StartFrame Net Framework

Manual Técnico

El objetivo de este manual es describir toda la información técnica pertinente para un arquitecto, analista o programador que desee utilizar el producto. Podrá encontrar dentro de este manual una descripción de la arquitectura interna del sistema, la estructura multicapas utilizada, el manejo multicultural, el sistema interno de seguridad, el mecanismo de control de errores, y el soporte para manejo multiempresa, entre otros aspectos igualmente importantes.

Se destaca que el lector de este manual debe tener un perfil sistémico avanzado.

iTSouth & CDT Consultores

Versión liberada el 01/10/2011

StartFrame Net Framework

Manual Técnico

Índice General

Contenido

INSTALACIÓN

Para instalar *StartFrame* en un entorno de desarrollo, deberá contar con Microsoft Visual Studio previamente instalado, ya que *StartFrame* se agrega como extensión del mismo. Luego de descargar *StartFrame* desde su sitio oficial, deberá seguir las instrucciones de la *Guía de Instalación*.

La mencionada extensión agrega un nuevo menú a su IDE de Visual Studio que le permitirá acceder a toda la documentación de *StartFrame*, agrega una serie de *code snippets*, registra algunos controles para utilizar en formularios y le permitirá utilizar este marco para crear nuevas aplicaciones a través de plantillas prediseñadas. Para crear una nueva solución, podrá utilizar los *templates de proyectos* que se instalan en su equipo al agregar la extensión antes mencionada.

Cada plantilla de solución ofrece información[1] de la máxima importancia en un archivo típico de cada instalación, el cual rara vez es leído:

> ***Leame.txt**: ofrece información elemental para configurar y ejecutar el sistema por primera vez.*

[1] Puede solicitar soporte adicional en StartFrame.Net

Procedimiento General de Instalación

Independientemente del tipo de plantilla de solución que desee utilizar, en general, los pasos de instalación son los siguientes:

1. Prerequisitos

En primer lugar, luego de crear el nuevo proyecto utilizando el *template* seleccionado, deberá verificar e instalar los prerequisitos de la plantilla elegida. En general, estos consisten en restaurar la base de datos modelo o instalar el runtime para los reportes de *SAP Crystal Reports* (si aplica al entorno elegido). La lista puntual de prerequisitos de cada plantilla está descripta en el archivo *Leame.txt*.

2. Plantilla de Solución

Al crear un nuevo proyecto basado en un *template* provisto por *StartFrame*, se crea una solución conteniendo uno o más proyectos adecuadamente configurados, y que podrá tomar como base para construir su aplicación sobre dicha estructura.

Si utilizó la plantilla de *Servicio Windows*, deberá configurarse para que inicie automáticamente cuando arranque el servidor. El servicio se puede instalar por medio de la aplicación *InstallUtil.exe*, que se encuentra en la carpeta *"C:\%WINDOWS%\Microsoft.NET\Framework\v…"* (donde la última carpeta depende de la versión del Framework en ejecución). La sintaxis para instalar el servicio es *InstallUtil NombreServicio.exe* (ejecutado desde el *Command Prompt* de *.Net*). Para desinstalarlo, simplemente agregarle el parámetro *"/u"* al comando anterior. Una vez instalado el servicio, es prudente verificar su configuración (cuenta utilizada, tipo de inicio, etc.). Cualquier error que genere el mencionado servicio podrá verse en el visor de sucesos de la aplicación. En caso de que se desee correr más de un servicio en el mismo servidor, debido a que existe más de una instancia del sistema, se deberá recompilar este ejecutable cambiando las siguientes propiedades:

- Installer1.vb → ServiceInstaller1.ServiceName = "nombreServicio"
- Installer1.vb → ServiceInstaller1.DisplayName = "Descrip. Del Servicio"

Por otro lado, si la plantilla a utilizar fuera la aplicación Web, podrá publicar la solución en un *IIS* (*Internet Information Services*) correctamente configurado en un entorno de producción. En un entorno de desarrollo, puede ejecutar la aplicación dentro de Visual Studio en su *localhost*.

3. Base de Datos

Como se explicará detalladamente en el capítulo *Conexión a la Base de Datos*, *StartFrame* trae un modelo de datos para soportar el esquema de seguridad, diccionario de datos y otras funcionalidades que requieren persistencia de almacenamiento. Este paso de la instalación, el cual está desarrollado en el archivo *Leame.txt*, consiste en implementar dicho modelo de datos.

Para instalar el mismo, deberá realizarse un *restore* de la base de datos modelo, o bien correrse los *scripts* que provee el instalador de los prerequisitos de *StartFrame*.

4. Usuario

Dependiendo de la plantilla de solución a utilizar, este paso podría no ser necesario. El mismo consiste en dar de alta un usuario dentro del esquema de seguridad de *StartFrame* para poder ejecutar la solución. El archivo *Leame.txt* brinda instrucciones a este respecto.

En el caso de la plantilla *Web App*, la aplicación está configurada por defecto para permitir registrar nuevos usuarios sin estar logueado a la misma.

5. Configuración Inicial

Este último paso consiste en modificar los archivos de configuración de la aplicación, cuyo nombre y formato particular varía según el tipo de plantilla de solución utilizada, pero siempre se trata de uno o más archivos *XML* con extensión *config*. Básicamente, encontrará en los mencionados archivos un *tag* similar al siguiente:

```
<configuration>
  <appSettings>
    <add key="cnnkey" value="sqlcnn" />
    <add key="sqlcnn" value="provider=SQLOLEDB.1;
        data source=NombreHost;
        database=NombreBase;
        user id=pRWPblld2UQ=;
        password=5SNDTwZBRQFflURclN2xjg==" />
        ...
  </appSettings>
    ...
</configuration>
```

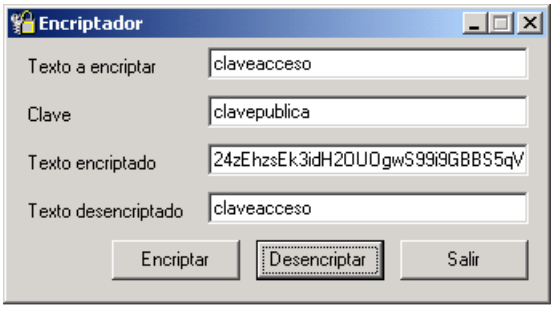
En el primer *tag* (cnnkey) se indicará el *tag* que contendrá la cadena de conexión a utilizar. Dentro del segundo, deberá modificar:

- El nombre del *OLEDB Provider* para conectarse al motor de base de datos (por defecto incluye el provider para *SQL Server*).
- El nombre del *host* o servidor (NombreHost) donde está instalada la base de datos.
- El nombre de la base de datos en sí (NombreBase).
- El nombre del usuario con acceso a dicha base (por defecto es "sa" encriptado).
- La clave de acceso de dicho usuario (por defecto es "admin" encriptado).

En caso de desea utilizar una conexión con seguridad integrada al sistema operativo, deberá realizar los siguientes cambios en el *tag* antes mencionado:

```
<configuration>
  <appSettings>
    <add key="sqlcnn" value="provider=SQLOLEDB.1;
        data source=NombreHost;
        database=NombreBase;
        user id=;
        password=;
        integrated security=SSPI" />
        ...
  </appSettings>
    ...
</configuration>
```

Para encriptar o desencriptar texto, podrá utilizar la aplicación provista en el menú *StartFrame* utilizando la clave pública "***cualquiera***" (sin las comillas). Dicha clave pública se encuentra compilada dentro de la aplicación en la propiedad *KEY* de la clase ***Common.Env***, la cual podrá ser modificada si lo desea.



Dependiendo del tipo de plantilla instalada, es posible que el o los archivos de configuración posean atributos con rutas absolutas o relativas a ciertas carpetas del aplicativo. Si este fuera el caso, verifique las mencionadas rutas.

En caso de utilizar *Remoting* como medio de comunicación inter-capas, referirse al capítulo *Remoting* para configurarlo apropiadamente.

En el caso de la plantilla *Web App*, si desea utilizar la seguridad por *forms*, deberá también establecer la cadena de conexión con el modelo de seguridad interno de *ASP.Net*, el cual reside en el elemento *SqlServices* dentro de *<connectionStrings>*. La forma de configurar dicha cadena de conexión es estándar y tiene la posibilidad de almacenarla en forma encriptada.

```
<connectionStrings configProtectionProvider="CustomProvider">
    <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
      xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Rsa Key</KeyName>
          </KeyInfo>
          <CipherData>
<CipherValue>NKJYZlonvRl8uxohGjKamAVydjgw5gP7AxCMaIbEyIgSGblYgKVxLuDGA3F0BjnASN9b2i9dCZ/
q/PNx/Xsit1i615WrldykU09iJJyVopqWi/USSCvgLEQoCtQvxVwdnTt6u1Y++gtCwua4OOMfX0qeCRpAf5c94rb
GF8BuN+E=</CipherValue>
          </CipherData>
        </EncryptedKey>
      </KeyInfo>
      <CipherData>
<CipherValue>HNPhI3xxL2zkQd+0ipYiGhNyShnyujN+/ZdaoGLtibs9CPqkNOQjU8YsPUnrQ9dmyDGubrP16p6
DZflVms5p4619pSS2qZ4vyu6B7qFz6vJIUZteE8yXCVTPGPHtXHDykdsfQHkQ1bOQ/6PXDlMszx3cuYCXa6zThnq
NsamNEF9ENpacuiejFyT/htaWAGoMfQ3MoH62hs+7THoYthefdfb2ZKKRtj/4KuStC98knmTOLwjKw8vcY3adGRP
v+NrohAcC7AoWpGvt/kWizhXutNLTYohdAIGYqeLzFWnJGWI=</CipherValue>
      </CipherData>
    </EncryptedData>
  </connectionStrings>
```

Para lograr esto, en primer lugar deberá abrir la consola de Visual Studio (buscar en *Inicio… Visual Studio… Tools*). En dicha consola, deberá ejecutar la siguiente línea para importar la *CustomKey* de forma tal de poder encriptar y desencriptar texto. Asegúrese que las rutas sean las correctas y el archivo *CustomKeys.xml* exista.

```
aspnet_regiis -pi "CustomKeys"
"D:\Desarrollo\Net\...\WebSite\SharedSources\CustomKeys.xml"
```

A continuación, para desencriptar la cadena de conexión, deberá ejecutar la siguiente línea en la misma consola de Visual Studio. El primer parámetro luego del modificador indica la sección del archivo *Web.config* a desencriptar, el segundo es la ruta de dicho archivo.

```
aspnet_regiis -pdf "connectionStrings" "D:\Desarrollo\Net\...\Fuentes\WebSite"
```

Luego de modificar la cadena de conexión como ya se explicó, deberá volver a encriptar dicha sección dentro del *web.config* ejecutando lo siguiente:

```
aspnet_regiis -pef "connectionStrings" "D:\Desarrollo\Net\...\Fuentes\WebSite" -prov
"CustomProvider"
```

Configuración Avanzada

Existen algunos *tags* en los archivos de configuración, dentro de la sección *appSettings*, que podría modificar para lograr personalizar la aplicación (algunos tags son aplicables sólo a ciertas arquitecturas de solución), a saber:

Tag	Descripción	Posible contenido
<serverusr>	Nombre del usuario con el cual se levantará la aplicación en el server.	Cualquier usuario real del sistema operativo (en mayúsculas y sin el dominio) y que esté registrado en el esquema interno de seguridad de *StartFrame*.
<eventlog>	Indica si deberán registrarse sucesos en el *Application Log* de *Windows.*	• True • False
<spconfigpath>	Contiene una ruta absoluta a la aplicación servidora donde el sistema buscará el archivo de definición de parámetros de los stored procedures	Cualquier ruta de red válida o bien puede comentarse todo este tag para que busque dicho archivo en la carpeta donde se está ejecutando la aplicación. Este parámetro es obligatorio en los casos de aplicaciones que ejecutan con objetos residentes en carpetas temporales de Windows.
<ActPath>	Ruta de red donde reside la última versión del cliente del sistema y desde la cual se realizarán las actualizaciones automáticamente (ver Actualizaciones).	Cualquier ruta de red válida o bien puede comentarse todo este tag para que no se actualice el sistema automáticamente.
<EstadoAbms>	Estado de apertura de las plantillas del tipo abm y consultor en el módulo desktop de *StartFrame.*	• Normal • Maximized
<EstiloAbms>	Indica el tipo de estilo a utilizar para las plantillas del tipo abm y consultor en el módulo desktop de *StartFrame.*	• PorDefecto • Normal • Flat • Gris • FlatGris
<ResizeGrillas>	Indica si la plantilla base del tipo abm para los formularios desktop realizará un *resize* automático de todas las grillas contenidas en el mismo.	• True • False
<WPMain.Estado>	Determina el estado inicial del formulario principal del módulo desktop.	• 1: Maximized • 2: Normal
<WPMain.Largo> **<WPMain.Alto>**	Determina el tamaño del formulario MDI del módulo desktop.	Cualquier entero mayor a 800*600 y menor a la resolución máxima del monitor.
<WPMain.X> **<WPMain.Y>**	Determina la posición inicial del formulario principal del módulo desktop.	Cualquier entero que represente un par de coordenadas de pantalla.
<MailErrores>	Dirección de mail a la cual se dirigirán los informes de errores en el procesamiento. Utilizado especialmente en la plantilla de servicios Windows de StartFrame.	Cualquier dirección de mail válida.
<DirLogs>	Directorio físico absoluto y ruta de red	Cualquier ruta de red válida con

<PathLogs>	asociada donde la aplicación dejará los logs de procesamiento. Utilizado en las plantillas de ejecución de procesos del lado del servidor y de servicios.	permisos de escritura.
<Intervalo>	Cantidad de minutos en la frecuencia de ejecución de los servicios Windows.	Cualquier entero mayor a 0.
<freeRegistration>	Indica si es o no posible crear nuevos usuarios sin estar previamente logueado a la aplicación, utilizado en la plantilla Web.	• On: permite crear nuevos usuarios aunque no esté logueado en la aplicación. • Off: sólo logueándose a la aplicación se puede crear un nuevo usuario.
<SqlServices>	Cadena de conexión para el modelo de datos de seguridad interna por forms de ASP.Net.	Cadena de conexión a la base de datos modelo.

Actualizaciones

La distribución (*deployment*) de los componentes de una determinada solución creada con *StartFrame* dependerá de la arquitectura elegida. Esto último condiciona la forma en que dicha aplicación deberá ser actualizada.

Para el caso de la arquitectura *N-Tiers en Windows Desktop*, el servidor deberá bajarse (ya sea en modo consola o servicio) para poder actualizar las DLLs que utiliza. El cliente, en cambio puede ser actualizado en forma semiautomática. Esto se logra copiando a la carpeta referenciada por el parámetro *<ActPath>* dentro del archivo de configuración local, todos los archivos que deban ser actualizados. El programa *US.loader.exe* se encargará de actualizar todos los archivos que sean más nuevos que la versión instalada en la terminal del cliente cuando vuelvan a ingresar al sistema. Se recomienda tener cuidado con los archivos de configuración que poseen parámetros que apuntan a otros equipos, ya que cualquier cambio podría hacer que la aplicación no funcione como se esperaba, que apunte a otro servidor de reglas de negocio o a otro motor de base de datos. Para los usuarios, los mencionados procesos son transparentes.

Para el caso de la arquitectura *Web App*, existen algunos archivos que podrían publicarse (copiarse) sin bajar el servidor Web, pero para otros será necesario detener el mismo. Esto depende más de IIS y la arquitectura de ASP.Net que *StartFrame* en sí.

Con respecto a las actualizaciones del producto *StartFrame*, al ser una extensión de Visual Studio, la misma se actualiza automáticamente. Esto incluye no sólo la documentación, *code snippets* y demás funcionalidades de *StartFrame*, sino también los *templates* para crear nuevas soluciones.

Ahora bien, las soluciones ya construidas con versiones anteriores de *StartFrame* podrán ser actualizadas mediante *Service Packs* conteniendo instrucciones detalladas que deberán seguirse. Por lo general, bastará con ejecutar el *Service Pack* para copiar los nuevos componentes de *StartFrame* a las correspondientes carpetas *SharedSources* y *SharedBinaries* y luego recompilar la solución. Sin embargo, hay que tener en cuenta que *StartFrame* está formado por algunos archivos adicionales de soporte que se encargan de tareas de configuración (*.config, menu*.xml). Algunos de ellos pueden ser sobreescritos directamente, ya que el desarrollador nunca debe modificarlos. En tanto que otros componentes deberán ser tratados con sumo cuidado debido a que los mismos podrán ser modificados tanto en *StartFrame* como en la aplicación desarrollada en base al mismo, a saber:

Componente	Observaciones
***.config**	Estos archivos de configuración contienen las entradas necesarias para la configuración de *Remoting* (para las arquitecturas *N-Tiers*), además de varios parámetros que determinan información general, la cadena de conexión, etc. Cuando un desarrollador utiliza *StartFrame* como herramienta de desarrollo, deberá agregar todas las entradas necesarias para el uso de los componentes vía *Remoting*. **Es necesario respetar siempre la estructura general del archivo establecida por *StartFrame***. El desarrollador deberá agregar toda la información necesaria sin alterar ni eliminar la información existente.
Common	Este proyecto es parte integral de *StartFrame* y la vía de comunicación del mismo con la aplicación generada. El mismo se instala y referencia a nivel proyecto incluido en la solución porque es necesario que el desarrollador pueda modificar: • **App.resources**: archivo de recursos de la aplicación, el cual puede ser tocado libremente.

- **Common.Start**: contiene eventos que pueden ser interceptados en la solución cada vez que se inicia un módulo, se cierra el mismo o se selecciona una opción del menú.

No deberá tocarse el resto del proyecto.

Podrá solicitar soporte adicional en StartFrame.Net para realizar este tipo de migraciones.

ARQUITECTURA

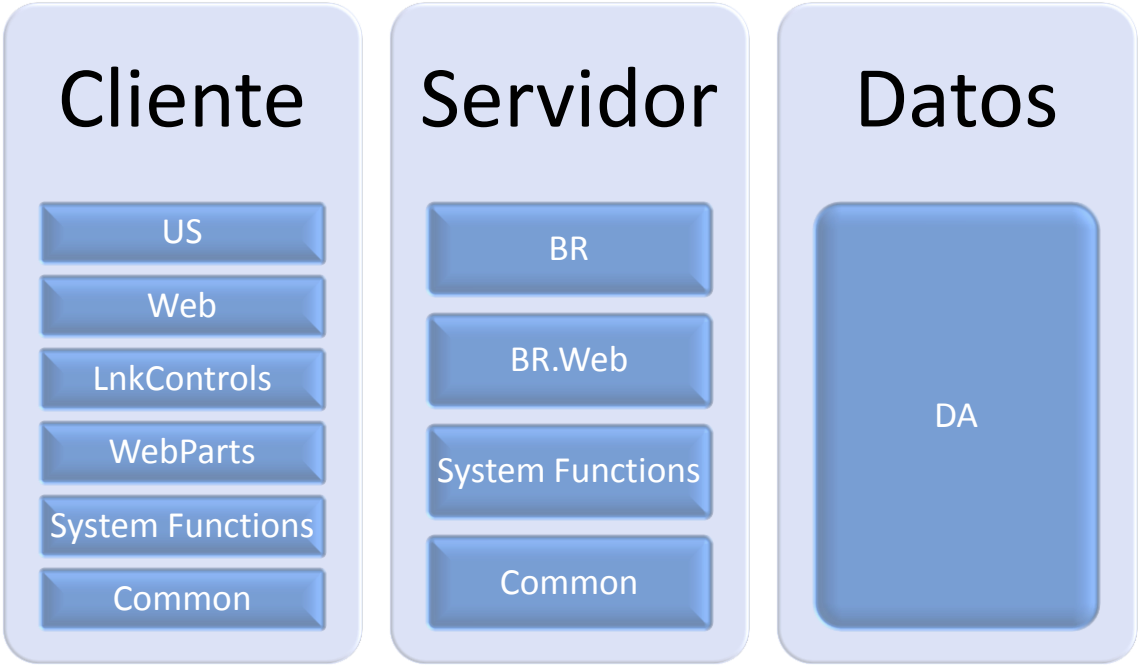
La arquitectura interna del producto responde a la base tecnológica establecida por la aplicación de las mejores prácticas sobre la plataforma *Microsoft dotNet*. Básicamente, *StartFrame* ofrece una serie de componentes reutilizables combinados en diferentes modelos de solución, a saber:

Cada arquitectura antes mencionada admite una o más plantillas diferentes. Por ejemplo, para la arquitectura desktop, pueden utilizarse plantillas para crear una aplicación *WinForms* o *Silverligth*, para la arquitectura Web, puede utilizarse la plantilla de aplicación *ASP* o *Silverlight*, para los servicios, pueden ser *Win Services* o *Web Services*.

Los diferentes modelos de solución pueden implementarse a través del uso de plantillas diseñadas a tal fin. La estructura física y lógica de cada modelo está íntimamente relacionada al diseño de dicha plantilla. Sin embargo, los componentes que lo conforman se mantienen en forma constante, independientemente de la arquitectura de la solución.

Componentes por Capa

Si bien la cantidad de capas de una solución depende de la arquitectura de la misma, podemos clasificar en forma genérica, sin ser tan estrictos, los componentes en tres capas bien diferenciadas:



Contenido de los Componentes

Se describe a continuación en forma genérica el contenido de cada componente antes mencionado. Podrá encontrar una completa descripción de cada elemento de *StartFrame* en los diagramas de clase, en el modelo *UML* de *StartFrame* y en las especificaciones de diseño detallado.

1. **US:** Servicios de usuario (*user services*) *desktop*. Está conformado por una serie de elementos que posibilitan la interfaz con el usuario a través de plantillas base para crear diferentes tipos de interfaces con el usuario, diálogos personalizables y controles vinculados a datos.
2. **Web:** Ídem anterior pero aplicado a las interfaces Web del tipo ASP.Net. Contiene además todos los elementos del entorno para configurar la aplicación (temas, archivos de configuración, páginas maestras, páginas de administración, páginas del usuario, librerías adicionales, scripts, etc.).
3. **LnkControls:** Librería de controles de usuario *desktop*.
4. **WebParts:** Librería de controles Web, los cuales pueden adicionarse a cualquier página HTML.
5. **SystemFunctions:** Librería de rutinas estándares para manipulación de cadenas, números, fechas, archivos, mails, etc.
6. **Common:** Componente común para todas las capas de una solución, conteniendo propiedades generales utilizables por toda la aplicación.
7. **BR:** Reglas del negocio (*business rules*) aplicables a cualquier modelo de solución corriendo del lado del servidor. Contiene una serie de clases abstractas para crear diferentes tipos de objetos, clases responsables de la seguridad interna del sistema, funciones de auditoría, diccionarios de datos, utilidades varias.
8. **BR.Web:** Objetos con las reglas del negocio vinculadas a las páginas de la aplicación Web. No se trata del *code behing* de cada página, sino del componente de negocio en sí.
9. **DA:** Acceso a datos (*data access*). Permite el acceso de lectura/escritura a cualquier motor de base de datos a través de *ADO.Net*.

Cada componente antes mencionado está formado por una o más librerías. Se unificó la lista precedente a efectos de ganar en claridad.

Estructura Física

La estructura física de la aplicación está vinculada al tipo de solución implementada, aplicando las mejores prácticas en cada caso.

Comunicación Interna

Al igual que la estructura lógica y física, la modalidad interna de comunicación de los componentes de *StartFrame* depende de la arquitectura de la aplicación que lo consuma.

En un entorno *LAN*, se utiliza *Remoting* por ser la opción más rápida para esta plataforma. En una *Aplicación Web*, del lado del servidor, se instancian las clases en forma directa. Si desea una mayor interoperabilidad, deberá crear un proyecto del tipo *WCF* para sus soluciones (*StartFrame* soporta cualquier modelo de comunicación).

En un entorno Web, la comunicación entre las páginas (html desplegado en el explorador) y el código del lado del servidor, es responsabilidad de la tecnología Web en general y la arquitectura de ASP estándar en particular. En cuanto a la comunicación del *code behing* con los componentes de negocios y las librerías de *StartFrame*, se realiza en forma local (instanciando las clases directamente), sin mediar ningún componente intermedio, logrando una óptima velocidad de respuesta.

Con respecto a la comunicación con la base de datos, se utiliza *ADO.Net* con *OLEDB provider* para lograr una mayor flexibilidad. En caso de optar por utilizar algún motor de base de datos diferente a *SQL Server* (la opción por defecto), tan solo deberá modificar en la cadena de conexión el nombre del *OLEDB provider* actual por el proveedor adecuado. Esta acción incluso puede realizarse en *runtime* sin necesidad de adaptar una sola línea de código.

Remoting

Como se dijo anteriormente, por una decisión de performance, se optó por utilizar *Remoting* como medio de comunicación de los componentes en entornos *LAN* distribuidos (cuando la lógica de negocios se encuentra corriendo en un equipo diferente al que contiene el ejecutable local de la interfaz del usuario).

Este esquema de comunicación cuenta con componentes del lado del cliente y del lado del servidor, a saber:

1. Elementos del lado del cliente:
    a. **US.ini.exe.config:** Archivo XML que contiene la información sobre la configuración y la definición de cada interfaz que utiliza con la ubicación del objeto de negocios que la implementa. Este modelo de configuración permite redirigir en grandes redes las diferentes terminales a distintos servidores de negocios. Este archivo se encuentra en la carpeta *SharedSources* y compila al mismo directorio que el ejecutable del producto. Pueden utilizarse *code snippets* para agregar nuevas entradas en este archivo.

```
<configuration>
  <system.runtime.remoting>
    <application>
      <client url="tcp://localhost:1234">
        <wellknown type="StartFrame.BR.Interfaces.ISql,IBR" url="tcp://localhost:1234/Sql" />
        <wellknown type="StartFrame.BR.Interfaces.ISca,IBR" url="tcp://localhost:1234/Sca" />
        <wellknown type="StartFrame.BR.Interfaces.IAbm,IBR" url="tcp://localhost:1234/Abm" />
          ...
          ...
      </client>
    </application>
  </system.runtime.remoting>
</configuration>
```

    b. **RemotingHelper:** Componente a través del cual se instancian todas las clases de la capa de negocios. Internamente, para obtener una referencia al objeto de negocios, se

basa en la información de la interfaz definida en el archivo antes mencionado. Pueden utilizarse *code snippets* que faciliten la escritura de este código.

```
Dim _Sql As ISql
_Sql = CType(RemotingHelper.getObject(GetType(ISql)), ISql)
```

2. Elementos del lado del servidor:
   a. **Server.exe.config**[2]**:** Archivo XML similar al residente en el cliente, pero conteniendo información sobre el canal de comunicación utilizado (por defecto *TCP*), un puerto y la lista de interfaces, objetos de negocios y forma de instanciarlos. Este archivo se encuentra en la carpeta *SharedSources* y compila al mismo directorio que el ejecutable del producto. Pueden utilizarse *code snippets* para agregar nuevas entradas en este archivo.

```
<configuration>
  <appSettings>
    <add key="test" value="provider=SQLOLEDB.1;
      data source=SERVEROV;database=StartFrame;
      user id=pRWPblld2UQ=;password=5SNDTwZBRQFf1URclN2xjg==" />
    <!-- <add key="spconfigpath" value="C:\Desarrollo\NET\StartFrame\Asemblies\Server\" /> -->
    <add key="serverusr" value="ADMINISTRADOR" />
    <add key="eventlog" value="true" />
  </appSettings>
  <system.runtime.remoting>
    <application name="Server">
      <channels>
        <channel ref="tcp" port="1234" />
      </channels>
      <service>
        <wellknown mode="Singleton" type="StartFrame.SqlHelper,BR" objectUri="Sql" />
        <wellknown mode="Singleton" type="StartFrame.BR.Sca,BR" objectUri="Sca" />
        <wellknown mode="Singleton" type="StartFrame.BR.Abm,BR" objectUri="Abm" />
         ...
         ...
      </service>
    </application>
  </system.runtime.remoting>
</configuration>
```

   b. **Server | SPMCServer**: Estas aplicaciones, hacen de *HOST* para los objetos de negocios. La primera de ellas funciona como una aplicación de consola, mientras que la segunda como un servicio. Puede utilizarse cualquiera de las dos. Ambas pueden ser reemplazadas por el uso de *Internet Information Server (IIS)*.

[2] Si el componente de negocios se levanta como servicio, el archivo de configuración utilizado es *SPMCServer.exe.config*

CONEXIÓN A LA BASE DE DATOS

Si bien *StartFrame* puede conectarse a cualquier *DBMS* estándar, el mismo trae un modelo de datos elemental para soportar el esquema de seguridad interno, la personalización de la interfaz y el diccionario de datos.

El motor de base de datos utilizado para la versión estándar es *SQL Server Express Edition*, por considerarse no sólo un estándar gratuito y de fácil utilización, sino que también posee una muy buena performance, potencia y escalabilidad hacia ambientes corporativos más robustos. Si desea prescindir de dicho motor empleando otro *DBMS*, deberá migrar el mencionado modelo elemental de *StartFrame* (tablas, índices y procedimientos almacenados).

> *Podrá solicitar soporte en StartFrame.Net para este tipo de conversiones. Nuestro equipo de desarrollo ha probado con éxito el producto con los siguientes motores: Oracle, PostgreSQL, MySQL.*

Para el caso de las tablas, deberá tener en cuenta que se utilizaron tipos de datos definidos por el usuario para estandarizar la estructura de las mismas, a saber:

Nombre	Tipo Básico	Longitud	Nulos	Valor Predet.
_autonumerico	Numeric	19, 0	No	
_cantidad	Numeric	12, 3	No	0
_codigo	Char	10	No	
_codigo_largo	Char	20	No	
_descripcion	Varchar	250	Si	
_entero	Int	4	No	0
_fechahora	Datetime	8	Si	GetDate()
_imagen	Image	16	Si	
_importe	Numeric	16, 6	No	0
_lista	Char	1	No	
_logico	Bit	1	No	0
_memo	Text	16	Si	
_nombre	Varchar	50	Si	
_porcentaje	Numeric	6, 2	No	0

Procedimientos Almacenados

Existen varios métodos para ejecutar procedimientos almacenados, dependiendo del tipo de resultado que el procedimiento arroje:

- Dataset
- Datareader
- Verdadero o falso
- Parámetros de salida

El tipo de dato de los parámetros de cada procedimiento se obtiene directamente del motor de base de datos.

APLICACIÓN MULTICULTURAL

Con la finalidad de poder migrar la aplicación a diferentes culturas, se tomaron ciertos recaudos con respecto a cualquier *"mensaje"* del sistema hacia la interfaz del usuario. De esta forma, existen únicamente cuatro elementos que habría que convertir si se desea migrar el sistema a otra cultura, a saber:

1. **Archivo de Recursos:** éste responde a la cultura e idioma seleccionados en la configuración regional del panel de controles de *Windows*. En dicho archivo se almacenan íconos, títulos de formularios, páginas y grillas, textos de etiquetas y botones, mensajes al usuario, *tooltips*, etc. En el caso de la arquitectura *Web App*, esta información está contenida en los *temas* con *skins y Style Sheets*.
2. **Documentación:** está formada por la documentación de ayuda en línea (proveniente de los diccionarios de datos del sistema).
3. **Menús:** por razones técnicas no se utilizó el archivo de recursos para nominar las etiquetas de los menús, sino que se utilizó un archivo XML asociado a cada menú, los cuales se encuentran en la carpeta *SharedSources.*
4. **Reportes:** por los mismos motivos que en el punto anterior, los reportes deberán convertirse por separado.

Archivo de Recursos

El archivo de recursos es un repositorio de imágenes y textos. Esto sirve para centralizar en un único lugar toda la información del tipo mencionado. La cultura utilizada en cada equipo determina el nombre del archivo de recursos. En un principio, *StartFrame* posee un archivo de recursos para la cultura de idioma español de Argentina (es-AR) y otro para la cultura neutra.

El sistema maneja dos archivos de recursos diferentes:

- **LNKfrmwrk.<cultura>.resources**: archivo de recursos utilizado por *StartFrame*, el cual se encuentra embebido dentro del componente *Common.dll*.
- **App.<cultura>.resources**: archivo de recursos utilizado por las aplicaciones desarrolladas utilizando *StartFrame*. Este archivo se encuentra en la misma ubicación que el anterior.

Es mediante estos archivos de recursos como se consigue personalizar la interfaz de *StartFrame*. Puede utilizar los mismos como base para generar su omónimo en otras culturas.

SISTEMA DE SEGURIDAD

El esquema de seguridad estándar del sistema debe analizarse desde diferentes aspectos:

1. Acceso a la base de datos
2. Conexión al sistema (logueo)
3. Uso de los componentes del sistema

Acceso a la Base de Datos

Lo primero que hay que aclarar es que existen dos tipos de usuarios diferentes:

- Usuario lógico
- Usuario físico

El primero es el usuario que está utilizando el equipo, el cual se logueó al aplicativo. Dicho usuario será validado por el sistema según se explica más adelante.

El segundo es el usuario con el cual se accederá al motor de base de datos. Se recomienda utilizar seguridad integrada al sistema operativo siempre que sea posible. Si no lo fuera, el nombre de usuario y su clave deberán encriptarse y especificarse dentro de los archivos de configuración (ver *3. Base de Datos* para mayores detalles).

En el caso de la plantilla *Web Application*, si decide utilizar la seguridad por *forms* de *ASP.NET*, será necesaria la utilización de una segunda base de datos (provista por *Microsoft*) que se utiliza para controlar la seguridad de la aplicación Web. *StartFrame* se encarga de mantener sincronizada la seguridad entre ambas bases de datos, ya que integró la base de *Microsoft* en la de *StartFrame*.

Por razones de *performance*, los parámetros de conexión se almacenan en la propiedad *ConnectionString* del componente ***Common.Env***. Esta propiedad es cargada recién cuando un usuario se loguea correctamente, al mismo tiempo que se cargan las propiedades *Operador* y *Password*. Por otro lado, la propiedad *Connection* de la clase *SQL* del componente ***DA***, retorna siempre un nuevo objeto del tipo *OLEDBconnection*.

Conexión al Sistema (logueo)

En una arquitectura *desktop*, *StartFrame* levanta el operador desde el sistema operativo y su clave de acceso desde la base de datos del sistema.

Con esos datos, instancia el método *Login* de las reglas del negocio (miembro del componente ***BR.SCA***). Este se encarga de validar al usuario dentro de las tablas de seguridad *wad_operadores* y *wld_miembros_grupo* de la base de datos del sistema.

Si todo está en orden, el acceso es permitido. En tal caso, carga las propiedades *Operador* y *Password* dentro del componente ***Common.Env***. Si por el contrario, se encuentra algún problema o inconsistencia, el acceso es denegado.

Si en lugar de utilizar el *front-end desktop* del sistema se utiliza uno diferente (web, servicio, etc.), debe llamarse al método *Login* anteriormente mencionado, pasándole como parámetros el operador que intenta conectarse al sistema y su clave de acceso. A partir de aquí, el procedimiento es igual al

anteriormente descripto. Dado que el sistema es una aplicación multicapas, las reglas de negocio (donde reside el mecanismo de seguridad) son independientes del front-end utilizado en cada caso. Por ello, es necesario que se ingrese siempre una clave de acceso para reconocer la validez de un usuario.

Utilizando BR desde otras plantillas

El siguiente estracto de código es un ejemplo de cómo instanciar las reglas de negocio de *StartFrame* en cualquier plantilla de solución.

Deberá insertar el siguiente código en el primer evento que levante la aplicación:

```
'Verifica si es un operador valido y lo registra en el Fwrk
If Not EsUnOperadorValido(sNombreOperador, sClave) Then
   MsgBox("El usuario " & sNombreOperador & _
      " no está correctamente habilitado para utilizar la aplicación.", _
      MsgBoxStyle.Critical + MsgBoxStyle.OkOnly)
End If
```

Toda la secuencia de control necesaria para validar el operador se encuentra en el siguiente método, el cual es invocado por el código anterior:

```
'Método que e encarga de loguear a un operador dentro de StartFrame.
'"NombreOperador": Operador a loguear según se haya registrado en StartFrame
'"Clave": Clave de acceso de NombreOperador según se haya registrado en StartFrame
Private Function EsUnOperadorValido(ByVal NombreOperador As String, _
                                    ByVal Clave As String) As Boolean

   Dim sNombreOperador As String = NombreOperador
   Dim sClaveAcceso As String = Clave
   'Componente de seguridad
   Dim sca As New StartFrame.BR.Sca

   Try
      'Verifica si existe el operador dentro del sistema
      If sca.ExisteOperador(sNombreOperador) Then
         'Verifica si es un usuario válido
         If sca.Login(sNombreOperador, sClaveAcceso) Then
            'Propiedades de la Common
            Common.Env.Operador = sNombreOperador
            Common.Env.Password = sClaveAcceso

            'Obtiene parametros y carga las prop. de Common
            Dim parametros As String = "EMPRESA,APPNAME,VERSION "
            Dim param As New StartFrame.BR.Utilitarios.Parametros
            Dim valores() As String = param.getParametro(parametros).Split(",c)
            Common.Env.Empresa = valores(0)
            Common.Env.AppName = valores(1)
            Common.Env.AppVersion = valores(2)

            'Logueo exitoso!
            Return True
         Else
            'Error en logueo
            Return False
         End If
      Else
         'No existe el operador
         Return False
      End If
   Catch ex As Exception
      'Error al intentar loguearse al Fwrk
      Return False
   End Try
End Function
```

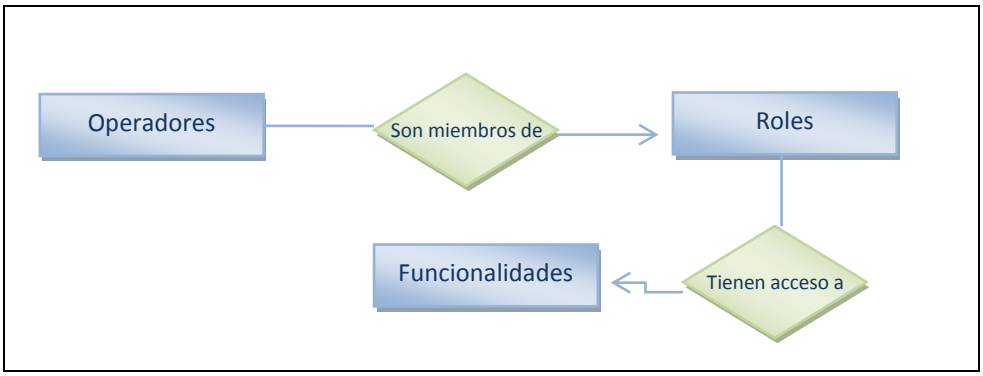
Opcionalmente, en caso de que desee loguearse a *StartFrame* utilizando un usuario genérico porque, por ejemplo, ya se validó al usuario logueado con su clave contra el modelo ASP.NET o contra el Active Directory, puede utilizarse este código para obtener la clave del mencionado usuario a pasarle al método *EsUnOperadorValido* antes demostrado.

```
'Obtiene el password y lo desencripta porque ya se autenticó en ASP.NET
sClaveAcceso = CType(Sql.Search(ConnectionString, "va_clave_acceso", _
   "wad_operadores", "cd_operador = " & _
   SystemFunctions.Strings.StringToSql(sNombreOperador)), String)
If sClaveAcceso <> "" Then
   sClaveAcceso = SystemFunctions.Strings.EncryptString(sClaveAcceso, _
      KEY, SystemFunctions.Strings.Accion.DECRYPT)
End If
```

Acceso a los Componentes

Cada vez que un cliente (*front-end*) intente instanciar un componente de las reglas de negocio, lo primero que se verificará es si está cargada la propiedad *ConnectionString* del componente ***Common.Env***. Dicha propiedad estará cargada únicamente si dicho cliente previamente se logueó al sistema utilizando *BR.SCA.Login* (ver el punto anterior para mayores detalles). La seguridad de cada componente se verifica en su constructor, por ello, si existiera algún problema de seguridad, el componente no podrá ser instanciado.

Paso seguido, se verificarán los permisos establecidos en las tablas del sistema que dicho operador tenga asignados. No se puede continuar con esta explicación sin describir el mecanismo de seguridad interno del sistema, el cual se aprecia en el siguiente esquema:



Los **operadores**, si bien tienen ciertos atributos particulares (usuario, clave, bloqueo, lleva tracking, etc.), no poseen una configuración de seguridad. La seguridad se establece a nivel de **grupo** o **rol**. La unión de los operadores con los grupos queda establecida en la tabla de **miembros** (*wld_miembros_grupo*). Cada operador puede pertenecer a varios grupos. El nivel de seguridad obtenido será el máximo posible dada la combinación de todos ellos.

A nivel de cada grupo, existen ciertos **privilegios** de uso del sistema, a saber:

- **Administrador**: puede entrar a cualquier funcionalidad del sistema, incluso a la configuración de la seguridad interna.
- **Supervisor**: puede realizar cualquier tarea que no tenga que ver con la administración del sistema en sí.

- **Modificación de datos**: puede dar altas, bajas y modificaciones de datos en cualquier objeto del sistema.
- **Consulta de información**: puede consultar información por pantalla, impresora o exportarla.

Por otro lado, existen una serie de **programas** o **clases**, que no son otra cosa que **funcionalidades** ofrecidas por el sistema en forma de clases con métodos, propiedades y eventos. Dichos programas o clases se combinan con los grupos para establecer una tabla de **accesos** (*wld_acceso_programas*). Los accesos se constituyen en la forma de **permisos**, de cuatro diferentes clases:

- **Ingreso**: tiene permitido instanciar el componente, pero no ejecutar ninguna funcionalidad del mismo salvo consultar información.
- **Alta/Ejecución**: puede ingresar nuevos registros (sólo aplicable a los abms).
- **Baja**: puede eliminar registros (sólo aplicable a los abms).
- **Modificación**: puede realizar modificaciones de registros (sólo aplicable a los abms).

Ahora bien, una vez que un usuario se registró apropiadamente en el sistema e instanció el componente deseado, se verificará si el mismo posee **privilegio** para ingresar al tipo de programa o funcionalidad que desea instanciar. Si se pasa este control, se verificará si dicho usuario tiene **permiso** para acceder al programa o funcionalidad en particular. Si este control es superado, se continúa normalmente. Caso contrario, no se permite instanciar el objeto.

Una vez instanciada exitosamente la clase (programa), cuando se intente ejecutar algún método de la misma, se verificará si el usuario tiene **permiso** para realizar dicha operación.

Auditoría

El sistema de auditoría interna consiste en la generación de un *tracking* o *pistas de auditoría* que se graban en una tabla interna tipo *log* (*wap_tracking*) de todas las altas, bajas y modificaciones de registros, ejecuciones de procesos, accesos a funcionalidades, consultas realizadas, etc., que el operador realice. En dicha tabla se graban datos tales como la fecha y hora de la operación, el operador, el nombre del programa y tabla utilizados, el tipo de operación realizada y los datos afectados.

Dado que el uso del tracking ocasiona una merma (aunque menor) en la performance del sistema y un notable incremento en el volumen de datos del mismo, existe la posibilidad de habilitar el tracking total o parcialmente, mediante los siguientes parámetros de configuración:

1. Configurando el parámetro *"TRACKING"* de la tabla *wap_parametros*, se logra habilitar o inhabilitar el trancking en forma global para todos los operadores del sistema.
2. Asignando el valor *"VERDADERO"* al campo *st_tracking* de la tabla *wad_operadores* para aquellos operadores de los cuales se desea llevar un registro de transacciones realizadas se puede decidir qué usuarios llevarán tracking y cuáles no lo harán.
3. Los objetos de negocios que lleven auditoría deberán configurar con el valor *"VERDADERO"* la propiedad *Tracking*. De esta forma se puede decidir de qué objetos se llevará un tracking de su utilización.

Cualquier cambio realizado en los parámetros anteriores toma efecto al iniciar una nueva sesión en el sistema.

Existe una funcionalidad en el módulo de utilitarios de *StartFrame* en su interfaz de administración desktop, que sirve para visualizar, imprimir y depurar el tracking de transacciones. Se recomienda consultar, imprimir, archivar y depurar esta información con cierta periodicidad.

CONTROL DE ERRORES

El mecanismo de control de errores es estándar en todas las plantillas de *StartFrame*. Los mismos son manejados como *excepciones*, de forma tal que un error no pueda pasar desapercibido, ya que si no se intercepta, la aplicación pincha.

Cada vez que se produce un error, se utiliza el método **LogError** para registrar dicho suceso en el log de errores del sistema, que no es otra cosa que la tabla *wap_log_errrores* y, opcionalmente, el *visor de sucesos* de Windows. Los datos almacenados son:

- Fecha y hora del error ocurrido
- Código interno del proceso que lo generó (suele ser el nombre del programa o el nombre de la clase/método)
- Nombre de la terminal en que se produjo el error
- Código del operador que estaba logueado cuando se produjo el error
- Código interno del error interceptado, según la codificación vigente en *wap_errores*
- Descripción adicional del error

Luego de registrar el error en el log, se propaga la excepción a la capa superior, la cual tratará al error de la misma forma antes descripta. Así se continúa hasta llegar a la capa de usuario. Esta última utilizará el método **ShowError** (en entornos desktop) o la página **GenericErrors** (en entornos Web) para informar al usuario del error ocurrido, con toda la información adicional que pudiera reunir al respecto.

ESQUEMA MULTIEMPRESA EN ENTORNOS DESKTOP

StartFrame brinda un mecanismo alternativo para soportar un manejo multiempresa en entornos desktop, el cual podrá o no ser adecuado según cada caso en particular. Para implementar el mismo, se asume que se utilizará **una base de datos independiente por cada empresa**. Por ello, cada vez que se decide cambiar de *empresa activa*, esto implica modificar la conexión a la base de datos, previo al cierre de todas las conexiones en uso.

Este sistema está diseñado para soportar el manejo multiempresa de dos diferentes formas:

1. Con procesamiento local
2. Con procesamiento distribuido

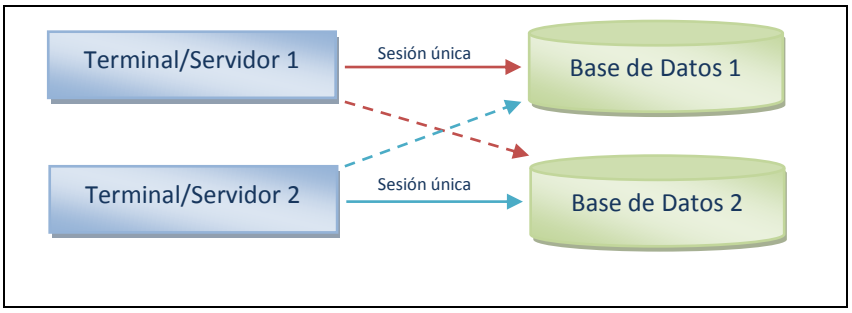
Como ya se mencionó, la arquitectura básica de la solución está formada por dos partes principales: **una aplicación cliente** (conteniendo el front-end del usuario) y **una aplicación servidora** (conteniendo las reglas del negocio y mecanismos de acceso a datos). Los dos modelos multiempresa antes mencionados difieren por la forma en que se configuran y distribuyen las mencionadas aplicaciones dentro de la red.

Esquema de Procesamiento Local

Esto significa que tanto **la aplicación cliente como la servidora, deberán residir en la terminal del cliente**. Si bien los archivos de configuración deberán apuntar a una determinada base de datos por defecto, con este esquema, el usuario está en libertad de utilizar la funcionalidad *Selectora de Empresa* ofrecida dentro del menú *Archivo* de cada módulo para seleccionar la empresa con la cual desea trabajar. Esto puede hacerlo cada vez que ingresa a un módulo, aunque todos los módulos deben apuntar a la misma empresa.

Tenga presente que cada vez que seleccione una empresa diferente, todos los módulos se cerrarán. Esto no incluye al *Selector de Módulos*, el cual permanecerá abierto. Para verificar en la aplicación cliente la empresa a la cual se encuentra conectado, puede verificar la barra de estado del formulario principal de cada módulo. Sin embargo, note que esta barra de estado se carga cada vez que se carga el módulo. Para realizar la misma verificación, pero del lado de la aplicación servidora, puede utilizar los comandos *Cnn* o *Bd*.

Existe una tabla *(wap_empresas*) en cada base de datos que es la que contiene el nombre de cada empresa y la conexión a utilizar. Esto es lo que le permitirá *navegar* entre empresas.

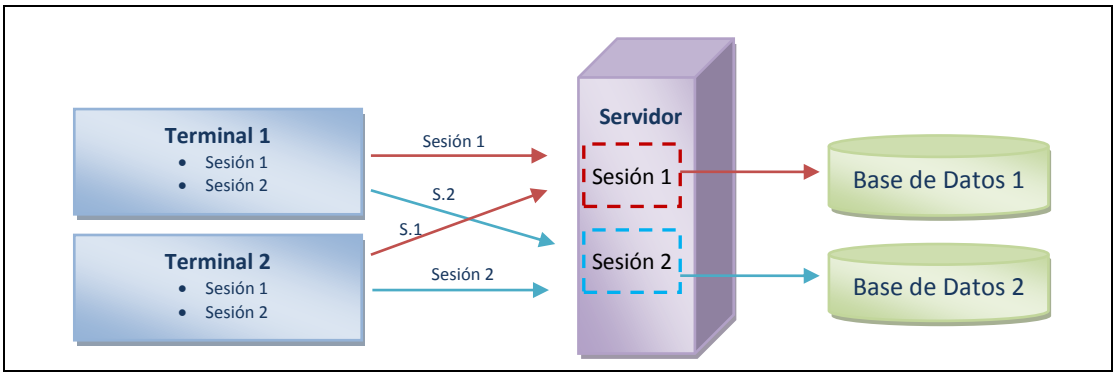


El esquema anterior ilustra cómo cada terminal del usuario (y servidor al mismo tiempo) puede tener una única sesión abierta del sistema al mismo tiempo y esta apunta a una única base de datos (empresa), pero en cualquier momento puede cambiar de base de datos.

Esquema de Procesamiento Distribuido

Esto significa que aunque la aplicación cliente resida en cada terminal, la aplicación servidora se encontrará en un determinado servidor de red. La diferencia está en que podrán existir tantas aplicaciones cliente como empresas vayan a utilizarse (quedando sin efecto el anteriormente mencionado *Selector de Empresas*), debido a que **cada aplicación cliente apuntará a una determinada base de datos y a una determinada aplicación servidora** mediante el uso de sus archivos de configuración. Es decir, que también en el servidor de red existirán tantas aplicaciones servidoras como empresas se deseen utilizar. Esto es posible debido a la modalidad de configuración de *Remoting* utilizando un puerto para cada sesión.

Para verificar en la aplicación cliente la empresa a la cual se encuentra conectado, puede visualizar la barra de estado del formulario principal de cada módulo. Para realizar la misma verificación, pero del lado de la aplicación servidora, puede utilizar los comandos *Cnn* o *Bd*.



El esquema anterior ilustra cómo cada terminal del usuario puede tener tantas sesiones del sistema como desee (según se hayan previamente instalado y configurado), cada una apuntando a una única base de datos (empresa) siempre a través de la misma aplicación servidora de reglas del negocio (aunque residan todas en un único servidor de red), sin posibilidad de modificar dicha condición.

TEST DRIVEN DEVELOPMENT (TDD)

StartFrame permite el *desarrollo orientado al testing* (TDD). De hecho, varios componentes del producto cuentan con completos test de unidad que garantizan su calidad y estabilidad ante cambios y actualizaciones de versiones.

En caso de que necesite levantar los objetos de negocios de *StartFrame* para loguearse y acceder al motor de base de datos o bien a las reglas del negocio, deberá seguir los siguientes pasos:

1. Crear un nuevo proyecto de testing para la clase o librería deseada.
2. Agregar a dicho proyecto un archivo de configuración *App.config* que contenga las siguientes etiquetas:

```
<configuration>
  <appSettings>
    <add key="test" value="provider=SQLOLEDB.1;
       data source=NombreHost;
       database=NombreBase;
       user id=pRWPblld2UQ=;
       password=5SNDTw2BRQFf1URclN2xjg==" />
    <add key="serverusr" value="ADMINISTRADOR" />
    <add key="spconfigpath" value="D:\Desarrollo\Net\StartFrame\Fuentes2010\SharedSources\" />
  </appSettings>
</configuration>
```

3. Agregar la clase *AppInit* que se encuentra dentro del proyecto *UT_DA* dentro de *Init.vb* (en realidad, basta con copiar ese archivo VB a la carpeta del proyecto e incluirlo). Esta clase tiene un método para levantar *StartFrame* utilizando el operador indicado en el *App.config*.

4. Utilizar el método provisto por el test de unidad para inicializar el entorno de testing poniendo el siguiente código que llame al procedimiento anterior:

```
    <ClassInitialize()> _
    Public Shared Sub MyClassInitialize(ByVal testContext As TestContext)
        'Instancia el Fwrk logueando a un operador genérico
        Dim init As New AppInit
        Init.Inicializar()
    End Sub
```

5. Utilizar el método provisto por el test de unidad para limpiar el entorno de testing ingresando lo siguiente:

```
<ClassCleanup()> _
    Public Shared Sub MyClassCleanup()
        Dim sca As New StartFrame.BR.Sca
        sca.Logout(Common.Env.Operador)
        sca = Nothing
    End Sub
```