transaccional, es decir, el delegate no se llama a través de COM+, como ocurre en el resto de los casos.

Nota: todos los modos, salvo **None**, se corresponden con las opciones de transacción de COM+. **None** quiere decir sencillamente que no se utilizará COM+ para llevar a cabo la operación.

`IsolationLevel` Especifica el nivel de aislamiento de la transacción.

Nombre de miembro	Descripción
Chaos	Los cambios pendientes de transacciones más aisladas no puede ser sobrescritos.
ReadCommitted	Los datos volátites no puede ser leidos durante la transacción, pero pueden ser modificados.
ReadUncommitted	Los datos volátines pueden ser leidos y modificados durante la transacción.
RepeatableRead	Los datos volátiles pueden ser leidos pero no modificados durante la transación. Nuevos datos pueden ser creados.
Serializable	Los datos volátiles pueden ser leidos pero no modificados, y no es posible crear nuevos datos durante la transacción.
Snapshot	Los datos volátiles pueden ser leidos. Antes de modificarse datos, se verifica que otra transacción los haya cambiado luego de haber sido leidos. Si los datos se actualizaron, se levanta una excepción

InitScope ()

Inicializa el ámbito de transacción.

Complete ()

Este método Indica que se debe completar la ejecución de la transacción.

Abort()

Aborta las actualizaciones realizadas dentro del alcance de la transacción.

Fwk.Security

Permite a los desarrolladores incluir funcionalidad de encriptación y hashing en sus aplicaciones. Permite a los desarrolladores incorporar funcionalidad de seguridad en sus aplicaciones. Dichas aplicaciones pueden servirse del Fwk.Security en distintas situaciones, como a la hora de autenticar y autorizar a los usuarios contra una base de datos, obtener información de perfiles y roles, y cachear dicha información de usuario.

Seccion Configuration

Estos bloques permiten a las aplicaciones leer y escribir información de configuración tanto a las aplicaciones clientes como servidores.

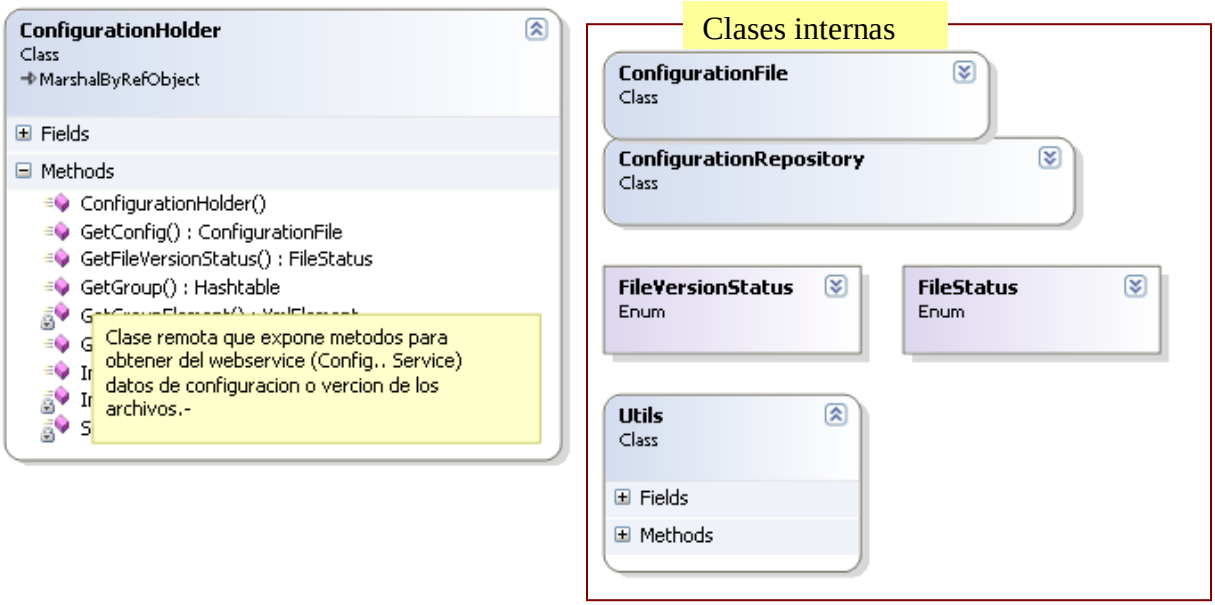
Fwk.Configuration

Ofrece una clase ConfigurationManager que tiene la inteligencia para obtener la configuración desde el repositorio indicado. Esta clase por si sola determina si la configuración la obtienen de manera local o remota a través de un servicio de Windows (ConfigurationProvider)

Fwk.Configuration.Common

Expone una serie de clases que tienen la lógica para obtener o proveer información de configuración y versionado de archivos de configuración a través de un servicio web

La clase mas importante es ConfigurationHolder que hereda de MarshalByRefObject para permitir obtener acceso a objetos a través de los límites de los dominios de las aplicaciones que admiten acceso remoto.



Fwk.Configuration.ConfigurationProvider

Los objetos de servicios remotos .NET anunciados arriba (ConfigurationHolder) se alojan en una infraestructura de servicios de componentes .NET. (servicio de windows). Este servicio es el encargado de obtener información sobre los archivos de configuración desde el WbService (ConfigurationService) que es el real problema

Fwk.Configuration.ConfigurationService

Es un servicio web al que todos los ConfigurationProvider o servicios de Windows pueden acceder para obtener los archivos de configuración.-
Básicamente este WS dispone de un xml con el catalogo de todos los archivos ConfigurationManager*.xml y un repositorio con todos los archivos físicos.

Seccion Bases

Esta sección ofrece un set de librerías que tienen por objetivo brindar las clases bases con implementaciones comunes para aquellas que la hereden, como por ejemplo : Clases de Back End DALC, Clases de Front End de formularios base.-, clases de servicios que constituyen los services contract, etc.-

También ofrecen ciertas interfaces de implementación para los servicios que implementan aquellas aplicaciones basadas en arquitecturas SOA del Framework de Fwk.

Las bases estan distribuidas en los siguientes bloques:

Fwk.Bases.BackEnd

Son un conjunto de clases bases e interfases de Back-End como las detalladas a continuación:

`BusinessComponent`

Clase abstracta de la que deben heredar los distintos componentes de negocio.

Las clases derivadas de ésta son las encargadas de implementar las reglas de negocio y realizar la validación de datos.

`DataAccessComponent`

Clase abstracta que debe ser heredada por las clases encargadas de controlar el acceso a datos.

Las clases que hereden de ésta tienen la responsabilidad de administrar la ejecución de componentes de persistencia.

`Entity`

Clase base abstracta que define comportamiento para las clses tipo Entidad.-

Las clases tipo entidad no tienen comportamiento de negocio solo los comportamientos heredados de su clase base.

`Entities`

Clase base que define el comportamiento general de todas las clases tipo colección de Entidad.

Las clases de este tipo son contenedoras o colecciones de las clases que heredan de `Entity` y definen el comportamiento común a todas estas colecciones.

La clase base `Entities` hereda de `List<T>`, absorbiendo todas sus ventajas automaticamente.

`BusinessService`

Clase base que representa un servicio de negocio abstracto. Es la clase de la que deben heredar todas aquellas clases que sean implementaciones de servicios de negocio.

`Request y Response`

Son las clases bases para todos los contratos de envio y recibo de informaccion de los servicios

- `Request` para interfases de envió.
- `Response` para interfases de recibo.

Estas son clases que reciben un generis de tipo `Entity` que define el comportamiento Funcional de envió o recibo de información. Esta es información de negocio

Para el registro de información técnica del servicio se utiliza un atributo "`ContextInformation`" que también es provisto por estas clases bases.

`ContextInformation`

Clase que establece la información de contexto de un `Request` de un servicio. La información puede ser por ejemplo el tiempo de procesamiento del servicio del lado del servidor o del cliente o el tiempo de respuesta para un `Response`.

Interfases

`IBusinessFacade`

Interfaz para clases que ejecutan un servicio de negocio.

Esta interfaz deberá ser implementada por todas las clases que ejecuten un servicio de negocio.

Estas clases deberán implementar el comportamiento no funcional que no es provisto por el servicio que expone los servicios de negocio (seguridad, transaccionalidad, logueo, etc).

`IServiceContract`

Interfaz que implementan las clases bases `Request y Response,` es decir es la interfaz común que todo contrato de servicio dispone.

`IServiceInterfaceWrapper`

Interfaz para clases que funcionan como wrappers de interfaces de servicio.

Las clases que implementen esta interfaz deben ser capaces de solicitar la ejecución de servicios de negocio a través de métodos de la interfaz de servicio utilizada, y devolver el resultado del mismo como resultado de la ejecución.

Fwk.Bases.FrontEnd y

Fwk.Bases.FrontEndWeb

Estos dos bloques exponen clases bases de Front End de la cual las aplicaciones, ya se Win32 o Web, las heredan con el fin de unificar la complejidad de ciertas tareas comunes como por ejemplo:

- La ejecución de un servicio
- La localización de un servidor Dispatcher Win service o Web Service
- Manejo del contexto de información del lado del cliente
- Manejo de Blocking
- Manejo del cacheo de servicios
- Etc

❑ Seccion Blocking

Esta sección cuenta con dos bloques que tienen como objetivo manejar todo la problemática de bloqueo de datos ante el acceso concurrente de información en un sistema.

Fwk.Blocking.PessimisticBlocking

Este bloque contiene el engine o motor de bloqueo que se encarga de administrar todas las marcas de bloqueos, valga la redundancia, de una base de datos.

Sus funciones principales son las siguientes:

- Liberación de un contexto de bloqueo, borrando todas las marcas que posean el mismo id de Contexto.
- Crear un contexto de bloqueo

Los componentes prioncipales de este bloque son `BlockingMark` y `ContextInfo` y se detallan a continuacion:

`BlockingMark`

Son las marcas de bloqueo definida por una tabla. Es decir con este componente podemos identificar que tabla de las base de datos de un sitema determinado esta sindo bloqueada.

Para identificar la informacion que esta ciendo bloqueada se utiliza la siguiente informacion dentro del componente `BlockingMark`:

- **Table:** que indica en que tabla estan los registros a bloquear
- **Attribute:** `que` indica cual es el nombre del campo que se utiliza como clave para ubicar un registro determinado. -

`ContextInfo`

Es lo que entendemos por Contexto de bloqueo. Es desir, toda la información de contexto de todos los registros a bloquear por un Blocking Service.-

La clase `ContextInfo` es la contenedora del contexto de bloqueo y contiene informacion como la que se muestra en la tabla a continuiacion :

Atributo	Descripción
ContextID	Guid del contexto de bloqueo.
TTL	Time-To-Live. En segundos
BlockingTime	Tiempo de espera antes de lanzar excepcion cuando se esta esperando que se liberen marcas de bloqueo.
User	Usuario que bloquea.
BackupMarks	Determina si el motor de bloqueo hace backup de las marcas de bloqueo.

Fwk.Blocking.PessimisticBlockingService

Es el Servicio de blocking. Se trata de un servicio de windows que corre en forma de Deamon leyendo la tabla de bloqueos y viendo por cada marca si se expira el tiempo para desmarcar el bloqueo y asi liberar el recurso.

Sección Logging

Este Block permite a los desarrolladores incorporar un sistema de Loggin estándar y funcionalidad de instrumentación en sus aplicaciones.

Seccion Implementation

Esta sección esta constituida por aplicaciones y componentes que utilizan los bloques anteriores y que se encargan de la administración, elección y configuración de los servicios.

Service Disptatcher - "SingleServiceInterface"

Es un servicio Web que actúa como despachador de servicios. Atiende todas las peticiones de los clientes y retorna el resultado de la ejecución del servicio a cada uno.

El Dispatcher dispone de un catalogo con toda la información de configuración de los servicios. Esta información puede estar implementada sobre una base de datos u un archivo Xml .

El Dispatcher también podría estar implementado en un Servicio de Windows que exponga una clase remota que realice exactamente lo mismo que es Web Service.

En este documento solo nombraremos el "***SingleServiceInterface***" ya que no se intenta aquí dar una información detallada de ventajas y desventajas de implementación en distintas tecnologías, sino que se pretende dar un pantallaso general de las distintos secciones y bloques que constituyen el Framework de Fwk.

Algunas de las funcionalidades básicas que provee como Dispatcher son:

- Entorno transaccional
- Control de ejecución de servicios
- Sistema de mensajes y configuración
- Manejo de excepciones
- Sistema de conectores para acceder a plataformas no integradas nativamente al sistema de información
- Manejo de cacheo de servicios
- Manejo del contexto de transacción de servicio
- etc.

- **Wrapper de aplicación**

El Wrapper es el único que puede interactuar con la capa de servicios de negocio para resolver requerimientos de negocio. Dicha interacción se realiza a través de un adaptador de un canal, enviando un `Request` y esperando la respuesta en un `Response`.

Los adaptadores de canal son piezas de código que permiten desacoplar el envío de un requerimiento a un servicio de negocio, con su correspondiente respuesta, de la tecnología de transporte necesaria para acceder a dicho servicio de negocio (de consulta o transaccional).

El sistema de adaptadores de canal es extensible, de esta forma se podrían ampliar los tipos de canales de acceso a los servicios. A través de ellos se podrían ejecutar servicios desde Web Services, Win Services, Páginas WEB, Aplicaciones Winform, Etc.

Para que se de esta flexibilidad en los adaptadores, el Wrapper debe implementar una interfaz llamada `IServiceInterfaceWrapper`. Es desir, las clases Wrapper deben ser capaces de solicitar la ejecución de servicios de negocio a través de métodos de la intefaz de servicio utilizada, y devolver el resultado del mismo como resultado de la ejecución.