



Resumen curso completo

Modelos Computacionales De Gestión Administrativa (Universidad Abierta
Interamericana)

Resumen para la primera parte de la materia: Teórico

Índice: Teoría Primer Parcial Modelos – Santiago Sábato

1

¿En qué se diferencia un sistema distribuido orientado a servicios de un sistema distribuido orientado a objetos?.....

2) ¿Cuáles son los elementos del WSDL?.....

3) ¿Cuál es la estructura de un mensaje SOAP?.....

4) ¿En qué consiste la serialización?.....

5) En este tipo de sistemas distribuidos, ¿Quién se encarga del concepto de publicación y descubrimiento?.....

6) ¿Qué tipo de información tiene un servidor UDDI?.....

7) ¿Cómo usaría los SOAPExtensions para desarrollar mecanismos de encriptación? Y, ¿Cómo lo haría a través de SOAPHeader?.....

8) ¿Cómo se relacionan los agentes y los servicios?.....

9) Grafique el modelo orientado a servicios.....

10) ¿Qué se entiende por coreografía?.....

11) ¿Cómo se explicaría el concepto de servicio relacionado con la interfaz y la semántica?.....

12) ¿Cuántos pasos implica el concepto de descubrimiento?.....

13) Describa los servicios de descubrimiento que conozca.....

14) ¿En qué consisten los servicios federados de búsqueda?.....

15) ¿Qué condiciones se deben establecer para estudiar la semántica de los sistemas orientados a servicios?.....

16) ¿Cuáles son las amenazas a nivel de mensajes?.....

17) ¿Cuáles son los requerimientos que diferencian un servicio de descubrimiento manual de uno autónomo?.....

18) ¿Qué tan confiables son los sistemas distribuidos orientados a servicios?.....

19) ¿Qué involucra la administración de un webService?.....

A) Un documento WSDL está constituido por un elemento raíz llamado y este a su vez está conformado por 5 elementos.....

B) SOAP no es un protocolo exactamente. ¿Que es, en rigor SOAP?.....

C) Dentro de los WebServices como se llama lo que define las interfaces soportadas por los servicios (Se define también en el mismo concepto las direcciones de los documentos WSDL de nuestros web services).....

D) El archivo Web.Config está relacionado con la seguridad de los webservices. Está involucrado el tag <authentication mode="windows"/> ¿Cuántos tipos de autenticación en modo windows existen? Nombres.....

E) Ejemplifica un custom validator.....

F) ¿Cuál fue el primer nombre de la especificación hoy conocida como SOAP?.....

G) En el ámbito de UDDI, ¿Que incluyen las páginas blancas?.....

H) El stack de protocolos de webservices es un conjunto de 4 capas. Nómbralas y descríbelas.....

I) Necesito usar HTTP - GET (Y no SOAP) ¿En que archivo declara que va a usar HTTP - GET? ¿En qué sección de dicho archivo y cómo?.....

J) ¿Cómo se consume un WebService en forma asíncrona? Ejemplifique mediante el código class wsTest:
system.web.services.webservice.....

K) ¿Con qué propiedad y cómo se sobrecarga un WebMethod de un WebService? Ejemplifique mediante
código.....

M) La UDDI es una especificación que permite publicar y localizar webservices. Contiene 5 componentes,
nombrarlos y describirlos.....

O) Clasificación taxonométrica de sistemas distribuidos.....

P) Ciclo de vida webservice.....

Q) Controles de validación:.....

Preguntas

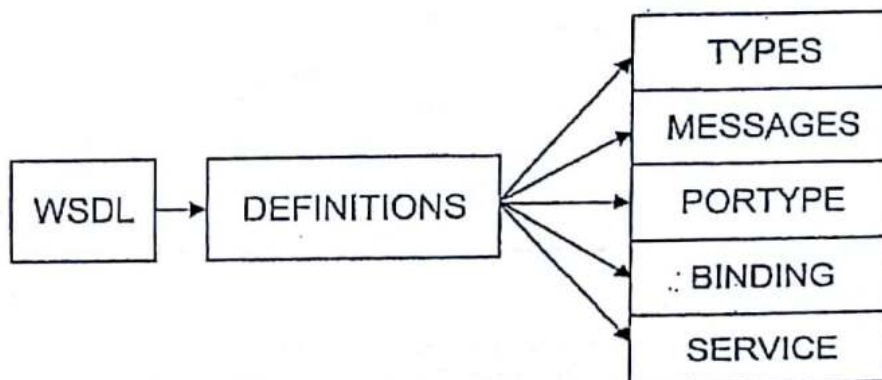
1) ¿En qué se diferencia un sistema distribuido orientado a servicios de un sistema distribuido orientado a objetos?

A comparación de un Sistema distribuido orientado a objetos, La SOA están formadas por servicios de aplicación débilmente acoplados y altamente interoperables, para comunicarse entre sí, estos servicios se basan en una definición formal independientemente de la plataforma subyacente y del lenguaje de programación

2) ¿Cuáles son los elementos del WSDL?

Los elementos del WSDL son

Types	Define el tipo de esquema a ser usado
Message	Se definen los mensajes de entrada / salida entre el cliente y el servidor. Hay varios elementos de este tipo para cada protocolo (HTTP, SOAP) y para cada WebMethod.
PortType	Define los tipos de mensajes a intercambiar entre el cliente y el servidor. Los tipos son: <ul style="list-style-type: none">- Request - Response- One-way- Solicit-Response- Notification
Binding	Se establecen en este elemento las definiciones de los vinculos de los protocolos como SOAP a un tipo de vinculo en particular.
Service	En este elemento se informa el punto de acceso a los servicios para cada uno de los protocolos a través de un elemento address.



3) ¿Cuál es la estructura de un mensaje SOAP?

SOAP está basado en xml y no se vincula con ninguna plataforma o lenguaje. No es un protocolo de comunicaciones entre mensajes, son documentos xml y requiere de otro protocolo como por ejemplo el HTTP.

La estructura de un mensaje soap está compuesta por:

ENVELOPE	Es un tag principal, comprende los datos específicos del mensaje. Se divide en header y body.
HEADER	Es una cabecera, puede o no tener relación con el mensaje en sí. Tiene elementos hijos llamados SOAPHeader
BODY	Es un cuerpo, donde están los elementos correspondientes al webmethod que queremos invocar.
FAULT	Elemento que permite indicar errores.

4) ¿En qué consiste la serialización?

La serialización es el proceso de convertir un dato binario a una representación de texto usando, por ejemplo caracteres ASCII.

Como SOAP está basado en XML y este no soporta el envío de cualquier carácter, es necesario utilizar la serialización.

5) En este tipo de sistemas distribuidos, ¿Quién se encarga del concepto de publicación y descubrimiento?

UDDI, es el servicio que permite realizar la publicación y descubrimientos de servicios.

6) ¿Qué tipo de información tiene un servidor UDDI?

Un servidor UDDI es un registro de WebService que contiene la siguiente información:

Business Entity	Información que registra la entidad de negocios y proveedores de servicios.
Contact	Se registra e-mails, teléfonos y direcciones de proveedores.
Business Service	Son los servicios que brindan los proveedores.
BindingTemplate	Información de vinculación, ubicación de los servicios, como por ejemplo la

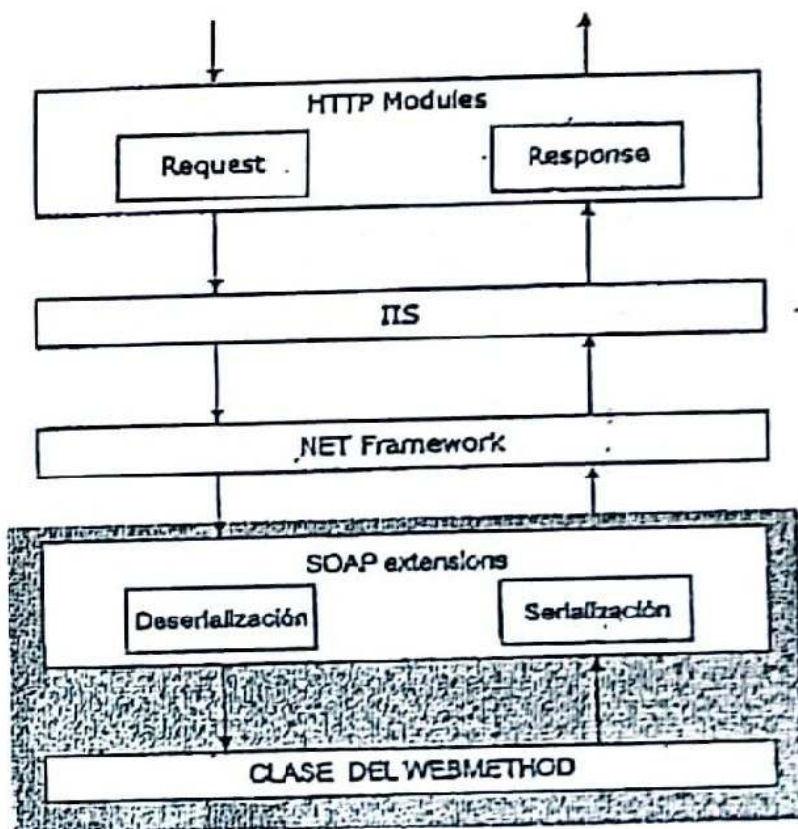
	URL de nuestro .asmx.
tModel	Define las interfaces soportadas por estos servicios. Se especifican las direcciones de los documentos WSDL de nuestros WebServices. Cada tModel debe tener una clave unica tModelKey.

7) **¿Cómo usaría los SOAPExtensions para desarrollar mecanismos de encriptación? Y, ¿Cómo lo haría a través de SOAPHeader?**

Encriptación por SoapExtensions:

Se usan para interceptar mensajes SOAP antes o después de invocar un webmethod. Permiten desarrollar mecanismos de compresión de mensajes SOAP, encriptación, caching, etc.

Para aplicarlo es necesario crear una clase que derive de la clase SOAPExtension, implementando los siguientes métodos: (ChainStram, Initialize, GetInitializer, ProcessMessage) Para usar el SOAPextention es necesario agregarlo como un nuevo atributo y debe ser configurado en la WebConfig.



Encriptación por SOAPHeaders:

Si usamos autenticación de usuarios a través de los SOAPHeaders, entonces será conveniente encriptar los mensajes SOAP. NET Framework tiene el System.Security.Cryptography que conoce los algoritmos de encriptación:

1. DES(Digital Encryption Standard)
2. SHA(Secure Hash Algorithm)
3. RSA(Rivest, Shamir, Adleman)
4. También está la posibilidad de encriptación simétrica y asimétrica

Crear una clase que derive de SoapExtension sobrescribiendo una serie de métodos abstractos.

8) ¿Cómo se relacionan los agentes y los servicios?

Se puede pensar en un webservice como una noción abstracta que debe ser implementado por un agente concreto.

El agente es una pieza concreta de software o hardware que envía y recibe mensajes, mientras que el servicio es el recurso caracterizado por un conjunto abstracto de funcionalidades.

Agente: Es un programa que actúa a favor de una persona u organización.

- Es un recurso computacional
- Tiene un propietario que es una persona u organización
- Puede realizar cero o más servicios
- Puede requerir o no de servicios

Servicio: Es un recurso abstracto que representa una capacidad de realizar tareas.

- Es un recurso
- Realiza una o mas tareas
- Tiene una descripción
- Tiene una interface
- Tiene semantica
- Tiene un identificador
- Tiene uno o mas roles
- Tiene una o mas politicas
- Puede ser propiedad de una persona u organización
- Es realizado por un agente proveedor
- Es usado por un agente demandante

Diferencias entre servicios y agentes:

- Un servicio tiene un dueño
- Debe ser realizado por un software que es un agente proveedor
- Un agente que hace un requerimiento puede interactuar con un agente proveedor.

9) Grafique el modelo orientado a servicios.

El modelo orientado a servicios hace uso de los meta-datos, que es una propiedad clave de la arquitectura orientada a servicios, y permiten documentar muchos aspectos de los servicios.

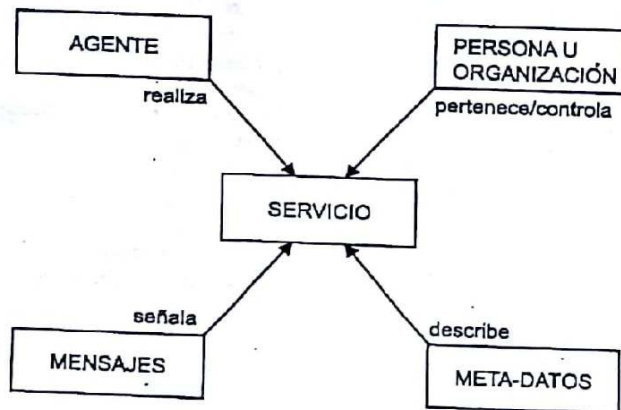
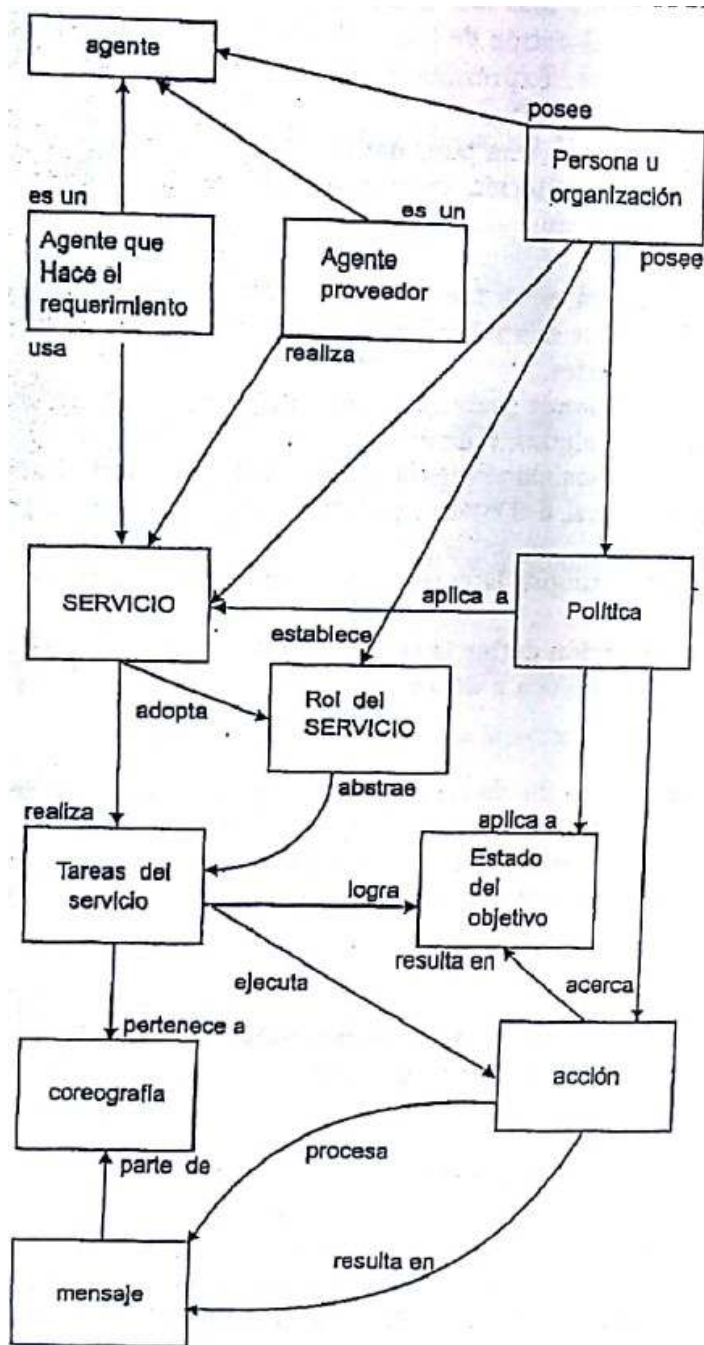


Gráfico detallado:



Este modelo se enfoca en aquellos aspectos relativos al servicio y a la acción. Su primer propósito es explicar la relación entre un agente y los servicios que provee y requiere.

10) ¿Qué se entiende por coreografía?

Define la secuencia y las condiciones debajo de la cual múltiples agentes independientes intercambian mensajes para realizar una tarea: para lograr el estado de un objetivo.

- Usa una o más interfaces de servicios

- Define el patrón de posibles interacciones entre un conjunto
- Puede ser expresada en un lenguaje de descripción de coreografía
- Pertenece a una tarea diaria
- Define la relación entre mensajes intercambiados y tareas de un servicio.

Es un modelo de secuencia de operaciones, estados y condiciones que controlan las interacciones involucradas en los servidores participantes.

11) **¿Cómo se explicaría el concepto de servicio relacionado con la interfaz y la semántica?**

Servicio:

Es un recurso abstracto que representa la capacidad de realizar tareas.

Semántica:

Es el comportamiento esperado cuando se interactúa con el servicio. Expresa un contrato entre las entidades proveedoras y demandantes.

Interfaz:

Es la abstracción que expone el servicio. Define el tipo de mensajes y el patrón de intercambio de mensajes que está involucrado en la interacción con el servicio junto con cualquier condición que impliquen estos servicios.

En resumen, una descripción del servicio es un conjunto de documentos que describen la interfaz y la semántica de un servicio.

12) **¿Cuántos pasos implica el concepto de descubrimiento?**

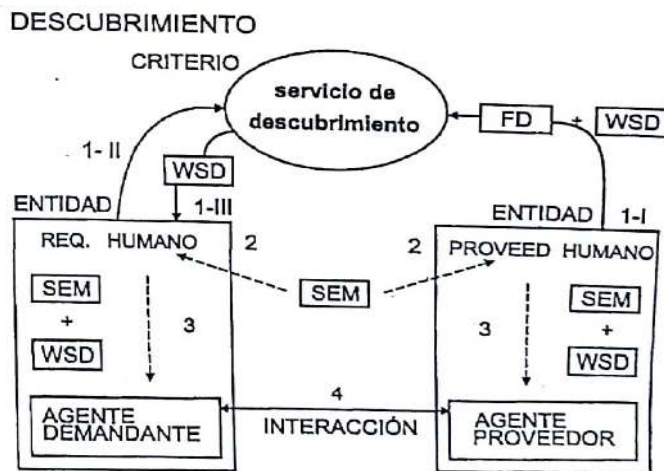
El descubrimiento es el acto de ubicar la descripción procesable por máquina, de un webservice que puede previamente haber sido desconocido y que reúne algunos criterios funcionales. El objetivo es encontrar un webservice apropiado.

El proceso de descubrimiento abarca los siguientes pasos:

- 1) El proveedor y el que hace el requerimiento “se conocen mutuamente”.
 - a. El servicio de descubrimiento de algún modo obtiene la descripción del WebService (WSD) y la descripción funcional asociada al servicio (FD).

- b. La entidad que hace el requerimiento provee criterios al servicio de descubrimiento para seleccionar una descripción de un webservice basada en su descripción funcional, sus capacidades y otras características potenciales
- c. El servicio de descubrimiento se desenvuelve una o más descripciones de webservices (o sus referencias) que reúnen un criterio especificado

- 2) El proveedor y el que hace el requerimiento acuerdan sobre la semántica de la interacción deseada.
- 3) La descripción y la semántica están embebidas apropiadamente para ambos agentes.
- 4) Ambos agentes intercambian mensajes SOAP en beneficio de ambos propietarios.



Describe los servicios de descubrimiento que conozca.

Registry	<p>Es un almacenamiento de información controlado centralmente. Cuando se publica un servicio es proveedor debe tener una participación activa:</p> <ul style="list-style-type: none"> - Decide quién tiene la autoridad de ingresar y actualizar el registro - Decide qué información es ubicada en el registro. <p>Ejemplo: UDDI</p> <p>Más apropiado en redes estáticas y controladas a donde la información no cambia frecuentemente.</p>
Índice	<p>Es la compilación o guía de información que está en todos lados.</p> <ul style="list-style-type: none"> - La publicación es pasiva - Cualquiera crea su índice - La información puede estar desactualizada - Puede incluir información de terceros - Diferentes clases de información - El mercado libre determina qué índice usará la gente para descubrir la información que busca. <p>Ejemplo: Google</p> <p>Más apropiado en situaciones que deban escalar y acomodar competencia y diversidad en estrategias de indexación.</p>
Punto a punto (P2P)	<p>Es un nodo en una red de puntos, que pueden o no ser webservices.</p>

	<ul style="list-style-type: none"> - Cualquier nodo responderá las búsquedas que reciba - cada nodo puede contener su propio índice de servicios web existentes - Los nodos se contactan entre sí de forma directa, de modo que la información que reciben esta actualizada. <p>Es más apropiado en entornos dinámicos en el cual la proximidad limita la necesidad de propagar requerimientos tales como computación ubicua.</p>
--	--

13) ¿En qué consisten los servicios federados de búsqueda?

La federación se refiere a la habilidad de consolidar los resultados de las búsquedas que involucran más que un simple índice o registro y que puedan aparecer en más que un servicio simple.

Un registro o índice puede contener información sobre otros registros u otros índices para soportar el concepto de federación.

Por ejemplo: Una búsqueda de un registro general de viajes.

Cada registro o índice puede proveer un webservice para descubrimiento, de modo que pueda ser apropiado usar lenguaje descriptivo de coreografía u orquestación para describir los cambios entre esos servicios requeridos por el concepto de federación.

14) ¿Qué condiciones se deben establecer para estudiar la semántica de los sistemas orientados a servicios?

Se debe establecer las siguientes condiciones:

- Debe haber una conexión física entre ellos.
- Debe haber concordancia sobre la forma de los datos.
- Dos o más programas deben compartir un acuerdo acerca del significado de datos.
- Debe haber un acuerdo implícito también relacionado con el procesamiento.

15) ¿Cuáles son las amenazas a nivel de mensajes?

Alteración de mensajes:	Afecta la integridad del mensaje. Un atacante puede eliminar parte del mensaje, modificarlo o insertar información extra dentro de él.
Confidencialidad	Las entidades no autorizadas obtienen acceso a información con un mensaje o parte de uno. Las partes originales pensarán que se están comunicando correctamente entre proveedor y demandante. El atacante puede tener acceso a los mensajes y modificarlos.

Spoofing	Es un ataque complejo que explota la confianza de las relaciones. El atacante asume la identidad de una entidad para sabotear la seguridad de la entidad destino.
Denegación del servicio(DoS)	Es fácil de implementar y puede causar un daño significativo. Cuando ataca efectivamente desconecta al agente del resto del sistema. Ataca el uso de recursos de más de una máquina para realizar ataques sincronizados.
Intercepción	Un intruso intercepta un mensaje y luego lo retorna al agente objetivo.

16) ¿Cuáles son los requerimientos que diferencian un servicio de descubrimiento manual de uno autónomo?

Descubrimiento manual: Un requerimiento humano usa un servicio de descubrimiento típicamente en tiempo de diseño para ubicar y seleccionar una descripción del servicio que une el funcionamiento deseado.

Descubrimiento autónomo: El agente demandante hace la tarea, ya sea en tiempo de diseño como en tiempo de ejecución.

- Requerimiento de interfaces: los requerimientos relacionados con la interacción humana son muy diferentes a los de interacción de la máquina.
- Necesidad de estandarización: las interfaces que soportan interacción humana necesitan menos estandarización.
- Confianza: la gente no confía en las máquinas para tomar decisiones que tengan consecuencias muy significativas.
- En el caso del descubrimiento autónomo la necesidad de semántica procesable por máquina, aumenta considerablemente.

Cuando el descubrimiento es manual, es un humano el que juzga confiar o no, en ese servicio. Cuando el descubrimiento es autónomo es la máquina la que toma las decisiones.

17) ¿Qué tan confiables son los sistemas distribuidos orientados a servicios?

No existe confiabilidad total en lo que respecta al servicio debido a que en el contexto de sistemas distribuidos, sobre una red pública donde los diferentes agentes son propiedad de distintos entes y están sujetos a diferentes políticas y administraciones.

Una forma de soportar confiabilidad transaccional es usar headers estandarizados.

En cuanto a la confiabilidad del mensaje, hay tres aspectos a tener en cuenta:

- ¿El mensaje que llega es el mismo que se envió?:

Esto se verifica con técnicas de conteo de bytes, firma digital, Checksums, etc.

- ¿Tiene el mensaje el formato concordado previamente?

Eso lo determinamos automáticamente mediante restricciones de sintaxis (XML bien formado) y restricciones estructurales (Validar contra una o más XMLSchema o definiciones de mensajes WSDL)

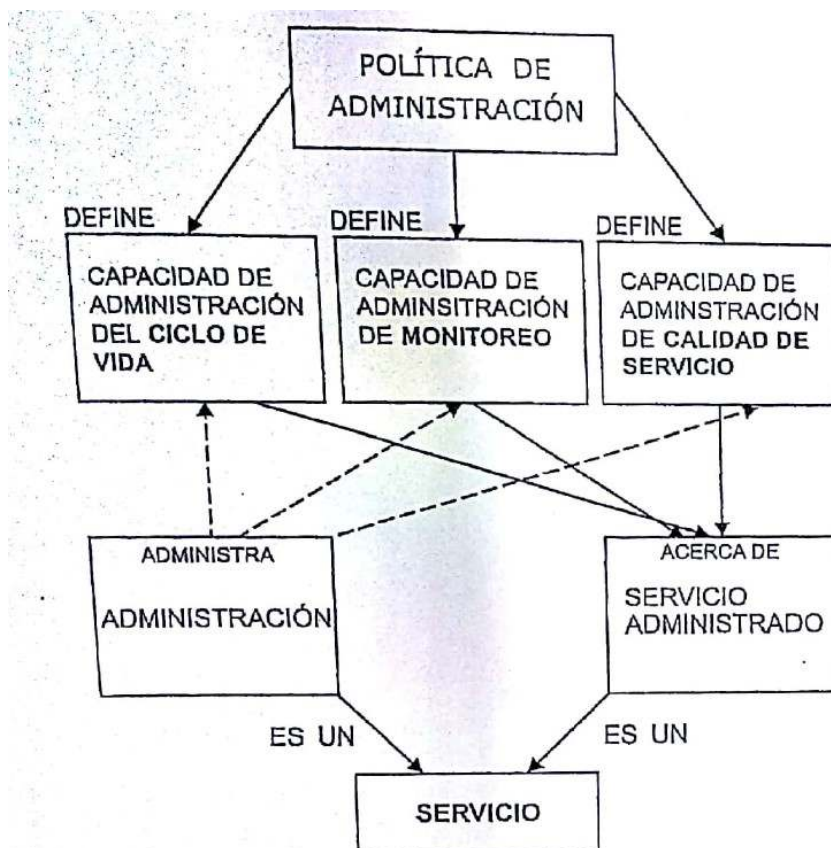
- ¿El mensaje reúne las reglas de negocio que espera el receptor del mensaje?

Para este propósito hay restricciones adicionales y chequeos de validación relacionados con el proceso de negocios típicamente monitoreado por administración de procesos lógicos y humanos.

18) ¿Qué involucra la administración de un webService?

Involucra control y reportes de calidad y uso de servicio.

- La calidad incluye disponibilidad.
- Los reportes están relacionados con un modelo de información de administración que típicamente contiene conteo de requerimientos y respuestas, timers de comienzo y final, estado de ciclo de vida, identificadores de identidad.



Preguntas extras profesor:

A) Un documento WSDL está constituido por un elemento raíz llamado... y este a su vez está conformado por 5 elementos. Nómbralos.

Un documento WSDL está compuesto por un elemento raíz llamado *Definitions* que a su vez está compuesto por los siguientes elementos:

- *Types*
- *Messages*
- *PortType*
- *Binding*
- *Service*

B) SOAP no es un protocolo exactamente. ¿Que es, en rigor SOAP?

SOAP es una plataforma independiente y por consecuencia permite la comunicación entre aplicaciones diversas.

SOAP está basado en xml y no se vincula con ninguna plataforma o lenguaje. No es un protocolo de comunicaciones entre mensajes, son documentos xml y requiere de otro protocolo como por ejemplo el HTTP.

C) Dentro de los WebServices como se llama lo que define las interfaces soportadas por los servicios (Se define tambien en el mismo concepto las direcciones de los documentos WDSL de nuestros web services)

tmodel: Es información que se encuentra en el servidor UDDI. Define las interfaces soportadas por estos servicios. Se especifican las direcciones de los documentos WSDL de nuestros WebServices. Cada tModel debe tener una clave unica tModelKey.

Stack: etapa 3: Descripción del Servicio: usado para describir la interfaz pública de un Servicio Web específico. El formato de interfaz Web Services Description Language - WSDL es típicamente usado para este propósito.

D) El archivo Web.Config está relacionado con la seguridad de los webservice. Está involucrado el tag <authentication mode="windows"/> ¿Cuántos tipos de autenticación en modo windows existen? Nómbralos.

Implícita: Técnica de hashing para enviar para enviar credenciales de usuarios de manera encriptada. (no es soportada en todas las plataformas).

Básica: envía los datos del usuario y contraseña en forma de texto codificado en base 64 no encriptado. (no es segura).

integrada: usa algoritmo de encriptación kerberos o NTLM. Toma los datos del usuario actual autenticado por windows.

E) Ejemplifica un custom validator.

Un Custom Validator permite que el programador pueda crear sus propios controles de validación. Realiza la validación definida por el usuario en un control de entrada.

Propiedades:

- ClientValidationFunction
- OnServerValidate

Un ejemplo podría ser un custom validator para determinar si un numero es par o no.

Control:

```
<asp:CustomValidator
    ID="CustomValidator1"
    runat="server"
    ErrorMessage="CLIENTE: No es numero par!"
    ClientValidationFunction="ClientValidate"
    ControlToValidate="txtNumero"
    Display="Dynamic"
    ForeColor="Red"
    ValidateEmptyText="True">
</asp:CustomValidator>
```

Función en javascript:

```
<script type="text/javascript">

function ClientValidate(source, arguments)
{

    if ((arguments.Value % 2) == 0)
        arguments.IsValid = true;
```



```

        else
            arguments.IsValid = false;
    }

</script>

```

Código en el botón validar:

```

if (Page.IsValid)
{
    lblOutput.Text = "Página válida!";
    lblOutput.ForeColor = System.Drawing.Color.Green;
}

```

F) ¿Cuál fue el primer nombre de la especificación hoy conocida como SOAP?

XML-RPC es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

G) En el ámbito de UDDI, ¿Qué incluyen las páginas blancas?

Páginas blancas - dirección, contacto y otros identificadores conocidos

H) El stack de protocolos de webservices es un conjunto de 4 capas. Nómbralas y descríbelas.

La Pila de Protocolos para servicios está comprendida principalmente por cuatro áreas:

- Servicio de Transporte: responsable del transporte de mensajes entre las Aplicaciones de red y los protocolos en los cuales se incluyen protocolos tales como HTTP, SMTP, FTP, así como también el más reciente Blocks Extensible Exchange Protocol (BEEP).
- Mensajería XML: responsable por la codificación de mensajes en un formato común XML así que ellos puedan ser entendidos en cualquier extremo de una conexión de red. Actualmente, esta área incluye protocolos tales como XML-RPC, SOAP y REST.
- Descripción del Servicio: usado para describir la interfaz pública de un Servicio Web específico. El formato de interfaz Web Services Description Language - WSDL es típicamente usado para este propósito.

- Descubrimiento de servicios: centraliza servicios en un registro común tal que los servicios Web de la red puedan publicar su localización y descripción, y hace que sea fácil descubrir que servicios están disponibles en la red. Actualmente, la API Universal Description Discovery and Integration - UDDI se utiliza normalmente para el descubrimiento de servicios.

I) Necesito usar HTTP - GET (Y no SOAP) ¿En que archivo declara que va a usar HTTP - GET? ¿En qué sección de dicho archivo y cómo?

Se declara en el archivo Web.config. Dentro del tag <protocols>.

```
<configuration>
  <system.web>
    <webServices>
      <protocols>
        <add name="HttpGet"/>
        <add name="HttpPost"/>
      </protocols>
    </webServices>
  </system.web>
</configuration>
```

J) ¿Cómo se consume un WebService en forma asíncrona? Ejemplifique mediante el código class wsTest: system.web.services.webservice

El método asincrónico procesa la llamada en el servidor sin esperar la respuesta, esto hace que la interfaz de usuario nunca se congele y pueda seguir utilizando la aplicación, inclusive si existe una excepción (tratado, claro) seguirán realizándose las tareas pendientes.

```
private async void Button_Click_2(object sender, RoutedEventArgs e)
{
    var cliente = new Servicio.WebService1SoapClient();
    Servicio.HolaMundoResponse response = await cliente.HolaMundoAsync();
    MessageBox.Show(response.Body.HolaMundoResult);
}
```

K) ¿Con qué propiedad y cómo se sobrecarga un WebMethod de un WebService? Ejemplifique mediante código.

Para sobrecargar un método de un webservice es necesario lo siguiente:

- 1) Se debe cambiar el tempuri.org por la URL donde instalemos o publiquemos nuestro web service:

```
[WebService(Namespace = "http://tempuri.org/")]
```

- 2) Declaro el método en el webservice, mediante la propiedad webmethod

```
[WebMethod]
public double Sumar(double a, double b)
{
    return a + b;
}
```

- 3) Instanciar el webservice, para ello debe haberse realizado la referencia se servicio correspondiente.

```
WebService WebSer = new WebService();
```

- 4) Luego, sobrecargo el método

```
resultadoSuma = WebSer.Sumar(a, b);
```

```
[WebMethod(MessageName = ".....")]
```

Es una propiedad de tipo string que nos permite darle un único nombre a los métodos que tengamos sobrecargados.

M) La UDDI es una especificación que permite publicar y localizar webservices. Contiene 5 componentes, nombrarlos y describirlos.

Un servidor UDDI es un registro de WebService que contiene la siguiente información:

<i>Business Entity</i>	Información que registra la entidad de negocios y proveedores de servicios.
<i>Contact</i>	Se registra e-mails, teléfonos y direcciones de proveedores.

Business Service	Son los servicios que brindan los proveedores.
BindingTemplate	Información de vinculación, ubicación de los servicios, como por ejemplo la URL de nuestro .asmx.
tModel	Define las interfaces soportadas por estos servicios. Se especifican las direcciones de los documentos WSDL de nuestros WebServices. Cada tModel debe tener una clave unica tModelKey.

O) Clasificación taxonomica de sistemas distribuidos

Los sistemas distribuidos se clasifican en:

- 1) De computo
 - a) Cluster
 - b) Grid
- 2) De información
 - a) RPTTransaccional
 - b) RMI-Jove

O también se clasifican en:

- 1) Basados en servicios
- 2) Basados en coordinación:
 - Los mensajes no contienen la dirección de su receptor
 - Son direccionados por tema - Middleware MDM.

P) Ciclo de vida webservice



Q) Controles de validación:

Se instancian tal como cualquier control.

RequestFieldValidator: comprueba que se haya ingresado un valor

RegularExpressionValidator: que el valor ingresado haga un determinado formato.
(Por ejemplo fecha)

RangeValidator: que el valor ingresado esté comprendido en un determinado intervalo

Las propiedades son:

a) Type

- Integer
- double
- string
- date
- currency

b) MinimumValue

c) Maximumvalue

CompareValidator: (*) que el valor ingresado sea aceptable comparándolo con otro valor o con el valor de otro control

Las propiedades son:

- ControlCompare
- Operator
 - Equal
 - NotEqual
 - GreaterThan
 - LessThan

CustomValidator ():** Permite que el programador pueda crear sus propios controles de validación.

Las propiedades son

- ClientValidationFunction
- OnServerValidate

¿Qué es IsValid?

Es una propiedad booleana que en caso de ser verdadera significa que se han superado todos los controles de validación.

Contenido

1.	Descripción General del Trabajo Práctico.....	3
1.1	Propósito.....	3
1.2	Flujo de la Aplicación.....	3
1.3	Requisitos que evalúa el WS.....	3
1.4	Lenguaje de programación utilizado.....	3
2.	IDE de Desarrollo.....	4
2.1	DetalleAlumnos.aspx.....	4
2.2	ComprobarRequisitos.aspx.....	6
2.3	WsUniversidad.asmx.....	6
2.4	Global.asax.....	7
2.5	Web.Config.....	8
2.6	Referencias WS.....	9
3.	Pantallas del sistema.....	11
3.1	Inicio (DetalleAlumnos.aspx).....	11
3.1.1	Controles de validación.....	11
3.2	Respuesta (ComprobarRequisitos.aspx).....	13
4.	Código fuente.....	14
4.1	DetalleAlumnos.aspx.cs.....	14
4.2	ComprobarRequisitos.aspx.cs.....	16
5.	Secuencia de Ejecución.....	18
	1er Paso: Carga de Formulario.....	18
	2do Paso: Se envían los datos.....	18
	3er Paso: Se validan los datos.....	19

1. Descripción General del Trabajo Práctico

1.1 Propósito

Este práctico tiene como objetivo comprender los aspectos básicos de la programación en ASP.NET y manejo del Web service, abordando temas de envío de información mediante cookies, QueryString, variables de Session, ViewState, Validadores como RequiredFieldValidator, CustomValidator entre otros.

1.2 Flujo de la Aplicación

Nuestra aplicación simula el ingreso de datos de alumnos con el fin de comprobar si los requisitos son válidos para el ingreso a dicha universidad.

Apellido: Valor requerido

Edad: "Valor requerido + Valor comprendido entre 17 y 99 años"

Cantidad de Materias Aprobadas: Valor requerido

Lenguajes: "C#"

Especialidad: "Software – Redes – Minería de datos"

¿Está Becado? Si / no

1.3 Requisitos que evalúa el WS

- a) Materias aprobadas mayor a 30
- b) Edad menor a 40 años
- c) Especialización elegida "Software"
- d) Nacionalidad elegida "Argentina"
- e) Algún lenguaje elegido: C#

Nota: Si el alumno es becado es válido para el ingreso sin importar ninguna otra condición.

1.4 Lenguaje de programación utilizado

Los lenguajes de programación utilizados son:

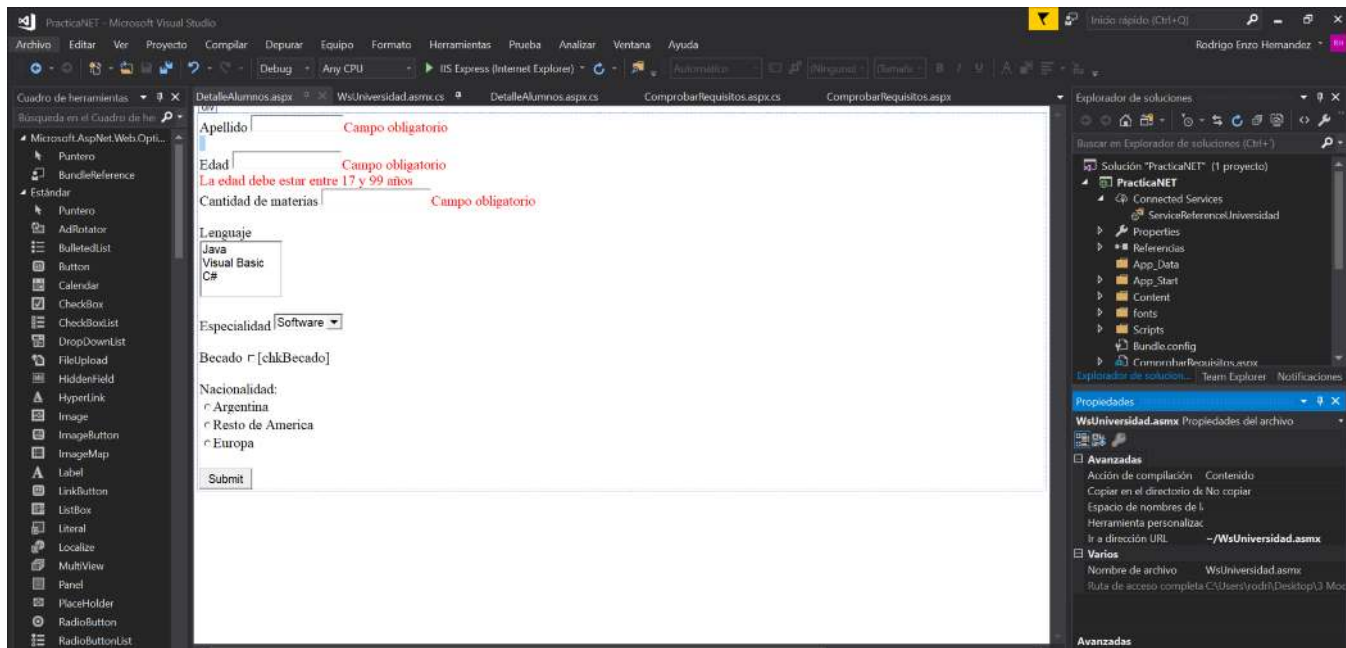
ASP.NET: Interfaz gráfica.

C#: Funcionalidad.

2. IDE de Desarrollo

El IDE de desarrollo utilizado es el de Visual Studio. Cuenta con:

- Cuadro de herramientas.
- Explorador de soluciones.
- Interfaz.



2.1 DetalleAlumnos.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DetalleAlumnos.aspx.cs"
Inherits="PracticaNET.DetalleAlumnos" %>
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
```

```
    <form id="form1" runat="server">
        <div>
            <%-- Paso 1) Ingreso de datos --%>
            Apellido
            <asp:TextBox ID="txtApellido" runat="server"
Width="151px"></asp:TextBox>
            <%-- Paso 1.1) Se valida que el campo tenga datos mediante un
RequiredFieldValidator --%>
```



```

        <asp:RequiredFieldValidator ID="rfvApellido" runat="server"
ControlToValidate="txtApellido" ErrorMessage="Campo obligatorio"
ForeColor="Red"></asp:RequiredFieldValidator>
    <br />
    <br />
    Edad <asp:TextBox ID="txtEdad" runat="server"
OnTextChanged="txtEdad_TextChanged" style="height: 25px"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="txtEdad" Display="Dynamic" ErrorMessage="Campo
obligatorio" ForeColor="Red"></asp:RequiredFieldValidator>
    <br />
    <%-- El campo edad se consiste mediante un Custom Validator con una
función en el servidor: "ComprobarRequisitos" --%>
    <asp:CustomValidator ID="cvEdad" runat="server"
ControlToValidate="txtEdad" Display="Dynamic" ErrorMessage="La edad debe estar
entre 17 y 99 años" ForeColor="Red" ValidateEmptyText="True"
OnServerValidate="cvEdad_ServerValidate"></asp:CustomValidator>
    <br />
    Cantidad de materias <asp:TextBox ID="txtMaterias" runat="server"
OnTextChanged="txtMaterias_TextChanged"></asp:TextBox>
    <%-- Paso 1.2) Se valida que el campo tenga datos mediante un
RequiredFieldValidator --%>
    <asp:RequiredFieldValidator ID="rfvMaterias" runat="server"
ControlToValidate="txtMaterias" ErrorMessage="Campo obligatorio"
ForeColor="Red"></asp:RequiredFieldValidator>
    <br />
    <br />
    Lenguaje <br />
    <asp:ListBox ID="lstLenguajes" runat="server"
SelectionMode="Multiple">
        <asp:ListItem>Java</asp:ListItem>
        <asp:ListItem>Visual Basic</asp:ListItem>
        <asp:ListItem>C#</asp:ListItem>
    </asp:ListBox>
    <br />
    <br />
    Especialidad <asp:DropDownList ID="ddlEspecialidad" runat="server">
        <asp:ListItem>Software</asp:ListItem>
        <asp:ListItem>Funcional</asp:ListItem>
        <asp:ListItem>QA</asp:ListItem>
    </asp:DropDownList>
    <br />
    <br />
    Becado
    <asp:CheckBox ID="chkBecado" runat="server" />
    <br />
    <br />
    Nacionalidad:
    <asp:RadioButtonList ID="rdlstNacionalidad" runat="server">
        <asp:ListItem>Argentina</asp:ListItem>
        <asp:ListItem>Resto de America</asp:ListItem>
        <asp:ListItem>Europa</asp:ListItem>
    </asp:RadioButtonList>
    <br />
    <%-- Boton que envía los datos al servidor --%>
    <asp:Button ID="btnEnviar" runat="server" OnClick="btnEnviar_Click"
Text="Submit" />
    <br />
</div>
</form>
</body>
</html>

```

2.2 ComprobarRequisitos.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="ComprobarRequisitos.aspx.cs"
Inherits="PracticaNET.ComprobarRequisitos" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
</div>
</form>
</body>
</html>
```

2.3 WsUniversidad.asmx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace PracticaNET
{
    /// <summary>
    /// Descripción breve de WsUniversidad
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // Para permitir que se llame a este servicio web desde un
    script, usando ASP.NET AJAX, quite la marca de comentario de la
    línea siguiente.
    // [System.Web.Script.Services.ScriptService]
    public class WsUniversidad : System.Web.Services.WebService
    {
        // Paso 10) Se validan los requerimientos con el Primer
        WebMethod
        // Primer WebMethod: Permite validar los requerimientos
        necesarios para que el alumno ingresado sea apto.
        [WebMethod]
        public bool ValidarRequerimientos(bool becado, int edad,
        int materiasAprobadas, string lenguajes, string especializacion,
        string nacionalidad)
        {
```

```

        bool tieneCsharp = false;

        //Si la lista contiene el elemento C#,
        (Seleccionado de la lista), pone en true la variable bool
        "tieneCsharp"
        if (lenguajes.Contains("C#"))
        {
            tieneCsharp = true;
        }

        if (!becado)
        {
            //Validar la condición del return
            // Condiciones
            // a) Materias aprobadas mayor a 30
            // b) Edad menor a 40 años
            // c) Especialización elegida "Software"
            // d) Nacionalidad elegida "Argentina"
            return (tieneCsharp && materiasAprobadas > 30 &&
edad < 40 && especializacion == "Software" && nacionalidad ==
"Argentina");
        }

        //Cumple las condiciones (Porque es becado)
        return true;
    }

    // Paso 2.2) Se consiste el campo edad
    // Segundo WebMethod: Permite consistir el campo edad (Edad
mayor a 17 y menor a 99)
    [WebMethod]
    public bool _ConsistirEdad(int edad)
    {
        if(edad > 17 && edad < 99)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
}

```

2.4 Global.asax

El archivo Global.asax se encuentra en el directorio raíz del proyecto ASP.NET y permite realizar configuraciones globales a nivel aplicación y sesión.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Optimization;
using System.Web.Routing;
using System.Web.Security;
using System.Web.SessionState;

namespace PracticaNET
{
    public class Global : HttpApplication
    {
        void Application_Start(object sender, EventArgs e)
        {
            // Código que se ejecuta al iniciar la aplicación
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}

```

2.5 Web.Config

El archivo Web.Config es un archivo con formato XML que permite almacenar valores que controlan el funcionamiento del sitio web. Un ejemplo útil sería configurar el String de conexión.

```

<?xml version="1.0" encoding="utf-8"?>
<!--
    Para obtener más información sobre cómo configurar la aplicación ASP.NET,
    visite
    https://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.5"/>
    <httpRuntime targetFramework="4.5"/>
    <pages>
      <namespaces>
        <add namespace="System.Web.Optimization"/>
      </namespaces>
      <controls>
        <add assembly="Microsoft.AspNet.Web.Optimization.WebForms"
namespace="Microsoft.AspNet.Web.Optimization.WebForms" tagPrefix="webopt"/>
      </controls>
    </pages>
  </system.web>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Antlr3.Runtime"
publicKeyToken="eb42632606e9261f"/>
        <bindingRedirect oldVersion="0.0.0.0-3.5.0.2" newVersion="3.5.0.2"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>

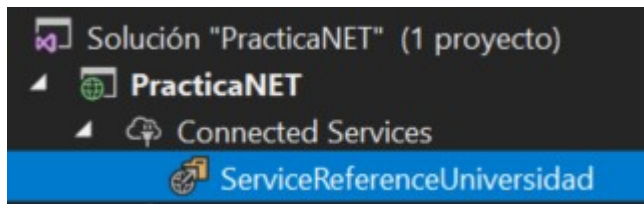
```

```

        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="Newtonsoft.Json"
publicKeyToken="30ad4fe6b2a6aeed"/>
            <bindingRedirect oldVersion="0.0.0.0-11.0.0.0" newVersion="11.0.0.0"/>
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Diagnostics.DiagnosticSource"
publicKeyToken="cc7b13ffcd2ddd51"/>
            <bindingRedirect oldVersion="0.0.0.0-4.0.2.1" newVersion="4.0.2.1"/>
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35"/>
            <bindingRedirect oldVersion="0.0.0.0-1.6.5135.21930"
newVersion="1.6.5135.21930"/>
        </dependentAssembly>
    </assemblyBinding>
</runtime>
<system.codedom>
    <compilers>
        <compiler language="c#;cs;csharp" extension=".cs"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
warningLevel="4" compilerOptions="/langversion:6
/nowarn:1659;1699;1701"/>
        <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
warningLevel="4" compilerOptions="/langversion:14 /nowarn:41008
/define:_MYTYPE=&quot;Web&quot; /optionInfer+"/>
    </compilers>
</system.codedom>
<system.serviceModel>
    <bindings>
        <basicHttpBinding>
            <binding name="WsUniversidadSoap" />
        </basicHttpBinding>
    </bindings>
    <client>
        <endpoint address="http://localhost:55050/WsUniversidad.asmx"
binding="basicHttpBinding" bindingConfiguration="WsUniversidadSoap"
contract="ServiceReferenceUniversidad.WsUniversidadSoap"
name="WsUniversidadSoap" />
    </client>
</system.serviceModel>
</configuration>

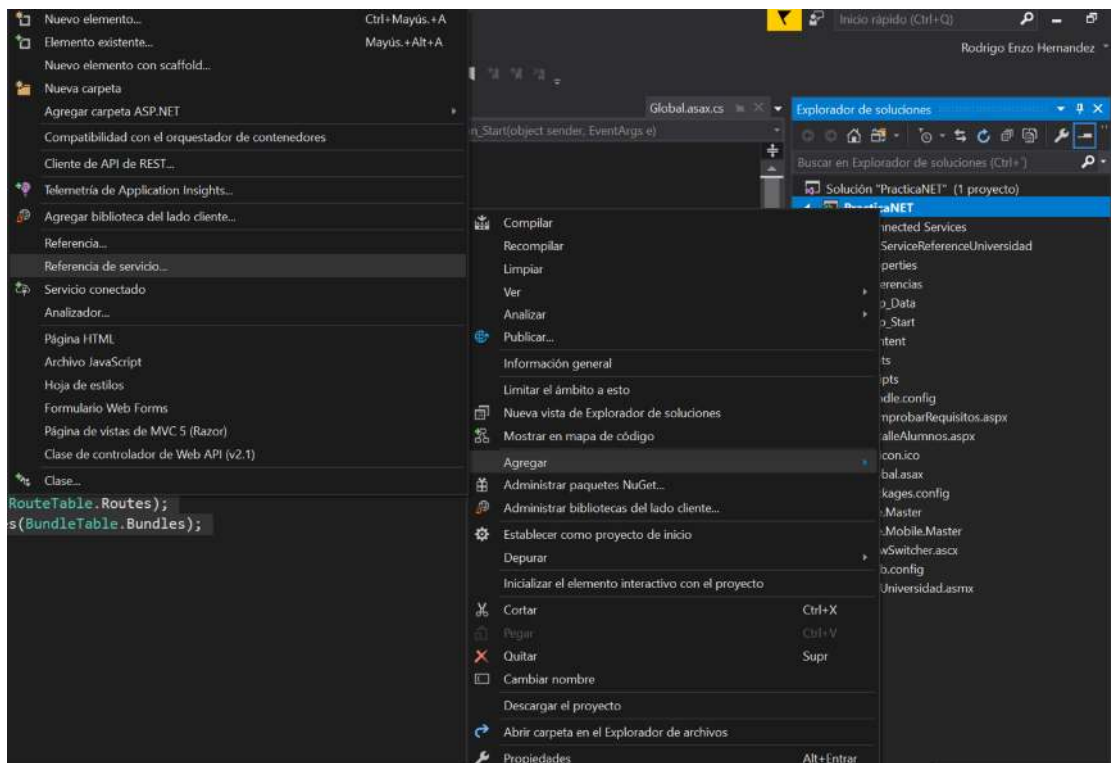
```

2.6 Referencias WS

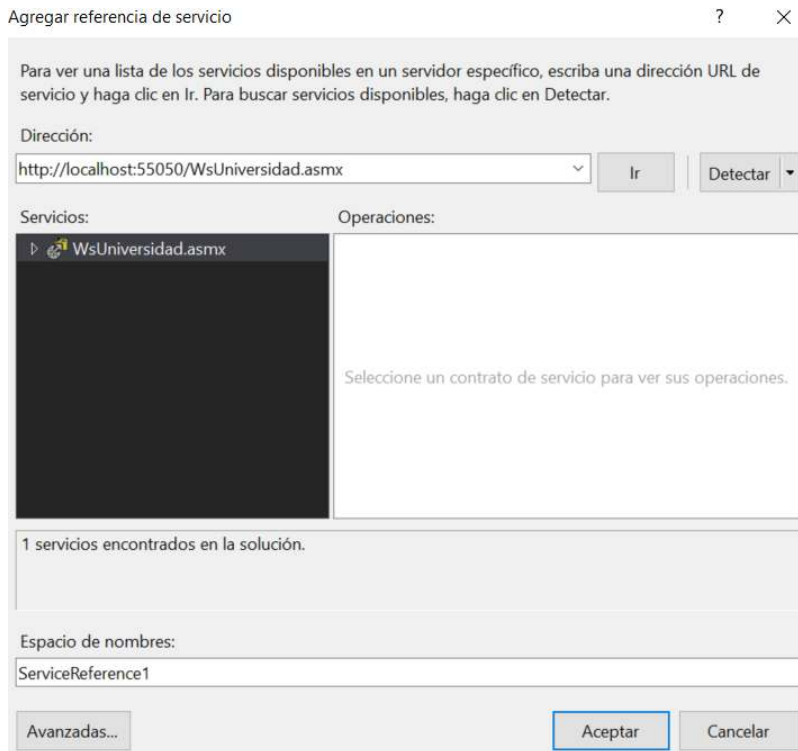


Es necesario crear una referencia con el objetivo de poder comunicarse con el Webservice y poder utilizar sus WebMethods.

Para agregar la referencia es necesario hacer clic derecho sobre el proyecto – agregar – Referencia de servicio. Se puede observar en la siguiente captura:



Luego debemos detectar el webservice y asignarle un nombre a la referencia:



3. Pantallas del sistema

3.1 Inicio (DetalleAlumnos.aspx)

3.1.1 Controles de validación

Control de validación	Campo validado
RequiredFieldValidator	txtApellido
RequiredFieldValidator	txtEdad
RequiredFieldValidator	txtMateria
CustomValidator	txtEdad

```
ComprobarRequisitos.aspx DetalleAlumnos.aspx* X
<equiv="Content-Type" content="text/html; charset=utf-8"/>
</title>

<form1 runat="server">
<!-- Paso 1) Ingreso de datos -->
Apellido
<asp:TextBox ID="txtApellido" runat="server" Width="151px"></asp:TextBox>
<!-- Paso 1.1) Se valida que el campo tenga datos mediante un RequiredFieldValidator -->
<asp:RequiredFieldValidator ID="rfvApellido" runat="server" ControlToValidate="txtApellido" ErrorMessage
<br />
<br />
Edad <asp:TextBox ID="txtEdad" runat="server" OnTextChanged="txtEdad_TextChanged" style="height: 25px">
<!-- Se valida que el campo tenga datos mediante un RequiredFieldValidator -->
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="txtEdad" Disp
<br />
<!-- El campo edad se consiste mediante un Custom Validator con una función en el servidor: "ComprobarRe
<asp:CustomValidator ID="cvEdad" runat="server" ControlToValidate="txtEdad" Display="Dynamic" ErrorMessa
<br />
Cantidad de materias <asp:TextBox ID="txtMaterias" runat="server" OnTextChanged="txtMaterias_TextChanged
<!-- Paso 1.2) Se valida que el campo tenga datos mediante un RequiredFieldValidator -->
<asp:RequiredFieldValidator ID="rfvMaterias" runat="server" ControlToValidate="txtMaterias" ErrorMessage
```

La validación se notifica al usuario de la siguiente forma:

Validación de campos vacíos:

http://localhost:55050/DetalleAlumnos

localhost x

Galería de Web Slice ▼

Apellido Campo obligatorio

Edad Campo obligatorio

Cantidad de materias Campo obligatorio

Validación rango de edad con un CustomValidator:

http://localhost:55050/DetalleAlumnos

localhost x

Galería de Web Slice ▼

Apellido

Edad La edad tiene que ser entre 17 y 99 años

Cantidad de materias

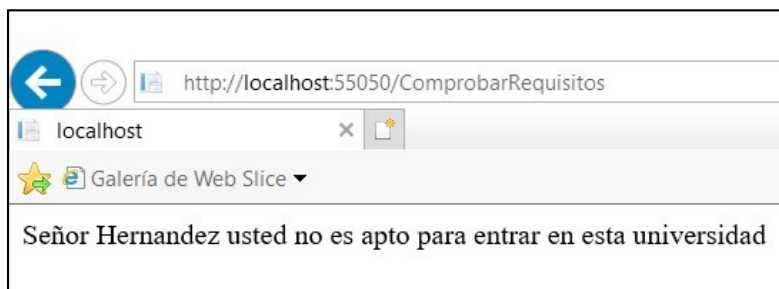
3.2 Respuesta (ComprobarRequisitos.aspx)

Permite indicarle al usuario si es apto para el ingreso a la universidad o no.

Usuario apto:



Usuario no apto:



4. Código fuente

4.1 DetalleAlumnos.aspx.cs

```
3. using System;
4. using System.Collections.Generic;
5. using System.Linq;
6. using System.Web;
7. using System.Web.UI;
8. using System.Web.UI.WebControls;
9.
10. namespace PracticaNET
11. {
12.     public partial class DetalleAlumnos : System.Web.UI.Page
13.     {
14.
15.         protected void btnEnviar_Click(object sender,
16.             EventArgs e)
17.         {
18.
19.             // Paso 3) Se guardan los datos en cookies
20.             Response.Cookies["Alumno"]["Apellido"] =
21.                 txtApellido.Text;
21.             Response.Cookies["Alumno"]["Edad"] =
22.                 txtEdad.Text;
22. }
```

```

23.         // Paso 4) Guardamos en la variable
"listaLenguajes", una concatenación de los lenguajes
seleccionados usando la función "ObtenerSeleccionados()"
24.         string listaLenguajes = ObtenerSeleccionados();
25.         Response.Cookies["Alumno"]["ListaLenguajes"] =
listaLenguajes;
26.         Response.Cookies["Alumno"]["CantidadMaterias"] =
txtMaterias.Text;
27.         Response.Cookies["Alumno"]["Especialidad"] =
ddlEspecialidad.SelectedItem.Text;
28.         Response.Cookies["Alumno"]["Becado"] =
chkBecado.Checked.ToString();
29.         Response.Cookies["Alumno"]["Nacionalidad"] =
rdlstNacionalidad.SelectedItem.Text;
30.
31.         // Paso 5) Se comprueba si la página es válida
32.         if (Page.IsValid)
33.         {
34.
35.             // Paso 6) Si la página es válida se
redirige al servidor "ComprobarRequisitos"
36.             Response.Redirect("ComprobarRequisitos.aspx");
37.         }
38.         else
39.         {
40.             // Paso 6.1) Si la página no es valida, se
muestra un mensaje de error.
41.             cvEdad.ErrorMessage = "La edad tiene que ser
entre 17 y 99 años";
42.         }
43.
44.
45.     }
46.
47.     // Función que permite concatenar todos los
lenguajes de programación seleccionados
48.     private string ObtenerSeleccionados()
49.     {
50.
51.         string lista = "";
52.         foreach (ListItem item in lstLenguajes.Items)
53.         {
54.             if (item.Selected)
55.             {
56.                 lista += item.Text;
57.             }
58.         }
59.         return lista;
60.
61.     }
62.
63.     // Paso 2) Consistir el campo Edad utilizando un
Custom Validator (Edad Mayor a 17 y menor a 99)
64.     protected void cvEdad_ServerValidate(object source,
ServerValidateEventArgs args)
65.     {
66.

```

```

67.             int edad = Convert.ToInt32(args.Value);
68.             bool ConsistirEdad =
    ComprobarRequisitos._ConsistirEdad(edad); //Llamamos a la
    función _ConsistirEdad que se encuentra en el servidor
69.
70.             if (ConsistirEdad)
71.             {
72.                 args.IsValid = true;
73.             }
74.             else
75.             {
76.                 args.IsValid = false;
77.             }
78.
79.         }
80.
81.         protected void txtEdad_TextChanged(object sender,
    EventArgs e)
82.         {
83.
84.         }
85.
86.         protected void txtMaterias_TextChanged(object
    sender, EventArgs e)
87.         {
88.
89.         }
90.     }
91. }

```

4.2 ComprobarRequisitos.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace PracticaNET
{
    public partial class ComprobarRequisitos : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Paso 7) Se declaran variables locales y se recuperan
            los datos provenientes del cliente "DetalleAlumnos.aspx.cs"
            #region Recuperando los valores de la otra pagina

            string apellido = Request.Cookies["Alumno"]
["Apellido"];
            int edad = Convert.ToInt32(Request.Cookies["Alumno"]
["Edad"]);
            int cantidadMaterias =
Convert.ToInt32(Request.Cookies["Alumno"]["CantidadMaterias"]);
            string listaLenguajes = Request.Cookies["Alumno"]
["ListaLenguajes"];

```

```

        string especialidad = Request.Cookies["Alumno"]
["Especialidad"].ToString();
        bool becado =
Convert.ToBoolean(Request.Cookies["Alumno"]["Becado"]);
        string nacionalidad = Request.Cookies["Alumno"]
["Nacionalidad"].ToString();

        #endregion

        // Paso 8) Se instancia el "WsUniversidad"
        WsUniversidad universidadService = new WsUniversidad();

        // Paso 9) Se llama al WebMethod
"ValidarRequerimientos" que se encuentra en el WebService
"WsUniversidad"

        bool aptoFacultad =
universidadService.ValidarRequerimientos(becado, edad,
cantidadMaterias, listaLenguajes, especialidad, nacionalidad);
        if (aptoFacultad)
        {
// Paso 10) Si cumple con los requerimientos se muestra un mensaje
indicando que el usuario es apto para la universidad.
            Response.Write("Señor " + apellido + " usted es
apto para entrar en esta universidad");
        }
        else if(!aptoFacultad)
        {
// Paso 11) Si NO cumple con los requerimientos se muestra un
mensaje indicando que el usuario es apto para la universidad.
            Response.Write("Señor " + apellido + " usted no es
apto para entrar en esta universidad");
        }
    }

    // Paso 2.1) Consistir el campo Edad (Edad Mayor a 17 y
menor a 99) consumiendo el Segundo WebMethod

    internal static bool _ConsistirEdad(int edad)
    {
        // Instanciación del webservice
        WsUniversidad universidadService = new
WsUniversidad();

        // Llamada al WebMethod _ConsistirEdad que se encuentra
en "WsUniversidad"
        bool aptoEdad =
universidadService._ConsistirEdad(edad);

        if (aptoEdad)
            return true;
        else
            return false;
    }
}
}

```

5. Secuencia de Ejecución

1er Paso: Carga de Formulario

```
DetalleAlumnos.aspx  WsUniversidad.asmx.cs  DetalleAlumnos.aspx.cs*  ComprobarRequisitos.aspx.cs  ComprobarRequisitos.aspx
13  <%-- Paso 1) Ingreso de datos --%>
14  Apellido
15  <asp:TextBox ID="txtApellido" runat="server" Width="151px"></asp:TextBox>
16  <%-- Paso 1.1) Se valida que el campo tenga datos mediante un RequiredFieldValidator --%>
17  <asp:RequiredFieldValidator ID="rfvApellido" runat="server" ControlToValidate="txtApellido" ErrorMessage="Campo obligatorio" />
18  <br />
19  <br />
20  Edad <asp:TextBox ID="txtEdad" runat="server" OnTextChanged="txtEdad_TextChanged" style="height: 25px"></asp:TextBox>
21  <%-- Se valida que el campo tenga datos mediante un RequiredFieldValidator --%>
22  <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="txtEdad" Display="Dynamic" ErrorMessage="El campo es obligatorio" />
23  <br />
24  <%-- El campo edad se consiste mediante un Custom Validator con una función en el servidor: "ComprobarRequisitos" --%>
25  <asp:CustomValidator ID="cvEdad" runat="server" ControlToValidate="txtEdad" Display="Dynamic" ErrorMessage="La edad debe ser mayor a 18" />
26  <br />
27  Cantidad de materias <asp:TextBox ID="txtMaterias" runat="server" OnTextChanged="txtMaterias_TextChanged"></asp:TextBox>
28  <%-- Paso 1.2) Se valida que el campo tenga datos mediante un RequiredFieldValidator --%>
29  <asp:RequiredFieldValidator ID="rfvMaterias" runat="server" ControlToValidate="txtMaterias" ErrorMessage="Campo obligatorio" />
30  <br />
31  <br />
32  Lenguaje <br />
33  <asp:ListBox ID="lstLenguajes" runat="server" SelectionMode="Multiple">
34  <asp:ListItem>Java</asp:ListItem>
35  <asp:ListItem>Visual Basic</asp:ListItem>
36  <asp:ListItem>C#</asp:ListItem>
37  </asp:ListBox>
38  <br />
39  <br />
40  Especialidad <asp:DropDownList ID="ddlEspecialidad" runat="server">
41  <asp:ListItem>Software</asp:ListItem>
42  <asp:ListItem>Funcional</asp:ListItem>
43  <asp:ListItem>QA</asp:ListItem>
```

```

Becado
<asp:CheckBox ID="chkBecado" runat="server" />
<br />
<br />
Nacionalidad:
<asp:RadioButtonList ID="rdlstNacionalidad" runat="server">
    <asp:ListItem>Argentina</asp:ListItem>
    <asp:ListItem>Resto de America</asp:ListItem>
    <asp:ListItem>Europa</asp:ListItem>
</asp:RadioButtonList>

```

2do Paso: Se envían los datos

```

<%-- Boton que envía los datos al servidor --%>
<asp:Button ID="btnEnviar" runat="server" onClick="btnEnviar_Click" Text="Submit" />
<br />

```

3er Paso: Se validan los datos

```

1 referencia
public bool ValidarRequerimientos(bool becado, int edad, int materiasAprobadas, string lenguajes, string especializacion, string nacionalidad)
{
    bool tieneCsharp = false;

    //Si la lista contiene el elemento C#, (Seleccionado de la lista), pone en true la variable bool "tieneCsharp"
    if (lenguajes.Contains("C#"))
    {
        tieneCsharp = true;
    }

    if (!becado)
    {
        //Validar la condición del return
        // Condiciones
        // a) Materias aprobadas mayor a 30
        // b) Edad menor a 40 años
        // c) Especialización elegida "Software"
        // d) Nacionalidad elegida "Argentina"
        return (tieneCsharp && materiasAprobadas > 30 && edad < 40 && especializacion == "Software" && nacionalidad == "Argentina");
    }

    //Cumple las condiciones (Porque es becado)
    return true;
}

```

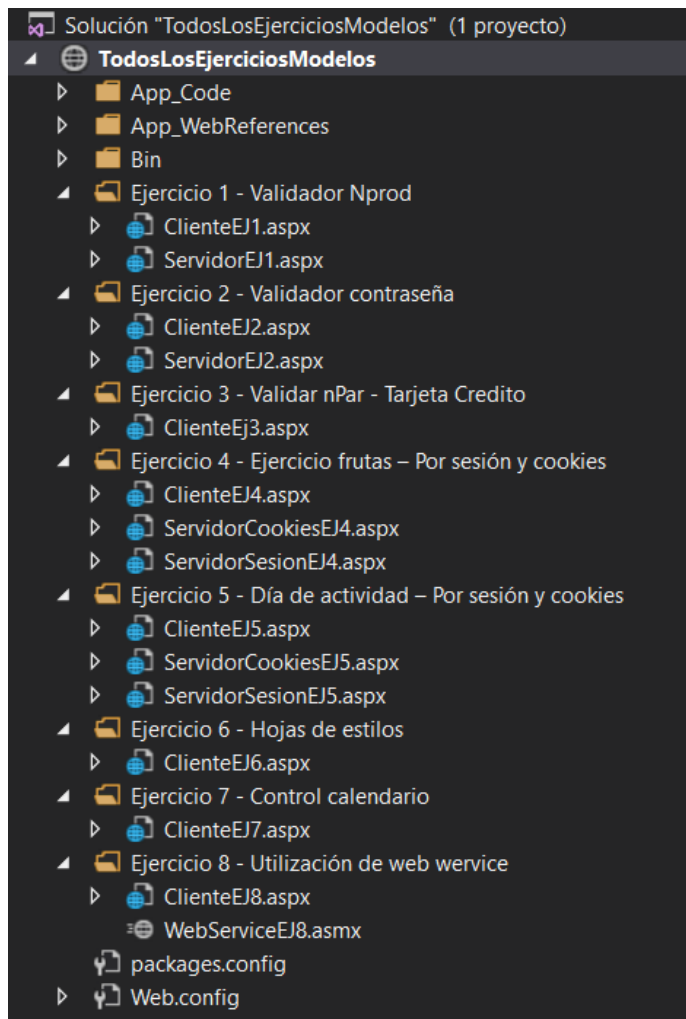
Resumen primer parte de la materia: Ejercicios prácticos clase a clase

Índice

Solución con todos los ejercicios:.....	4
Clase 1 15/08/2018.....	5
Ejercicio 1: Validadores.....	5
ClienteEJ1.aspx.....	5
ServidorEJ1.aspx.....	6
Clase 2 - 22/08/2018.....	7
Ejercicio 2: Validador de contraseña.....	7
ClienteEJ2.aspx.....	7
ServidorEJ2.aspx.....	8
Ejercicio 3: CustomValidator.....	9
ClienteEj3.aspx.....	9
ClienteEJ3.aspx.cs.....	10
Clase 3 – 29/08/18.....	12
Ejercicio 4: Ejercicio frutas – Por sesión y cookies.....	12
ClienteEJ4.aspx.....	12
ClienteEJ4.aspx.cs.....	13
ServidorSesionEJ4.aspx.cs (Session).....	14
ServidorCookiesEJ4.aspx.cs (Cookies).....	14

Ejercicio 5: “Día de actividad – Por sesión y cookies”	15
ClienteEJ5.aspx.....	15
ClienteEJ5.aspx.cs.....	16
ServidorSesionEJ5.aspx (Session).....	17
ServidorSesionEJ5.aspx.cs (Session).....	18
ServidorCookiesEJ5.aspx.cs (Cookies).....	19
ServidorCookiesEJ5.aspx.cs (Cookies).....	19
Ejercicio 6: “Hojas de estilos”	20
ClienteEJ7.aspx.....	20
ClienteEJ7.aspx.cs.....	21
Clase 4 – 05/09/18.....	22
Ejercicio 7: “Control calendario con Range Validator”	22
ClienteEJ7.aspx.....	22
ClienteEJ7.aspx.cs.....	23
Ejercicio 8: “Utilización de web wervice”	24
ClienteEJ8.aspx.....	24
ClienteEJ8.aspx.cs.....	25
WebServiceEJ8.cs.....	25

Solución con todos los ejercicios:



Clase 1 15/08/2018

Ejercicio 1: Validadores

Consigna: "Insertar código de producto que comience con P mayúscula y con 4 números seguidos"

ClienteEJ1.aspx

INSERTE UN CODIGO DE PRODUCTO QUE COMIENZE CON "P" (MAYUSCULA) Y CON CUATRO NUMEROS SEGUIDOS.


```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ClienteEJ1.aspx.cs"
Inherits="Validadores.WebForm1" %>
<!DOCTYPE html>
<script runat="server">
    protected void botonClick(object sender, EventArgs e)
    {
        if(IsValid)
        {
            Response.Redirect("ServidorEJ1.aspx");
        }
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Que comience con P y 4 numeros seguidos</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            INSERTE UN CODIGO DE PRODUCTO QUE COMIENZE CON "P" (MAYUSCULA) Y CON
            CUATRO NUMEROS SEGUIDOS.

            <asp:TextBox ID="CodigoProducto" runat="server"></asp:TextBox>

            <asp:RequiredFieldValidator ID="AA" ControlToValidate="CodigoProducto"
Text="Ese campo DEBE LLENARSE" runat="server"></asp:RequiredFieldValidator>
            <asp:RegularExpressionValidator ID="bb"
ControlToValidate="CodigoProducto" Text="Ese codigo de producto es invalido"
ValidationExpression="P[0-9]{4}" runat="server"></asp:RegularExpressionValidator>

            <asp:Button ID="cc" Text="ENVIAR" OnClick="botonClick" runat="server" />
        </div>
    </form>
</body>
</html>
```

ServidorEJ1.aspx

HA PASADO EL CONTROL DE VALIDACION.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ServidorEJ1.aspx.cs"
Inherits="Validadores.Resultado" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            HA PASADO EL CONTROL DE VALIDACION.
        </div>
    </form>
</body>
</html>
```

Clase 2 - 22/08/2018

Ejercicio 2: Validador de contraseña

Consigna: Validar que la contraseña tenga:

- Entre 3 y 20 caracteres
- 1 número y una letra
- Que el control tenga contenido

ClienteEJ2.aspx

Password

Enviar

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ClienteEJ2.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html>

<script runat="server">

    protected void botonCLICK(object sender, EventArgs e)
    {
        if (IsValid)
        {
            Response.Redirect("ServidorEJ2.aspx");
        }
    }

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Validators 1</title>
</head>
<body>
    <form id="form2" runat="server">
        <div>
            Password

            <asp:TextBox ID="password" Columns="30" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:RequiredFieldValidator
                ID="AA"
                controltovalidate="password"
                display="Dynamic"
                text="Debe escribir la contraseña"
                runat="server"
            >
            </asp:RequiredFieldValidator>
```

```

        <asp:Button
            ID="cc"
            Text="Enviar"
            onclick="botonCLICK"
            runat="server"
        />

    <asp:RegularExpressionValidator
        ID="bb"
        controltovalidate="password"
        text="Su password debe contener entre 3 y 20 caracteres"
        validationexpression="\w{3,20}"
        runat="server"
    >
</asp:RegularExpressionValidator>

<br />
<br />
<asp:RegularExpressionValidator
    ID="dd"
    controltovalidate="password"
    text="Su password debe contener al menos 1 numero y un caracter"
    validationexpression="[a-zA-Z]+\w*\d+\w*"
    runat="server" >

</asp:RegularExpressionValidator>
<br />
<br />

</div>
</form>
</body>
</html>

```

ServidorEJ2.aspx

Paso las validaciones.

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ServidorEJ2.aspx.cs"
Inherits="zzzz" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Paso las validaciones.
        </div>
    </form>
</body>
</html>

```

Ejercicio 3: CustomValidator

Consigna: Validar que sea un número par – Validar que sea una tarjeta de crédito válida

ClienteEj3.aspx

Ingrese un numero par

Validar

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ClienteEj3.aspx.cs"
Inherits="ClienteEj3" %>
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
<title>CustomValidator</title>
```

```
<script type="text/javascript">
```

```
function ClientValidate(source, arguments){
    if ((arguments.Value % 2) == 0)
        arguments.IsValid = true;
    else
        arguments.IsValid = false;
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

```
<div>
```

```
<h2>Custom validator</h2>
```

```
<asp:Label ID="lblOutput" runat="server" Text="Label"></asp:Label>
```

```
<br />
```

```
<br />
```

```
<h4>Ingrese un numero par</h4>
```

```
<p>
```

```
<asp:TextBox ID="txtNumero" runat="server"></asp:TextBox>
```

```
</p>
```

```
<br />
```

```
<p>
```

```
<asp:CustomValidator
```

```
    ID="CustomValidator1"
```

```
    runat="server"
```

```
    ErrorMessage="CLIENTE: No es numero par!"
```

```
    ClientValidationFunction="ClientValidate"
```

```

        ControlToValidate="txtNumero"
        Display="Dynamic"
        ForeColor="Red"
        ValidateEmptyText="True"></asp:CustomValidator>
    </p>
    <p>
        <asp:CustomValidator
            ID="CustomValidator2"
            runat="server"
            ErrorMessage="SERVIDOR : No es una tarjeta de credito valida!"
            OnServerValidate="CustomValidator2_ServerValidate"
            ControlToValidate="txtNumero"
            Display="Dynamic"
            ForeColor="Red"></asp:CustomValidator>

    </p>

</div>
    <asp:Button ID="btnEnviar" runat="server" Text="Validar"
OnClick="btnEnviar_Click" />
</form>
</body>
</html>

```

ClienteEJ3.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Ejercicio_2___Pag_4_ClienteEj2 : System.Web.UI.Page
{
    protected void CustomValidator2_ServerValidate(object source,
ServerValidateEventArgs args)
    {
        string num = args.Value;
        if (num.Length < 16)
        {
            args.IsValid = false;
        }
        else
        {
            args.IsValid = true;
        }
    }

    protected void btnEnviar_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            lblOutput.Text = "Pagina valida!";
            lblOutput.ForeColor = System.Drawing.Color.Green;
        }
        else

```



```
{
    lblOutput.Text = "La pagina NO ES VALIDA!";
    lblOutput.ForeColor = System.Drawing.Color.Red;
}
}
```

Clase 3 – 29/08/18

Ejercicio 4: Ejercicio frutas - Por sesión y cookies.

ClienteEJ4.aspx

Fruta

Banana
Manzana
Pera
Frutilla

☐ Fruta de Estacion

Enviar por sesion

Enviar por cookies

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ClienteEJ4.aspx.cs"
Inherits="Ejercicio_4___Ejercicio_frutas___Por_sesión_y_cookies_ClienteEJ4" %>
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
<title></title>
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

```
<div style="height: 254px">
```

```
<asp:Label ID="lblfrutas" runat="server" Text="Fruta"></asp:Label>
```

```
<br />
```

```
<br />
```

```
<asp:ListBox ID="ListBox1" runat="server" Height="81px"
style="margin-bottom: 0px" Width="97px">
```

```
<asp:ListItem>Banana</asp:ListItem>
```

```
<asp:ListItem>Manzana</asp:ListItem>
```

```
<asp:ListItem>Pera</asp:ListItem>
```

```
<asp:ListItem>Frutilla</asp:ListItem>
```

```
</asp:ListBox>
```

```
<br />
```

```
<br />
```

```
<asp:CheckBox ID="ChkFruta" runat="server" Text="Fruta de Estacion" />
```

```
<br />
```

```
<br />
```

```
<asp:Button ID="btnEnviarCookies" runat="server"
```

```
OnClick="btnEnviarCookies_Click" Text="Enviar por sesion" />
```

```
<br />
```

```
</div>
```

```
<asp:Button ID="btnEnviarPorSesion" runat="server"
```

```
OnClick="btnEnviarPorSesion_Click" Text="Enviar por cookies" />
```

```

    </form>
</body>
</html>

```

ClienteEJ4.aspx.cs

Frutilla

la fruta de estacion

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _ClienteEJ4 : System.Web.UI.Page
{
    protected void btnEnviar_click(object sender, EventArgs e)
    {
        if (this.ListBox1.SelectedValue == "")
        {
            Session.Add("fruta", "no se eligio");
            this.ChkFruta.Enabled = false;
        }
        else
        {
            Session.Add("fruta", this.ListBox1.SelectedValue);
            this.ChkFruta.Enabled = true;
            if (this.ChkFruta.Checked==true)
            {
                Session.Add("estacion", "si");
            }
            else
            {
                Session.Add("estacion", "no");
            }
            Response.Redirect("ServidorSesionEJ4.aspx");
        }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (this.ListBox1.SelectedValue == "")
        {
            Response.Cookies["Fruta"]["1"] = "No se eligio";
            this.ChkFruta.Enabled = false;
        }
        else
        {
            Response.Cookies["Fruta"]["1"] = this.ListBox1.SelectedValue;
            this.ChkFruta.Enabled = true;
            if (this.ChkFruta.Checked == true)
            {
                Response.Cookies["Estacion"]["1"] = "si";
            }
        }
    }
}

```

```

        }
        else
        {
            Response.Cookies["Estacion"]["1"] = "no";
        }
        Response.Redirect("ServidorCookiesEJ4.aspx");
    }
}
}

```

ServidorSesionEJ4.aspx.cs (Session)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class ServidorSesionEJ4 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.lblfruta.Text = Session["fruta"].ToString();
        if (Session["estacion"] == "si")
        {
            this.lblestacion.Text = "la fruta de estacion";
        }
        else
        {
            this.lblestacion.Text = "la fruta no es de estacion";
        }
    }
}

```

ServidorCookiesEJ4.aspx.cs (Cookies)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class ServidorCookiesEJ4 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.lblfruta.Text = Request.Cookies["Fruta"]["1"].ToString();
        if (Request.Cookies["Estacion"]["1"] == "si")
        {
            this.lblestacion.Text = "la fruta de estacion";
        }
        else
        {
            this.lblestacion.Text = "la fruta no es de estacion";
        }
    }
}

```

```

        {
            this.lblestacion.Text = "la fruta no es de estacion";
        }
    }
}

```

Ejercicio 5: “Día de actividad - Por sesión y cookies”

ClienteEJ5.aspx

Elija un dia de actividad

☐ Lunes

☐ Martes

☐ Miercoles

☐ Jueves

☐ Viernes

Enviar por sesion

Enviar por cookies

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class
Ejercicio_5__Día_de_actividad__Por_sesión_y_cookies_ClienteEJ5 :
System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Session["Lunes"] = "";
        Session["Martes"] = "";
        Session["Miercoles"] = "";
        Session["Jueves"] = "";
        Session["Viernes"] = "";
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (CheckBox1.Checked)
            Session["Lunes"] = CheckBox1.Text;

        if (CheckBox2.Checked)
            Session["Martes"] = CheckBox2.Text;
    }
}

```

```

        if (CheckBox3.Checked)
            Session["Miercoles"] = CheckBox3.Text;

        if (CheckBox4.Checked)
            Session["Jueves"] = CheckBox4.Text;

        if (CheckBox5.Checked)
            Session["Viernes"] = CheckBox5.Text;
        Response.Redirect("ServidorSesionEJ5.aspx");
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        if (CheckBox1.Checked)
            Response.Cookies["Dia"]["Lunes"] = CheckBox1.Text;

        if (CheckBox2.Checked)
            Response.Cookies["Dia"]["Martes"] = CheckBox2.Text;

        if (CheckBox3.Checked)
            Response.Cookies["Dia"]["Miercoles"] = CheckBox3.Text;

        if (CheckBox4.Checked)
            Response.Cookies["Dia"]["Jueves"] = CheckBox4.Text;

        if (CheckBox5.Checked)
            Response.Cookies["Dia"]["Viernes"] = CheckBox5.Text;

        Response.Redirect("ServidorCookiesEJ5.aspx");
    }
}

```

ClienteEJ5.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class
Ejercicio_5___Día_de_actividad___Por_sesión_y_cookies_ClienteEJ5 :
System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Session["Lunes"] = "";
        Session["Martes"] = "";
        Session["Miercoles"] = "";
        Session["Jueves"] = "";
        Session["Viernes"] = "";
    }

    protected void Button1_Click(object sender, EventArgs e)
    {

```

```

        if (CheckBox1.Checked)
            Session["Lunes"] = CheckBox1.Text;

        if (CheckBox2.Checked)
            Session["Martes"] = CheckBox2.Text;

        if (CheckBox3.Checked)
            Session["Miercoles"] = CheckBox3.Text;

        if (CheckBox4.Checked)
            Session["Jueves"] = CheckBox4.Text;

        if (CheckBox5.Checked)
            Session["Viernes"] = CheckBox5.Text;
        Response.Redirect("ServidorSesionEJ5.aspx");
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        if (CheckBox1.Checked)
            Response.Cookies["Dia"]["Lunes"] = CheckBox1.Text;

        if (CheckBox2.Checked)
            Response.Cookies["Dia"]["Martes"] = CheckBox2.Text;

        if (CheckBox3.Checked)
            Response.Cookies["Dia"]["Miercoles"] = CheckBox3.Text;

        if (CheckBox4.Checked)
            Response.Cookies["Dia"]["Jueves"] = CheckBox4.Text;

        if (CheckBox5.Checked)
            Response.Cookies["Dia"]["Viernes"] = CheckBox5.Text;

        Response.Redirect("ServidorCookiesEJ5.aspx");
    }
}

```

ServidorSesionEJ5.aspx (Session)

Usted realiza las actividades de los dias

Martes
Miercoles

```

<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="ServidorSesionEJ5.aspx.cs"
Inherits="Ejercicio_5___Día_de_actividad___Por_sesión_y_cookies_ServidorSesionEJ5"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">

```

```

        <div>

            <asp:Label ID="lblMensaje" runat="server" Text="Label"></asp:Label>
        </div>
    </form>
</body>
</html>

```

ServidorSesionEJ5.aspx.cs (Session)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class
Ejercicio_5___Día_de_actividad___Por_sesión_y_cookies_ServidorSesionEJ5 :
System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string lun = Session["Lunes"].ToString();
        string mar = Session["Martes"].ToString();
        string mierc = Session["Miercoles"].ToString();
        string juev = Session["Jueves"].ToString();
        string vier = Session["Viernes"].ToString();

        lblMensaje.Text = "Usted realiza las actividades de los días<br>" + lun +
        "<br>" + mar + "<br>" + mierc + "<br>" + juev + "<br>" + vier;

    }
}

```

ServidorCookiesEJ5.aspx.cs (Cookies)

Usted realiza las actividades de los dias

Martes
Miercoles

```

<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="ServidorCookiesEJ5.aspx.cs"
Inherits="Ejercicio_5___Día_de_actividad___Por_sesión_y_cookies_ServidorCookiesEJ
5" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

```



```

<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

<asp:Label ID="lblMensaje" runat="server" Text="Label"></asp:Label>

</div>
</form>
</body>
</html>

```

ServidorCookiesEJ5.aspx.cs (Cookies)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class
Ejercicio_5___Día_de_actividad___Por_sesión_y_cookies_ServidorCookiesEJ5 :
System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string lun = Request.Cookies["Dia"]["Lunes"];
        string mar = Request.Cookies["Dia"]["Martes"];
        string mierc = Request.Cookies["Dia"]["Miercoles"];
        string juev = Request.Cookies["Dia"]["Jueves"];
        string vier = Request.Cookies["Dia"]["Viernes"];

        lblMensaje.Text = "Usted realiza las actividades de los dias<br>" + lun +
"<br>" + mar + "<br>" + mierc + "<br>" + juev + "<br>" + vier;

    }
}

```

Ejercicio 6: “Hojas de estilos”

ClienteEJ7.aspx

[Demostracion hojas de estilo con clases](#)

[Fuente Script](#)

[Fuente verdana](#)

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ClienteEJ6.aspx.cs"
Inherits="Ejercicio_6___Hojas_de_estilos_ClienteEJ6" %>

```

```

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <style>
        .miclase1
        {
            font:18;
            color:blue;
        }
        .miclase2
        {
            font:24ptverdana;
            color:red;
        }
    </style>
    <title>Ejercicio hojas de estilo</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="milabel" runat="server" Text="Demostracion hojas de
estilo con clases"></asp:Label>
            <br />
            <br />
        </div>
        <asp:LinkButton ID="Script" runat="server" OnClick="Script_Click">Fuente
Script</asp:LinkButton>
        <br />
        <br />
        <asp:LinkButton ID="Verdana" runat="server"
OnClick="Verdana_Click">Fuente verdana</asp:LinkButton>
        <br />
    </form>
</body>
</html>

```

ClienteEJ7.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Ejercicio_6___Hojas_de_estilos_ClienteEJ6 :
System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Script_Click(object sender, EventArgs e)
    {
        milabel.CssClass = "miclase1";
    }
}

```

```
}  
  
protected void Verdana_Click(object sender, EventArgs e)  
{  
    milabel.CssClass = "miclase2";  
}  
}
```

Clase 4 – 05/09/18

Ejercicio 7: “Control calendario con Range Validator”

ClienteEJ7.aspx

< septiembre de 2018 >						
lu.	ma.	mi.	ju.	vi.	sá.	do.
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

La fecha de la reunion no debe superar los 2 meses, ni debe ser menor a la fecha de hoy

Enviar

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ClienteEJ7.aspx.cs"
Inherits="Ejercicio_7___Control_calendario_ClienteEJ7" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <asp:AdRotator ID="publicidad" runat="server" BorderColor="Black"
BorderWidth="1px" />
            <br />
            <br />
            <asp:TextBox ID="txtfecha" runat="server"></asp:TextBox>
            <br />
            <br />
            <br />
            <asp:Calendar ID="Calendario" runat="server"
OnSelectionChanged="Calendario_SelectionChanged"></asp:Calendar>
            <br />
            <asp:RangeValidator ID="fechaReunion" runat="server"
ErrorMessage="RangeValidator" ControlToValidate="txtfecha" Display="Dynamic"
Type="Date">La fecha de la reunion no debe superar los 2 meses, ni debe ser menor
a la fecha de hoy</asp:RangeValidator>
            <br />
            <br />
            <asp:Button ID="btnEnviar" runat="server" Text="Enviar" />
            <br />
            <br />

        </div>
    </form>
</body>
</html>
```

ClienteEJ7.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Ejercicio_7___Control_calendario_ClienteEJ7 :
System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        ValidationSettings.UnobtrusiveValidationMode =
System.Web.UI.UnobtrusiveValidationMode.None;
        fechaReunion.MinimumValue = DateTime.Now.ToShortDateString();
        fechaReunion.MaximumValue =
DateTime.Now.AddMonths(2).ToShortDateString();

        txtfecha.Text = DateTime.Now.ToShortDateString();
    }

    protected void Calendario_SelectionChanged(object sender, EventArgs e)
    {
        this.txtfecha.Text = this.Calendario.SelectedDate.ToShortDateString();
    }

    protected void Application_Start(object sender, EventArgs e)
    {
        ValidationSettings.UnobtrusiveValidationMode =
System.Web.UI.UnobtrusiveValidationMode.None;
    }
}
```

Ejercicio 8: “Utilización de web wervice”

ClienteEJ8.aspx



Numero 1

Numero 2

Resultado suma

Resultado resta

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ClienteEJ8.aspx.cs"
Inherits="Ejercicio_8___Utilización_de_web_wervice_ClienteEJ8" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

<br />
<asp:Label ID="Label1" runat="server" Text="Numero 1"></asp:Label>
<asp:TextBox ID="txta" runat="server"></asp:TextBox>
<br />
<br />
<asp:Label ID="Label2" runat="server" Text="Numero 2"></asp:Label>
<asp:TextBox ID="txtB" runat="server"></asp:TextBox>

<br />
<br />
<asp:Button ID="btnEnviar" runat="server" OnClick="Button1_Click"
Text="Enviar" Width="235px" />
<br />
<br />

<asp:Label ID="Label3" runat="server" Text="Resultado suma"></asp:Label>

<br />

<asp:TextBox ID="txtSuma" runat="server"></asp:TextBox>
<br />
<br />
<asp:Label ID="Label4" runat="server" Text="Resultado resta"></asp:Label>
<br />
<asp:TextBox ID="txtResta" runat="server"></asp:TextBox>
<br />

</div>
</form>
</body>
```

```
</html>
```

ClienteEJ8.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Ejercicio_8___Utilización_de_web_wervice_ClienteEJ8 :
    System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        WebService WebSer = new WebService();

        double a;
        double b;
        double resultadoSuma;
        double resultadoResta;

        a = Convert.ToDouble(txta.Text);
        b = Convert.ToDouble(txtB.Text);

        resultadoSuma = WebSer.Sumar(a, b);
        txtSuma.Text = resultadoSuma.ToString();

        resultadoResta = WebSer.Resta(a, b);
        txtResta.Text = resultadoResta.ToString();
    }
}
```

WebServiceEJ8.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

/// <summary>
/// Descripción breve de WebService
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// Para permitir que se llame a este servicio web desde un script, usando ASP.NET
// AJAX, quite la marca de comentario de la línea siguiente.
// [System.Web.Script.Services.ScriptService]
public class WebService : System.Web.Services.WebService {
```

```

    public WebService () {

        //Elimine la marca de comentario de la línea siguiente si utiliza los
        componentes diseñados
        //InitializeComponent();
    }

    [WebMethod]
    public double Sumar(double a, double b)
    {

        return a + b;
    }

    [WebMethod]
    public double Resta(double a, double b)
    {

        return a - b;
    }

}

```

Ejercicio Fruto por sesión y Cookies

Default.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <div style="height: 254px">

            <asp:Label ID="lblfrutas" runat="server" Text="Fruta"></asp:Label>

            <br />

            <br />

            <asp:ListBox ID="ListBox1" runat="server" Height="81px" style="margin-bottom: 0px"
Width="97px">

```



```

        <asp:ListItem>Banana</asp:ListItem>

        <asp:ListItem>Manzana</asp:ListItem>

        <asp:ListItem>Pera</asp:ListItem>

        <asp:ListItem>Frutilla</asp:ListItem>

    </asp:ListBox>

    <br />

    <asp:CheckBox ID="ChkFruta" runat="server" Text="Fruta de Estacion" />

    <br />

    <br />

    <asp:Button ID="btnEnviar" runat="server" Text="Enviar Session"
OnClick="btnEnviar_click"/>

    <br />

    <br />

    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Enviar
Cookies" />

    <br />

</div>

</form>

</body>

</html>

```

Default.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void btnEnviar_click(object sender, EventArgs e)
    {
        if (this.ListBox1.SelectedValue == "")
        {
            Session.Add("fruta", "no se eligio");
            this.ChkFruta.Enabled = false;
        }
    }
}

```

```

    }
    else
    {
        Session.Add("fruta", this.ListBox1.SelectedValue);
        this.ChkFruta.Enabled = true;
        if (this.ChkFruta.Checked==true)
        {
            Session.Add("estacion", "si");
        }
        else
        {
            Session.Add("estacion", "no");
        }
        Response.Redirect("resultado.aspx");
    }

    }

protected void Button1_Click(object sender, EventArgs e)
{
    if (this.ListBox1.SelectedValue == "")
    {
        Response.Cookies["Fruta"]["1"] = "No se eligio";
        this.ChkFruta.Enabled = false;
    }
    else
    {
        Response.Cookies["Fruta"]["1"] = this.ListBox1.SelectedValue;
        this.ChkFruta.Enabled = true;
        if (this.ChkFruta.Checked == true)
        {
            Response.Cookies["Estacion"]["1"] = "si";
        }
        else
        {
            Response.Cookies["Estacion"]["1"] = "no";
        }
        Response.Redirect("Resultadocookies.aspx");
    }
}
}

```

Resultado.aspx.cs (Session)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Resultado : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.lblfruta.Text = Session["fruta"].ToString();
        if (Session["estacion"] == "si")
        {
            this.lblestacion.Text = "la fruta de estacion";
        }
        else
    }
}

```

```

        {
            this.lblestacion.Text = "la fruta no es de estacion";
        }
    }

    protected void btnvolver_click(object sender, EventArgs e)
    {
        Response.Redirect("Default.aspx");
    }
}

```

Resultado.aspx.cs (Cookies)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Resultadocookies : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.lblfruta.Text = Request.Cookies["Fruta"]["1"].ToString();
        if (Request.Cookies["Estacion"]["1"] == "si")
        {
            this.lblestacion.Text = "la fruta de estacion";
        }
        else
        {
            this.lblestacion.Text = "la fruta no es de estacion";
        }
    }

    protected void btnvolver_click(object sender, EventArgs e)
    {
        Response.Redirect("Default.aspx");
    }
}

```

Ejercicio Dia de actividad (Session y Cookies)

Default.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

    <title></title>

</head>

<body>

```

```

<form id="form1" runat="server">
<div>
</div>
<p>
    Elija un dia de actividad</p>
<p>
    &nbsp;   </p>
<asp:CheckBox ID="CheckBox1" runat="server" Text="Lunes" />
<br />
<br />
<asp:CheckBox ID="CheckBox2" runat="server" Text="Martes" />
<br />
<br />
<asp:CheckBox ID="CheckBox3" runat="server" Text="Miercoles" />
<br />
<br />
<asp:CheckBox ID="CheckBox4" runat="server" Text="Jueves" />
<br />
<br />
<asp:CheckBox ID="CheckBox5" runat="server" Text="Viernes" />
<br />
<br />
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Enviar por
sesion" />
<br />
<br />
<asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="Enviar por
cookies" />
<br />
</form>
</body>
</html>

```

Default.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Session["Lunes"] = "";
        Session["Martes"] = "";
        Session["Miercoles"] = "";
        Session["Jueves"] = "";
        Session["Viernes"] = "";
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (CheckBox1.Checked)
            Session["Lunes"] = CheckBox1.Text;

        if (CheckBox2.Checked)
            Session["Martes"] = CheckBox2.Text;

        if (CheckBox3.Checked)
            Session["Miercoles"] = CheckBox3.Text;

        if (CheckBox4.Checked)
            Session["Jueves"] = CheckBox4.Text;

        if (CheckBox5.Checked)
            Session["Viernes"] = CheckBox5.Text;
        Response.Redirect("Vuelta.aspx");
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        if (CheckBox1.Checked)
            Response.Cookies["Dia"]["Lunes"] = CheckBox1.Text;

        if (CheckBox2.Checked)
            Response.Cookies["Dia"]["Martes"] = CheckBox2.Text;

        if (CheckBox3.Checked)
            Response.Cookies["Dia"]["Miercoles"] = CheckBox3.Text;

        if (CheckBox4.Checked)
            Response.Cookies["Dia"]["Jueves"] = CheckBox4.Text;

        if (CheckBox5.Checked)
```

```

        Response.Cookies["Dia"]["Viernes"] = CheckBox5.Text;

        Response.Redirect("VueltaCookies.aspx");
    }
}

```

Vuelta.aspx.cs (Session)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Vuelta : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string lun = Session["Lunes"].ToString();
        string mar = Session["Martes"].ToString();
        string mierc = Session["Miercoles"].ToString();
        string juev = Session["Jueves"].ToString();
        string vier = Session["Viernes"].ToString();

        lblMensaje.Text = "Usted realiza las actividades de los dias<br>" + lun +
"<br>" + mar + "<br>" + mierc + "<br>" + juev + "<br>" + vier;
    }
}

```

Vuelta.aspx.cs (Cookies)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class VueltaCookies : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string lun = Request.Cookies["Dia"]["Lunes"];
        string mar = Request.Cookies["Dia"]["Martes"];
        string mierc = Request.Cookies["Dia"]["Miercoles"];
        string juev = Request.Cookies["Dia"]["Jueves"];
        string vier = Request.Cookies["Dia"]["Viernes"];

        lblMensaje.Text = "Usted realiza las actividades de los dias<br>" + lun +
"<br>" + mar + "<br>" + mierc + "<br>" + juev + "<br>" + vier;
    }
}

```

Dispositivos Móviles

1) Ventajas y desventajas de Stand-Alone y Soluciones Online.

Soluciones Online

Ventajas:

No es necesario distribuir ni instalar ninguna aplicaciones

Se pueden hacer operaciones complejas ya que la ejecución se hace en el servidor

Se puede trabajar con gran cantidad de información.

Desventajas:

No se puede acceder a capacidades de bajo nivel del equipo. Se necesita estar conectado para poder usarlo.

No se pueden usar todos los controles de ingreso disponibles, solo los propuestos por el lenguaje en cuestión (HTML o WML)

Ejecución más lenta(Se debe recargar información contra el servidor.

Stand Alone

Ventajas:

Ejecución Veloz. Manejo de Memoria. Se aprovecha las características de bajo de nivel del equipo.

Se puede trabajar sin necesidad de estar conectado.

Soporte de sincronización con un equipo de escritorio.

Desventajas:

Se debe desarrollar diferentes versiones para cada S.O

No se pueden consultar con centros de datos remotos.

No pueden soportar grandes cantidades de información para búsqueda o almacén.

2) Diferencias entre código nativo y código manejado

El código nativo tiene más rapidez de ejecución, tiene un acceso al 100% de las capacidades del equipo. No requiere la instalación de ningún agregado de ejecución y posee un acceso directo a memoria y de bajo nivel.

Por su parte, el código manejado permite que con un solo proyecto se puede ejecutar en diversos sistemas de hardware. No posee acceso directo a memoria y genera un solo paquete de instalación para todos los equipos. No accede al 100% de los recursos y no posee un acceso directo a memoria.

Tiempo Global

1) ¿Cuáles son los factores que hay que considerar en sistemas distribuidos acerca del estado global?

- Que cada nodo posee su propio reloj
- Que existen los retardos
- Que cada nodo tienen una visión subjetiva del estado global.

2) Respecto del tema anterior, explique en qué consiste el concepto de “causalidad”

La causalidad es un concepto que se desprende de la sincronización interna, la cual mide el intervalo entre dos eventos producidos en tiempos diferentes usando los relojes locales de cada nodo.

Para muchas aplicaciones es más importante mantener bien sincronizados entre sí los relojes locales que conseguir una gran precisión en la sincronización externa (el instante preciso en que se produce un evento). La causalidad, entonces, es la capacidad de ordenar dichos eventos para que se encuentren bien sincronizados.

3) ¿Qué es NTP? Describa sus objetivos

NTP: Protocolo de tiempo de Red, define una arquitectura para un servicio de tiempo y un protocolo para distribuir la información de forma precisa del tiempo sobre internet.

Sus objetivos son:

- Proporcionar un servicio que permita a los clientes a lo largo de internet estar sincronizados de forma precisa a UTC.
- Proporcionar un servicio fiable que pueda sobrevivir a pérdidas largas de conectividad: Hay servidores redundantes y recorridos redundantes entre los servidores.
- Proporcionar a los clientes re sincronizar con suficiente frecuencia para compensar las tasas de deriva encontrados en la mayoría de las computadoras: el servicio está diseñado para escalar a gran número de clientes y servidores.
- Proporcionar protección contra la interferencia con el servicio de tiempo: el servicio de tiempo usa técnicas de autenticación para comprobar que los datos de tiempo se originan de las fuentes verdaderas. También valida las direcciones de devolución de las mensajes enviados hacia él.

4) ¿Cómo se sincronizan entre sí los servidores NTP?

Tiene tres formas:

Modo multidifusión: Pensado para uso LAN. Uno o más servidores reparten periódicamente el tiempo a los servidores que se ejecutan en otras computadoras conectadas en la LAN, que fijan sus relojes suponiendo un pequeño retardo. Este modo puede alcanzar sólo precisiones relativamente bajas, pero que no obstante son consideradas suficientes para muchos propósitos.

Modo de llamada a procedimiento: Un servidor acepta solicitudes de otras computadoras, que él procesa respondiendo con su marca de tiempo (lectura actual del reloj). Es aplicable donde se requieran precisiones más altas que las que se pueden conseguir con multidifusión o donde la multidifusión no viene soportada por hardware.

Modo simétrico: Pensado para que lo utilicen servidores que proporcionan información del tiempo en LANs y por los niveles más altos de la subred de sincronización, donde se deben obtener precisiones más altas. Un par de servidores operando en modo simétrico intercambian mensajes llevando información del tiempo. Los datos del tiempo son retenidos como parte de

una asociación entre los servidores que se mantiene con el fin de mejorar la precisión de su sincronización en el tiempo.

HTML5

1) ¿Qué elementos aprovecha la API Geolocation?

La API Geolocation aprovecha nuevos sistemas, como triangulación de red y GPS, para retornar una ubicación precisa del dispositivo que está accediendo a la aplicación. Esta valiosa información nos permite construir aplicaciones que se adaptarán a las particulares necesidades del usuario o proveerán información localizada de forma automática.

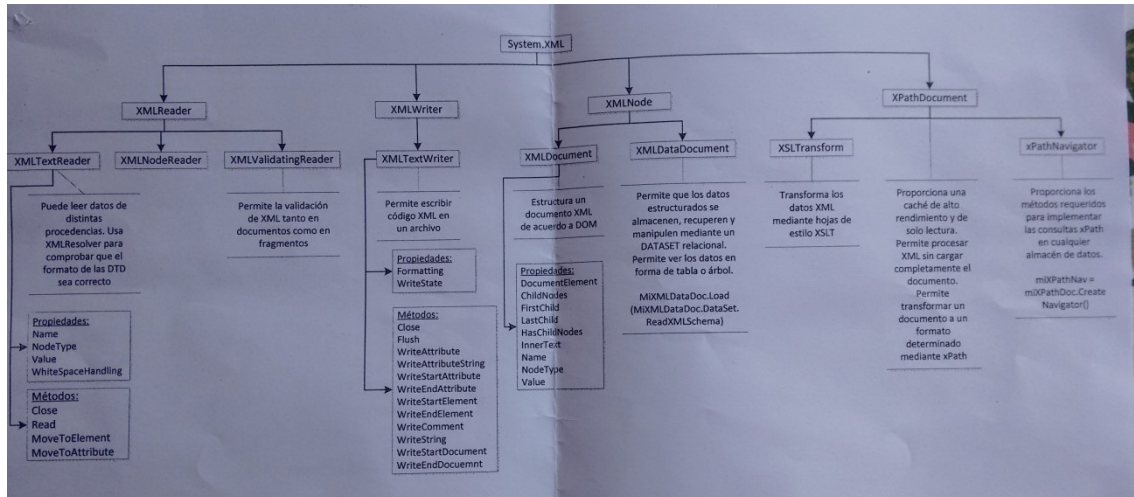
2) La API anterior cuenta con tres métodos. Explique dos o aplique código que permita obtener información acerca de la ubicación de su usuario.

Los tres métodos son:

- 1) **getCurrentPosition(ubicación, error, configuración):** Este es el método utilizado para consultas individuales. Puede recibir hasta tres atributos: una función para procesar la ubicación retornada, otra para procesar los errores retornados, y un objeto para configurar cómo la información será adquirida.
- 2) **watchPosition(ubicación, error, configuración):** Este método es similar al anterior, excepto que comenzará un proceso de vigilancia para la detección de nuevas ubicaciones. Repite el proceso automáticamente en determinados períodos de tiempo, de acuerdo a la configuración por defecto o a los valores de sus atributos.
- 3) **clearWatch(id):** El método *watchPosition()* retorna un valor que puede ser almacenado en una variable para luego ser usado como referencia por el método *clearwatch()* y así detener la vigilancia.

XML

1) Dibuje el modelo de objetos de XML.NET



Acá obviamente no haría falta poner los métodos ni explicarlos, solo nombrar desde System.XML hasta el XPathNavigator.

2) ¿Cuáles son los nodos que tienen hijos?

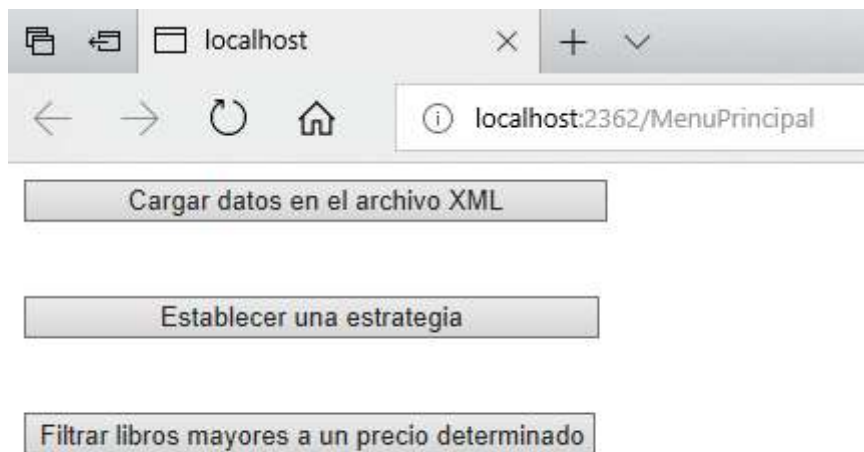
Los nodos que tienen hijos, y sus respectivos hijos, son:

- **DOCUMENT:** Document Type, Element, Processing Instruction, Comment.
- **ELEMENT:** Element, Text, Comment, Processing Instruction, CDATA Section.
- **ATRIBUTTE:** Text, Entity Reference.

Resumen segunda parte de la materia: Práctica

MenuPrincipal

MenuPrincipal.aspx



```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuPrincipal.aspx.cs"
Inherits="MenuPrincipal" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

<asp:Button ID="Button1" runat="server" Text="Cargar datos en el archivo
XML" Width="291px" OnClick="Button1_Click" />

<br />
<br />
<br />

<asp:Button ID="Button2" runat="server" Text="Establecer una estrategia"
Width="287px" OnClick="Button2_Click" />

<br />
<br />
```

```

        <br />

        <asp:Button ID="Button3" runat="server" Text="Filtrar libros mayores a
un precio determinado" Width="285px" OnClick="Button3_Click" />

    </div>

</form>
</body>
</html>

```

MenuPrincipal.aspx.cs

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuPrincipal.aspx.cs"
Inherits="MenuPrincipal" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <asp:Button ID="Button1" runat="server" Text="Cargar datos en el archivo
XML" Width="291px" OnClick="Button1_Click" />

            <br />
            <br />
            <br />

            <asp:Button ID="Button2" runat="server" Text="Establecer una estrategia"
Width="287px" OnClick="Button2_Click" />

            <br />
            <br />
            <br />

            <asp:Button ID="Button3" runat="server" Text="Filtrar libros mayores a
un precio determinado" Width="285px" OnClick="Button3_Click" />

        </div>

    </form>
</body>
</html>

```

EjercicioB.aspx


```

string pathxml = @"E:\Segundo Parcial\EjercicioLibros.xml";

XmlDocument doc = new XmlDocument();
doc.Load(pathxml);

XmlNode nodeLib = doc.CreateElement("Libro");

XmlElement nodeID = doc.CreateElement("ID");
nodeID.InnerText = txtCantLibros.Text;

//XmlElement nodenomlib = doc.CreateElement("NombreLibro");
XmlElement NameBook = doc.CreateElement("NombreLibro");
NameBook.InnerText = txtNombre.Text;

XmlElement nodeAutor = doc.CreateElement("Autor");
nodeAutor.InnerText = txtAutor.Text;

XmlElement nodeEdit = doc.CreateElement("Editorial");
nodeEdit.InnerText = txtEditorial.Text;

XmlNode nodePre = doc.CreateElement("Precio");
nodePre.InnerText = txtPrecio.Text;

nodeLib.AppendChild(nodeID);
nodeLib.AppendChild(NameBook);
nodeLib.AppendChild(nodeAutor);
nodeLib.AppendChild(nodeEdit);
nodeLib.AppendChild(nodePre);

doc.SelectSingleNode("Inventario").AppendChild(nodeLib);

doc.Save(pathxml);

Response.Redirect("EjercicioLibros.xml");
}

}

```

EjercicioC.aspx

Producto: Cantidad:

Precio:

Total:


```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="EjercicioC.aspx.cs"
Inherits="EjercicioC" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <label>Producto: </label>
            <asp:DropDownList ID="ddlLibros"
runat="server"></asp:DropDownList>
            <label style="margin-left:20px">Cantidad: </label>
            <asp:TextBox ID="txtCantidad" runat="server"></asp:TextBox>
        </div>
        <hr />

        <asp:Button ID="btnEnviar" Text="Enviar" runat="server"
OnClick="btnEnviar_Click" />
        <br />
        <div>
            <label>Precio: </label>
            <asp:Label ID="lblPrecio" runat="server"></asp:Label>
            <br />
        </div>
        <div>
            <label>Total: </label>
            <asp:Label ID="lblTotal" runat="server"></asp:Label>
        </div>
    </div>

    </div>
</form>
</body>
</html>

```

EjercicioC.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;

public partial class EjercicioC : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {

```

```

        XmlDocument miDoc = new XmlDocument();
        XmlTextReader miLector = new
XmlTextReader(Server.MapPath("EjercicioLibros.xml"));

        miLector.WhitespaceHandling = WhitespaceHandling.None;
        miDoc.Load(miLector);
        Session.Add("DocumentoEnSesion", miDoc);

        for (int i = 0; i < miDoc.DocumentElement.ChildNodes.Count; i++)
        {

ddlLibros.Items.Add(miDoc.DocumentElement.ChildNodes[i].ChildNodes[1].InnerText);
        }
        miLector.Close();
    }

}

protected void btnEnviar_Click(object sender, EventArgs e)
{

    int n;
    int cantidad = 1;
    double precio;
    XmlDocument miDoc;

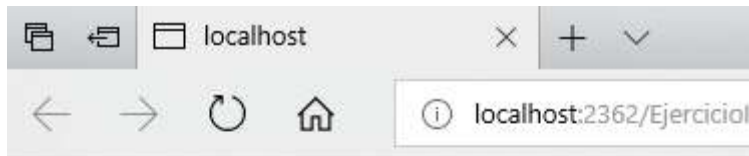
    miDoc = (XmlDocument)Session["DocumentoEnSesion"];
    n = ddlLibros.SelectedIndex;
    cantidad = Convert.ToInt32(txtCantidad.Text);
    precio =
Convert.ToDouble(miDoc.DocumentElement.ChildNodes[n].ChildNodes[4].InnerText);

    lblPrecio.Text = precio.ToString();
    lblTotal.Text = (cantidad * precio).ToString();

}
}

```

Ejerciciol.aspx



Libros disponibles

NombreLibro	Autor	Editorial	Precio
Siberiano	JK	LaSalta	500
A	B	C	800
J	Q	I	400
Siberiano1	T	H	450

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="EjercicioI.aspx.cs"
Inherits="EjercicioI" %>
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
<title></title>
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

```
<div>
```

```
<p><asp:Xml ID="TransformacionXSLT" runat="server"
DocumentSource="~/EjercicioLibros.xml"
TransformSource="XSLTFile1.xslt"></asp:Xml></p>
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

XSLTFile.xslt

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">
  <xsl:template match="/">
    <h4>Libros disponibles</h4>
    <table border="2" cellpadding="5">
      <thead>
```

```

        <th>NombreLibro</th>
        <th>Autor</th>
        <th>Editorial</th>
        <th>Precio</th>
    </thead>
    <tbody>
        <xsl:for-each select="Inventario/Libro">
            <xsl:if test="Precio>70">
                <tr>
                    <td>
                        <xsl:value-of select="NombreLibro"/>
                    </td>
                    <td>
                        <xsl:value-of select="Autor"/>
                    </td>
                    <td>
                        <xsl:value-of select="Editorial"/>
                    </td>
                    <td>
                        <xsl:value-of select="Precio"/>
                    </td>
                </tr>
            </xsl:if>
        </xsl:for-each>
    </tbody>
</table>
</xsl:template>
</xsl:stylesheet>

```

EjercicioLibros.xml

```

<?xml version="1.0"?>
<!--Escribir datos en XML-->
<Inventario NombreTipo="Guardar">
    <Libro>
        <ID>1</ID>
        <NombreLibro>Siberiano</NombreLibro>
        <Autor>JK</Autor>
        <Editorial>LaSalta</Editorial>
        <Precio>500</Precio>
    </Libro>
    <Libro>
        <ID>2</ID>
        <NombreLibro>A</NombreLibro>
        <Autor>B</Autor>
        <Editorial>C</Editorial>
        <Precio>800</Precio>
    </Libro>
    <Libro>
        <ID>3</ID>
        <NombreLibro>J</NombreLibro>
        <Autor>Q</Autor>
        <Editorial>I</Editorial>
        <Precio>400</Precio>
    </Libro>
    <Libro>
        <ID>10</ID>
        <NombreLibro>Siberiano1</NombreLibro>
        <Autor>T</Autor>
    </Libro>
</Inventario>


```

```

        <Editorial>H</Editorial>
        <Precio>450</Precio>
    </Libro>
    <Libro>
        <ID>9</ID>
        <NombreLibro>P</NombreLibro>
        <Autor>L</Autor>
        <Editorial>K</Editorial>
        <Precio>45</Precio>
    </Libro>
</Inventario>

```

Resultado luego de agregar un libro:



The screenshot shows a web browser window with two tabs labeled 'localhost'. The address bar displays 'localhost:2362/EjercicioLibros.xr'. The main content area shows an XML document with the following structure:

```

<?xml version="1.0" ?>
<!--Escribir datos en XML-->
- <Inventario NombreTipo="Guardar">
  - <Libro>
    <ID>1</ID>
    <NombreLibro>Siberiano</NombreLibro>
    <Autor>JK</Autor>
    <Editorial>LaSalta</Editorial>
    <Precio>500</Precio>
  </Libro>
  - <Libro>
    <ID>2</ID>
    <NombreLibro>A</NombreLibro>
    <Autor>B</Autor>
    <Editorial>C</Editorial>
    <Precio>800</Precio>
  </Libro>
  - <Libro>
    <ID>3</ID>
    <NombreLibro>J</NombreLibro>
    <Autor>Q</Autor>
    <Editorial>I</Editorial>
    <Precio>400</Precio>
  </Libro>
  - <Libro>
    <ID>10</ID>
    <NombreLibro>Siberiano1</NombreLibro>
    <Autor>T</Autor>
    <Editorial>H</Editorial>
    <Precio>450</Precio>
  </Libro>
  - <Libro>
    <ID>9</ID>
    <NombreLibro>P</NombreLibro>
    <Autor>L</Autor>
    <Editorial>K</Editorial>
    <Precio>45</Precio>
  </Libro>
</Inventario>

```

Ejercicio A

EjercicioA.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;

namespace EjerciciosXML
{
    public partial class EjercicioA : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            XmlTextReader miLector = new
            XmlTextReader(Server.MapPath("InventarioEjercicioA.xml"));
            int n;

            while (miLector.Read())
            {
                Response.Write(miLector.NodeType.ToString() + ": " +
                miLector.Name + ":" + miLector.Value + "<br/> <br/>");

                if (miLector.HasAttributes)
                {
                    for (n=0;n < miLector.AttributeCount; n++)
                    {
                        miLector.MoveToAttribute(n);
                        Response.Write("<b>" + miLector.NodeType.ToString() + ":"
                        + miLector.Name + ": " + miLector.Value + "<b><br/>");
                    }

                    miLector.MoveToElement();
                }
            }

            miLector.Close();
        }
    }
}
```

InventarioEjercicioA.xml

```
<?xml version="1.0" encoding="utf-8" ?>

<Inventario>
  <Producto Color="Rojo" SubministradorID="2">
    <ProductoID>1</ProductoID>
    <ProductoNombre>Manzanas</ProductoNombre>
    <ProductoPrecio>4.23</ProductoPrecio>
  </Producto>

  <Producto Color="Rojo" SubministradorID="2">
    <ProductoID>2</ProductoID>
    <ProductoNombre>Peras</ProductoNombre>
    <ProductoPrecio>3.12</ProductoPrecio>
  </Producto>
</Inventario>
```

Ejercicio B

EjercicioB.aspx

[illegible]


```

</div>

</form>

</body>
</html>

```

Ejercicio B.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;
namespace EjerciciosXML
{
    public partial class EjercicioB : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnEscribir_Click(object sender, EventArgs e)
        {

            XmlTextWriter miEscriitor = new
            XmlTextWriter(Server.MapPath("EjercicioBClientes.xml"), null);

            miEscriitor.Formatting = Formatting.Indented;
            miEscriitor.WriteStartDocument();
            miEscriitor.WriteComment("Escribir datos en XML");

            //Escribimos el primer elemento llamado "Datoscliente"
            miEscriitor.WriteStartElement("DatosCliente");
            miEscriitor.WriteAttributeString("NombreTipo", "Guardar");
            miEscriitor.WriteStartElement("NombreNumero", "");
            miEscriitor.WriteString(txtNombre.Text);
            miEscriitor.WriteEndElement();

            miEscriitor.WriteStartElement("Apellidos", "");
            miEscriitor.WriteString(txtApellido.Text);
            miEscriitor.WriteEndElement();

            miEscriitor.WriteStartElement("Direccion", "");
            miEscriitor.WriteString(txtDireccion.Text);
            miEscriitor.WriteEndElement();

            miEscriitor.WriteEndDocument();
            miEscriitor.Flush();
            miEscriitor.Close();

            Response.Redirect("EjercicioBClientes.xml");
        }
    }
}

```

```
}  
  }  
}
```

EjercicioBArchivoXML

```
<?xml version="1.0" encoding="utf-8" ?>  
  
<!-- Escribir datos en XML -->  
  
<DatosCliente NombreTipo="Guardar">  
  
  <NombreNumero>Mauro</NombreNumero>  
  <Apellidos>Gomez</Apellidos>  
  <Direccion>106 n 930</Direccion>  
  
</DatosCliente>
```

Ejercicio C

EjercicioC.aspx

[illegible]

[illegible]

Ejercicio C.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;

namespace EjerciciosXML
{
    public partial class Ejercicioc : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // if (IsPostBack == false)
            if (!IsPostBack)
            {
                XmlDocument midoc = new XmlDocument();
                XmlTextReader milector = new
                XmlTextReader(Server.MapPath("EjercicioCInventario.xml"));
                milector.WhitespaceHandling = WhitespaceHandling.None;
                midoc.Load(milector);
                Session.Add("DocumentoEnSesion", midoc);
                for (int i = 0; i < midoc.DocumentElement.ChildNodes.Count; i++)
                {
                    ddlProducto.Items.Add(midoc.DocumentElement.ChildNodes[i].ChildNodes[1].InnerText
                    );
                }
                milector.Close();
            }
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            int n;
            int cantidad = 1;
            double precio;
            XmlDocument midoc;
            midoc = (XmlDocument)Session["DocumentoEnSesion"];
            n = ddlProducto.SelectedIndex;
        }
    }
}
```

```

        cantidad = Convert.ToInt32(txtCantidad.Text);
        precio =
Convert.ToDouble(midoc.DocumentElement.ChildNodes[n].ChildNodes[2].InnerText);

        lblPrecio.Text = precio.ToString();
        lblTotal.Text = (cantidad * precio).ToString();

    }
}
}

```

EjercicioCArchivoXML

```

<?xml version="1.0" encoding="utf-8" ?>
<Inventario>
  <Producto Color="Rojo" SuministradorId="2">
    <ProductoId>1</ProductoId>
    <ProductoNombre>Manzanas</ProductoNombre>
    <ProductoPrecio>4,23</ProductoPrecio>
  </Producto>

  <Producto Color="Verde" SuministradorId="2">
    <ProductoId>2</ProductoId>
    <ProductoNombre>Peras</ProductoNombre>
    <ProductoPrecio>3,12</ProductoPrecio>
  </Producto>
</Inventario>

```

Ejercicio D

1.0 Interfaz:

← → ↺ 🏠 ⓘ localhost:2154/Default

XpathDocument

cuentas en barcelona

2.0 Default.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">

<Font size ="4"><strong>XPathDocument</strong></Font>
<p></p>
<p><em><strong>cuentas en barcelona</strong></em>
<asp:Listbox ID="listbox1" runat="server" Width="137px"
Rows="2"/>
</p>
<p>&nbsp;</p>
<p>
<asp:Button ID="Button1" runat="server" Text="Actualizar cuentas"
/>
</p>

</form>
</body>
</html>

```

3.0 Default.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;
using System.Xml.XPath;
using System.Xml.Xsl;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {

        XPathDocument xpathdocum = new
XPathDocument(Server.MapPath("banco.xml"));

```

```

XPathNavigator xnavegador;
XPathNodeIterator iterador;

//se basa en crear un navegador (variable "xnavegador") que nos permita
acceder a los nodos del documento "banco.xml"
xnavegador = xpathdocum.CreateNavigator();
iterador =
xnavegador.Select("Banco/Cuenta[CuentaSucursal='Barcelona']/CuentaNombre");
//recuperamos secuencialmente los valores de los nodos <CuentaNombre>
Usando un iterador de la clase XPathnodeIterator (la propiedad currrent.value)
while (iterador.MoveNext())
{
    listBox1.Items.Add(iterador.Current.Value);
}

}
}

```

4.0 Documento XML

```

<Cuenta>
<CuentaNum>CU123</CuentaNum>
<CuentaNombre>Miguel Hernández</CuentaNombre>
<CuentaSaldo>5623,12</CuentaSaldo>
<CuentaSucursal>Badajoz</CuentaSucursal>
</Cuenta>
<Cuenta>
<CuentaNum>CA009</CuentaNum>
<CuentaNombre>Esteban Ruíz</CuentaNombre>
<CuentaSaldo>623,12</CuentaSaldo>
<CuentaSucursal>Burgos</CuentaSucursal>
</Cuenta>
<Cuenta>
<CuentaNum>CX555</CuentaNum>
<CuentaNombre>Antoria Pérez</CuentaNombre>
<CuentaSaldo>6023,12</CuentaSaldo>
<CuentaSucursal>Barcelona</CuentaSucursal>
</Cuenta>

```

```
<Cuenta>
<CuentaNum></CuentaNum>
<CuentaNombre></CuentaNombre>
<CuentaSaldo></CuentaSaldo>
<CuentaSucursal></CuentaSucursal>
</Cuenta>
</Banco>
```

5.0 Resultado



Ejercicio E

1.0 Interfaz

xpathNavigator

Selecciones una cuenta

Saldo de cuenta

Activo

2.0 EjercicioE.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="EjercicioE.aspx.cs"
Inherits="EjercicioE" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

        <br />
        xpathNavigator<br/ >
        <br />
        Selecciones una cuenta
        <asp:DropDownList ID="DropDownList1" runat="server" Height="18px"
Width="229px"/>
        <br />
        <br />
        Saldo de cuenta&nbsp;
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        <asp:DropDownList ID="DropDownList2" runat="server"></asp:DropDownList>
        &nbsp; Activo
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br />
        <br />
        <hr />
        <asp:Button ID="Button1" runat="server" Text="Pedir Datos"
OnClick="Button1_click" />
    </div>
</form>
</body>
</html>
```

```

        </div>
    </form>
</body>
</html>

```

3.0 EjercicioE.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;
using System.Xml.XPath;
using System.Xml.Xsl;

public partial class EjercicioE : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (IsPostBack == false)
        {
            XPathDocument xpathdocum = new
XPathDocument(Server.MapPath("BancoEjercicioE.xml"));
            XPathNavigator xnavegador;
            XPathNodeIterator iterador;
            xnavegador = xpathdocum.CreateNavigator();
            iterador = xnavegador.Select("Banco/Cuenta/@cuentanum");
            while (iterador.MoveNext())
            {
                DropDownList1.Items.Add(iterador.Current.Value);
            }
            iterador = xnavegador.Select("Banco/Cuenta/CuentaNombre");
            while (iterador.MoveNext())
            {
                DropDownList2.Items.Add(iterador.Current.Value);
            }
        }
    }

    protected void Button1_click(object sender, EventArgs e)
    {
        string cuentan = DropDownList1.SelectedItem.Text.Trim();
        string cuentanob = DropDownList2.SelectedItem.Text.Trim();
        XPathDocument xpathdocum = new
XPathDocument(Server.MapPath("BancoEjercicioE.xml"));
        XPathNavigator xnavegador;
        XPathNodeIterator iterador;
    }
}

```

```

        xnavegador = xpathdocum.CreateNavigator();
        iterador = xnavegador.Select("Banco/Cuenta[@cuentanum='" + cuentan +
        "']/CuentaSaldo");
        //recuperamos secuencialmente los valores de los nodos <CuentaNombre>
        Usando un iterador de la clase XPathnodeIterator (la propiedad currrent.value)
        iterador.MoveNext();
        TextBox1.Text = string.Format("{0:C}", iterador.Current.Value);
        iterador = xnavegador.Select("Banco/Cuenta[CuentaNombre='" + cuentanob +
        "']/CuentaSaldo");
        iterador.MoveNext();
        TextBox2.Text = string.Format("{0:C}", iterador.Current.Value);

    }
}

```

4.0 Documento XML

```

<?xml version="1.0" encoding="utf-8" ?>

<Banco>

<Cuenta cuentanum="CU1230">

<CuentaNombre>Miguel</CuentaNombre>

<CuentaSaldo>500</CuentaSaldo>

<CuentaSucursal>Badajoz</CuentaSucursal>

</Cuenta>

<Cuenta cuentanum="CU1231">

<CuentaNombre>Jose</CuentaNombre>

<CuentaSaldo>501</CuentaSaldo>

<CuentaSucursal>Barcelona</CuentaSucursal>

</Cuenta>

<Cuenta cuentanum="CU1232">

<CuentaNombre>Juan</CuentaNombre>

<CuentaSaldo>502</CuentaSaldo>

<CuentaSucursal>Lanus</CuentaSucursal>

</Cuenta>

<Cuenta>cuentanum="CU1232">

<CuentaNombre>Perez</CuentaNombre>

<CuentaSaldo>503</CuentaSaldo>

```

<CuentaSucursal>Barcelona</CuentaSucursal>

</Cuenta>

</Banco>

5.0 Resultado

xpathNavigator

Selecciones una cuenta

Saldo de cuenta

Activo

Ejercicio I

1.0 Ejercicioi.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Ejercicioi.aspx.cs"
Inherits="Ejercicioi" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<p><font size=4><strong>Transformaciones XSLT</font></p>
<p><asp:Xml ID="TransformacionXSLT" runat="server"
DocumentSource="~/BancoEjercicioi.xml"
TransformSource="XSLTFile1.xslt"></asp:Xml></p>

</div>
```

```
</form>
</body>
</html>
```

2.0 Documento XML

```
<?xml version="1.0" encoding="utf-8" ?>
<Banco>
  <Cuenta cuentanum="CU123">
    <CuentaNombre>MiguelHernandez</CuentaNombre>
    <CuentaSaldo>5623,12</CuentaSaldo>
    <CuentaSucursal>Badajoz</CuentaSucursal>
  </Cuenta>
  <Cuenta cuentanum="CA056">
    <CuentaNombre>NeusEspinal</CuentaNombre>
    <CuentaSaldo>12603,12</CuentaSaldo>
    <CuentaSucursal>Barcelona</CuentaSucursal>
  </Cuenta>
  <Cuenta cuentanum="CH009">
    <CuentaNombre>EstebanRuiz</CuentaNombre>
    <CuentaSaldo>623,12</CuentaSaldo>
    <CuentaSucursal>Burgos</CuentaSucursal>
  </Cuenta>
  <Cuenta cuentanum="CX555">
    <CuentaNombre>AntoniaPerez</CuentaNombre>
    <CuentaSaldo>0623,12</CuentaSaldo>
    <CuentaSucursal>Barcelona</CuentaSucursal>
  </Cuenta>
</Banco>
```

3.0 Documento XSLT

```
<?xml version ="1.0" encoding ="utf-8"?>
```

```

<xsl:stylesheet version ="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">

<xsl:template match="/">

<h4>Cuenta de Titulares</h4>

<table border="2" cellpadding="5">

<thead>

<th>Num.Cuenta</th>

<th>Titular</th>

<th>Saldo</th>

<th>Sucursal</th>

</thead>

<tbody>

<xsl:for-each select="Banco/Cuenta[CuentaSucursal='Barcelona']">

<tr align ="center">

<td>

<xsl:value-of select="@cuentanum"/>

</td>

<td>

<xsl:value-of select="CuentaNombre"/>

</td>

<td>

<xsl:value-of select="CuentaSaldo"/>

</td>

<td>

<xsl:value-of select="CuentaSucursal"/>

</td>

</tr>

</xsl:for-each>

</tbody>

</table>

</xsl:template>

```

</xsl:stylesheet>

4.0 Resultado



Transfromaciones XSLT

Cuenta de Titulares

Num.Cuenta	Titular	Saldo	Sucursal
CA056	NeusEspinal	12603,12	Barcelona
CX555	AntoniaPerez	0623,12	Barcelona

Ejercicio C

1.0 Interfaz



DOM: Elija un producto

Producto: Cantidad

Precio individual
Label

Precio Total: Label

2.0 Ejercicioc.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="EjercicioC.aspx.cs"
Inherits="EjercicioC" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

<h3>DOM: Elija un producto</h3>

<label>Producto: </label>
<asp:DropDownList ID="ddlProducto" runat="server"></asp:DropDownList>
<asp:Label ID="Label1" runat="server" Text="Cantidad"></asp:Label>
<asp:TextBox ID="txtCantidad" runat="server"></asp:TextBox>

</div>

<asp:Button ID="btnEnviar" runat="server" OnClick="Button1_Click"
Text="Button" Width="135px" />

<br />
Precio individual<div>

<br><asp:Label ID="lblPrecio" runat="server" Text="Label"></asp:Label>

<br />
<br />

</div>

Precio Total:<br><br>

<asp:Label ID="lblTotal" runat="server" Text="Label"></asp:Label>

</form>
</body>
</html>

```

3.0 EjercicioC.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

```



```

using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;

public partial class EjercicioC : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (IsPostBack == false)
        {
            XmlDocument midoc = new XmlDocument();
            XmlTextReader milector = new
            XmlTextReader(Server.MapPath("Inventario2.xml"));
            milector.WhitespaceHandling = WhitespaceHandling.None;
            midoc.Load(milector);
            Session.Add("DocumentoEnSesion", midoc);
            for (int i = 0; i < midoc.DocumentElement.ChildNodes.Count; i++)
            {

                ddlProducto.Items.Add(midoc.DocumentElement.ChildNodes[i].ChildNodes[1].InnerText
                );
            }
            milector.Close();
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            int n;
            int cantidad = 1;
            double precio;
            XmlDocument midoc;
            midoc = (XmlDocument)Session["DocumentoEnSesion"];
            n = ddlProducto.SelectedIndex;
            cantidad = Convert.ToInt32(txtCantidad.Text);
            precio =
            Convert.ToDouble(midoc.DocumentElement.ChildNodes[n].ChildNodes[2].InnerText);
            lblPrecio.Text = precio.ToString();
            lblTotal.Text = (cantidad * precio).ToString();
        }
    }
}

```

4.0 Documento XML

```

<?xml version="1.0" encoding="utf-8" ?>

<Inventario>

<Producto Color="Rojo" SuministradorId="2">

<ProductId>1</ProductId>

<ProductoNombre>Peras</ProductoNombre>

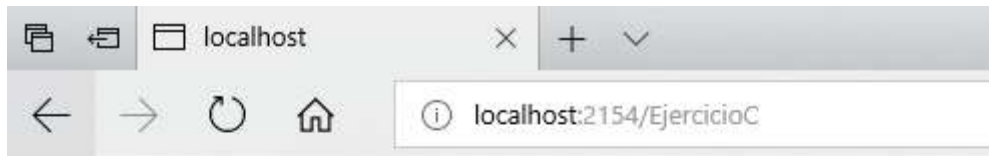
<ProductoPrecio>3,12</ProductoPrecio>

```

</Producto>

</Inventario>

5.0 Resultado



DOM: Elija un producto

Producto: Cantidad

Precio individual
3,12

Precio Total: 31,2

Resumen segunda parte de la materia: Ejercicios Mobile

Contenido

Ejercicio 1: Calculadora.....	3
Ejercicio 2: Votación.....	6
Ejercicio 3: Calcular IMC.....	10
Ejercicio 4: Carrito de compras.....	15

Ejercicio 1: Calculadora

Interfaz



```
<Page
  x:Class="App1.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:App1"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
  Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

  <Grid>
    <TextBlock x:Name="textBlock" HorizontalAlignment="Left"
      Margin="67,142,0,0" TextWrapping="Wrap" Text="Operador 1" VerticalAlignment="Top"
      Height="21" Width="140" FontSize="20"/>
    <TextBox x:Name="txtNum1" HorizontalAlignment="Left" Margin="244,142,0,0"
      TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="89"/>
    <TextBox x:Name="txtNum2" HorizontalAlignment="Left" Margin="244,210,0,0"
      TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="89"/>
    <TextBlock x:Name="textBlock1" HorizontalAlignment="Left"
      Margin="67,225,0,0" TextWrapping="Wrap" Text="Operador 2" VerticalAlignment="Top"
      FontSize="20" RenderTransformOrigin="0.443,-0.422" Width="122"/>
```

```

        <ComboBox x:Name="cmb" HorizontalAlignment="Left" Margin="67,308,0,0"
VerticalAlignment="Top" Width="266">
        <ComboBoxItem Content="+" HorizontalAlignment="Left"
Width="114.999992370605"/>
        <ComboBoxItem Content="-" HorizontalAlignment="Left"
Width="114.999992370605"/>
        <ComboBoxItem Content="*" HorizontalAlignment="Left"
Width="114.999992370605"/>
        <ComboBoxItem Content="/" HorizontalAlignment="Left"
Width="114.999992370605"/>
        </ComboBox>
        <TextBox x:Name="txtResult" HorizontalAlignment="Left"
Margin="67,440,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
TextChanged="txtResult_TextChanged" Width="266"/>
        <Button x:Name="btnCalcular" Content="Calcular"
HorizontalAlignment="Left" Margin="67,534,0,0" VerticalAlignment="Top"
Width="266" Click="btnCalcular_Click"/>

    </Grid>
</Page>

```

MainPage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

// La plantilla de elemento Página en blanco está documentada en
http://go.microsoft.com/fwlink/?LinkId=391641

namespace App1
{
    /// <summary>
    /// Página vacía que se puede usar de forma independiente o a la que se puede
    navegar dentro de un objeto Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();

            this.NavigationCacheMode = NavigationCacheMode.Required;
        }

        /// <summary>

```

```

    /// Se invoca cuando esta página se va a mostrar en un objeto Frame.
    /// </summary>
    /// <param name="e">Datos de evento que describen cómo se llegó a esta
página.
    /// Este parámetro se usa normalmente para configurar la página.</param>
    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
        // TODO: Preparar la página que se va a mostrar aquí.

        // TODO: Si la aplicación contiene varias páginas, asegúrese de
        // controlar el botón para retroceder del hardware registrándose en
e1
        // evento Windows.Phone.UI.Input.HardwareButtons.BackPressed.
        // Si usa NavigationHelper, que se proporciona en algunas plantillas,
        // el evento se controla automáticamente.
    }

    double num1, num2, result;

    private void txtResult_TextChanged(object sender, TextChangedEventArgs e)
    {

    }

    string operacion;

    private void btnCalcular_Click(object sender, RoutedEventArgs e)
    {

        num1 = Convert.ToDouble(txtNum1.Text);
        num2 = Convert.ToDouble(txtNum2.Text);

        operacion = cmb.SelectedIndex.ToString();
        switch(operacion)
        {
            case "0":
                result = num1 + num2;
                break;

            case "1":
                result = num1 - num2;
                break;
            case "2":
                result = num1 * num2;
                break;
            case "3":
                result = num1 / num2;
                break;

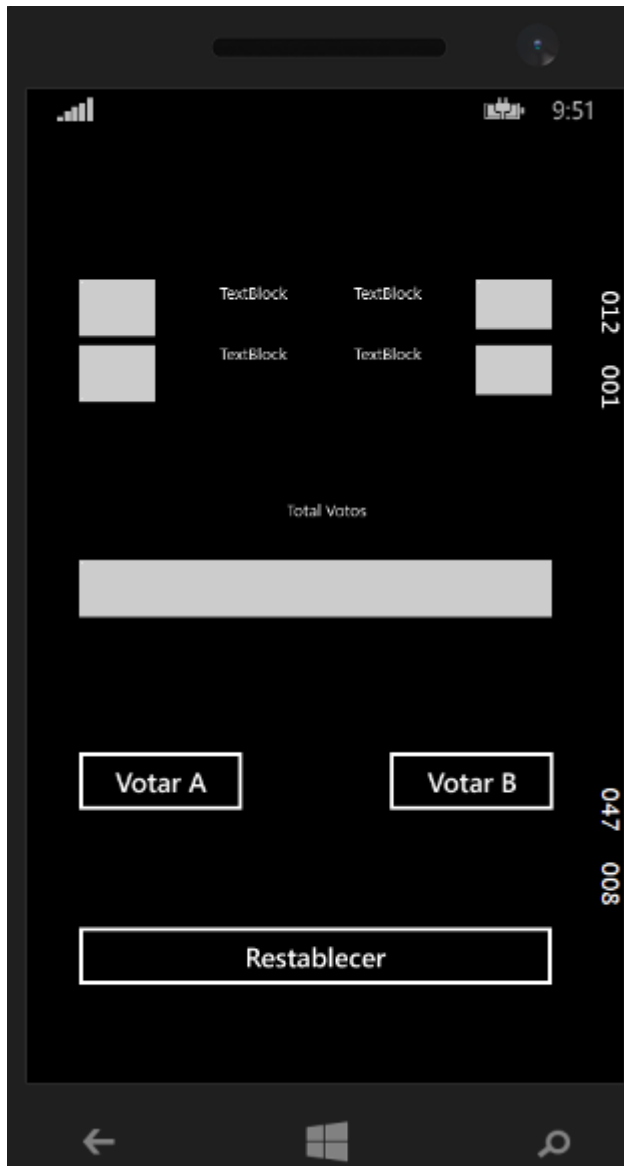
        }

        txtResult.Text = result.ToString();
    }
}
}
}

```

Ejercicio 2: Votación

Interfaz



```
<Page
  x:Class="App2.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:App2"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
```

```

        Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

        <Grid>
            <TextBox x:Name="txtA" HorizontalAlignment="Left" Margin="35,101,0,0"
                TextWrapping="Wrap" Text="" VerticalAlignment="Top"/>
            <TextBox x:Name="txtB" HorizontalAlignment="Left" Margin="35,145,0,0"
                TextWrapping="Wrap" Text="" VerticalAlignment="Top"/>
            <TextBlock x:Name="textBlock" HorizontalAlignment="Left"
                Margin="129,104,0,0" TextWrapping="Wrap" Text="TextBlock"
                VerticalAlignment="Top"/>
            <TextBlock x:Name="textBlock1" HorizontalAlignment="Left"
                Margin="129,145,0,0" TextWrapping="Wrap" Text="TextBlock"
                VerticalAlignment="Top"/>
            <TextBlock x:Name="textBlock2" HorizontalAlignment="Left"
                Margin="219,104,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top"
                Height="26"/>
            <TextBlock x:Name="textBlock3" HorizontalAlignment="Left"
                Margin="219,145,0,0" TextWrapping="Wrap" Text="TextBlock"
                VerticalAlignment="Top"/>
            <TextBox x:Name="txtPorcentajeA" HorizontalAlignment="Left"
                Margin="301,101,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
                Height="23" RenderTransformOrigin="-0.248,0.163"/>
            <TextBox x:Name="txtPorcentajeB" HorizontalAlignment="Left"
                Margin="301,145,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
                Height="31" RenderTransformOrigin="1.321,1.976"/>
            <TextBlock x:Name="textBlock4" HorizontalAlignment="Left"
                Margin="174,250,0,0" TextWrapping="Wrap" Text="Total Votos"
                VerticalAlignment="Top"/>
            <TextBox x:Name="txtTotal" HorizontalAlignment="Left" Margin="35,289,0,0"
                TextWrapping="Wrap" Text="" VerticalAlignment="Top" RenderTransformOrigin="-
                1.292,-0.222" Width="317"/>
            <Button x:Name="btnA" Content="Votar A" HorizontalAlignment="Left"
                Margin="35,409,0,0" VerticalAlignment="Top" Click="btnA_Click"/>
            <Button x:Name="btnB" Content="Votar B" HorizontalAlignment="Left"
                Margin="243,409,0,0" VerticalAlignment="Top" Click="btnB_Click"/>
            <Button x:Name="btnRestablecer" Content="Restablecer"
                HorizontalAlignment="Left" Margin="35,527,0,0" VerticalAlignment="Top"
                Width="317" Click="btnRestablecer_Click"/>

        </Grid>
    </Page>

```

MainPage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;

```



```

using Windows.UI.Xaml.Navigation;

// La plantilla de elemento Página en blanco está documentada en
http://go.microsoft.com/fwlink/?LinkId=391641

namespace App2
{
    /// <summary>
    /// Página vacía que se puede usar de forma independiente o a la que se puede
    navegar dentro de un objeto Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {

        private static int total;
        private static int totalA;
        private static int totalB;

        //Constructor
        public MainPage()
        {

            initParams();

            this.InitializeComponent();

            this.NavigationCacheMode = NavigationCacheMode.Required;
        }

        private void initParams()
        {

            total = 0;
            totalA = 0;
            totalB = 0;
            ///txtA.Text = "";
            //txtB.Text = "";
            //txtPorcentajeA.Text = "";
            //txtPorcentajeB.Text = "";
            //txtTotal.Text = "";

        }

        private void initParamsComplete()
        {

            total = 0;
            totalA = 0;
            totalB = 0;
            txtA.Text = "";
            txtB.Text = "";
            txtPorcentajeA.Text = "";
            txtPorcentajeB.Text = "";
            txtTotal.Text = "";

        }

        /// <summary>
        /// Se invoca cuando esta página se va a mostrar en un objeto Frame.
        /// </summary>
    }
}

```

```

    /// <param name="e">Datos de evento que describen cómo se llegó a esta
página.
    /// Este parámetro se usa normalmente para configurar la página.</param>
    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
        // TODO: Preparar la página que se va a mostrar aquí.

        // TODO: Si la aplicación contiene varias páginas, asegúrese de
        // controlar el botón para retroceder del hardware registrándose en
el
        // evento Windows.Phone.UI.Input.HardwareButtons.BackPressed.
        // Si usa NavigationHelper, que se proporciona en algunas plantillas,
        // el evento se controla automáticamente.
    }

    private void btnA_Click(object sender, RoutedEventArgs e)
    {
        total++;
        totalA++;
        viewResults();
    }

    private void btnB_Click(object sender, RoutedEventArgs e)
    {
        total++;
        totalB++;
        viewResults();
    }

    private void viewResults()
    {
        txtA.Text = totalA.ToString();
        txtPorcentajeA.Text = Convert.ToString((totalA * 100) / total) + "%";

        txtB.Text = totalB.ToString();
        txtPorcentajeB.Text = Convert.ToString((totalB * 100) / total) + "%";

        txtTotal.Text = total.ToString();

    }

    private void btnRestablecer_Click(object sender, RoutedEventArgs e)
    {
        initParamsComplete();
    }
}

```

Ejercicio 3: Calcular IMC

Interfaz

The screenshot shows a mobile application interface for calculating BMI (IMC). The interface is dark-themed. At the top, there is a status bar with signal strength, battery level, and the time 11:06. Below the status bar, there are two input fields: "Peso (kg)" and "Altura (mts)". Below these, there is a label "Estructura corporal:" followed by a dropdown menu with the text "elige una opción". Below that, there is a label "Sexo" followed by another dropdown menu with the text "elige una opción". At the bottom, there is a button labeled "Calcular IMC". The bottom navigation bar shows a back arrow, the Windows logo, and a search icon.

```
<Page
  x:Class="App3.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:App3"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
  Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
```

```

<Grid>
    <TextBlock x:Name="textBlock" HorizontalAlignment="Left"
Margin="61,74,0,0" TextWrapping="Wrap" Text="Peso (kg)" VerticalAlignment="Top"
FontSize="20"/>
    <TextBlock x:Name="textBlock1" HorizontalAlignment="Left"
Margin="61,118,0,0" TextWrapping="Wrap" Text="Altura (mts)"
VerticalAlignment="Top" FontSize="20"/>
    <TextBox x:Name="txtPeso" HorizontalAlignment="Left" Margin="218,74,0,0"
TextWrapping="Wrap" Text="" VerticalAlignment="Top" Height="39" Width="112"/>
    <TextBox x:Name="txtAltura" HorizontalAlignment="Left"
Margin="218,118,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
Height="39" Width="112"/>
    <TextBlock x:Name="textBlock2" HorizontalAlignment="Left"
Margin="10,227,0,0" TextWrapping="Wrap" Text="Estructura corporal:"
VerticalAlignment="Top" FontSize="20"/>
    <ComboBox x:Name="cmbSexo" HorizontalAlignment="Left"
Margin="121,409,0,0" VerticalAlignment="Top" Width="209">
        <ComboBoxItem Content="Femenino" HorizontalAlignment="Left"
Width="114.999992370605"/>
        <ComboBoxItem Content="Masculino" HorizontalAlignment="Left"
Width="114.999992370605"/>
    </ComboBox>
    <TextBlock x:Name="textBlock3" HorizontalAlignment="Left"
Margin="45,419,0,0" TextWrapping="Wrap" Text="Sexo" VerticalAlignment="Top"
FontSize="20"/>
    <Button x:Name="btnCalcular" Content="Calcular IMC"
HorizontalAlignment="Left" Margin="10,582,0,0" VerticalAlignment="Top"
Click="btnCalcular_Click"/>
    <TextBox x:Name="txtResultado" HorizontalAlignment="Left"
Margin="174,536,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
Width="184"/>
    <TextBox x:Name="txtResultadoTextual" HorizontalAlignment="Left"
Margin="174,592,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
Width="184"/>
    <ComboBox x:Name="cmbEstructuraCorporal" HorizontalAlignment="Left"
Margin="45,246,0,0" VerticalAlignment="Top" Width="233">
        <ComboBoxItem Content="Grande" HorizontalAlignment="Left"
Width="228.33332824707"/>
        <ComboBoxItem Content="Mediano" HorizontalAlignment="Left"
Width="228.33332824707"/>
        <ComboBoxItem Content="Chico" HorizontalAlignment="Left"
Width="228.33332824707"/>
    </ComboBox>

</Grid>
</Page>

```

MainPage.xaml

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;

```

```

using Windows.UI.Xaml.Navigation;

// La plantilla de elemento Página en blanco está documentada en
http://go.microsoft.com/fwlink/?LinkId=391641

namespace App3
{
    /// <summary>
    /// Página vacía que se puede usar de forma independiente o a la que se puede
    navegar dentro de un objeto Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        private static double altura;
        private static double peso;
        private static double total;
        private static string Resultado;
        private static string EstructuraCorporal;
        private static string Sexo;
        public MainPage()
        {
            initParams();

            this.InitializeComponent();

            this.NavigationCacheMode = NavigationCacheMode.Required;
        }

        private void initParams()
        {
            altura = 0;
            peso = 0;
            total = 0;
        }

        /// <summary>
        /// Se invoca cuando esta página se va a mostrar en un objeto Frame.
        /// </summary>
        /// <param name="e">Datos de evento que describen cómo se llegó a esta
        página.
        /// Este parámetro se usa normalmente para configurar la página.</param>
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            // TODO: Preparar la página que se va a mostrar aquí.

            // TODO: Si la aplicación contiene varias páginas, asegúrese de
            // controlar el botón para retroceder del hardware registrándose en
            // evento Windows.Phone.UI.Input.HardwareButtons.BackPressed.
            // Si usa NavigationHelper, que se proporciona en algunas plantillas,
            // el evento se controla automáticamente.
        }

        private void btnCalcular_Click(object sender, RoutedEventArgs e)
        {

```

```

peso = Convert.ToDouble(txtPeso.Text);
altura = Convert.ToDouble(txtAltura.Text);

total = (peso / (Math.Pow(altura, 2)));

EstructuraCorporal = cmbEstructuraCorporal.SelectedIndex.ToString();

switch(EstructuraCorporal)
{
    case "0":
        total = total + ((total * 10) / 100);
        break;

    case "1":
        total = total + ((total * 5) / 100);
        break;
    case "2":
        total = total + ((total * 1) / 100);
        break;

}

Sexo = cmbSexo.SelectedIndex.ToString();
switch(Sexo)
{
    case "0":
        total = total - ((total * 10) / 100);
        break;

    case "1":
        break;
}

txtResultado.Text = total.ToString();


if(total < 18)
{
    Resultado = "Peso Bajo";
}
else if(total >= 18 && total <= 24.9)
{
    Resultado = "Normal";
}
else if(total >= 25 && total <= 26.9)
{
    Resultado = "Sobrepeso";
}
else if(total > 27 && total <= 29.9)
{
    Resultado = "Obeso";
}
else if (total > 29.9 && total <= 39.9)
{
    Resultado = "Obeso I";
}

```

```

    }
    else if (total > 39.9)
    {
        Resultado = "Obeso II";
    }

    txtResultadoTextual.Text = Resultado;

}
}
}

```

Resultado

021 000

043 006

Peso (kg) 85

Altura (mts) 1,75

Estructura corporal:

Mediano

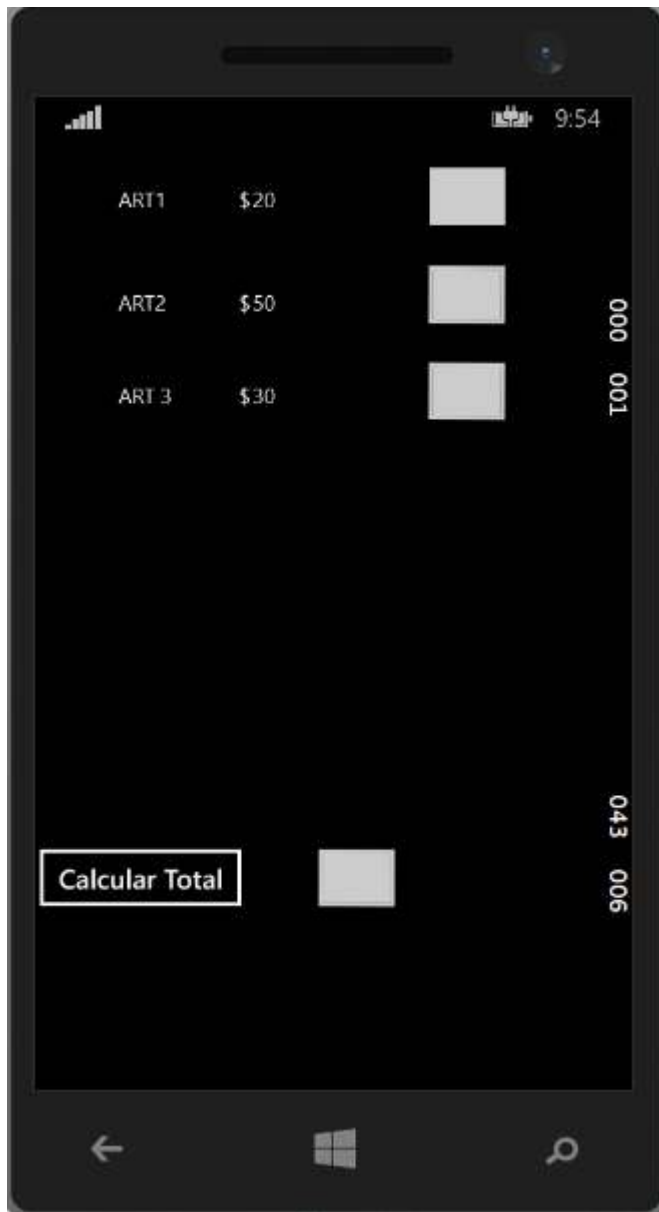
Sexo Masculino

29,1428571428571

Calcular IMC Obeso

Ejercicio 4: Carrito de compras

Interfaz:



```
<Page
  x:Class="App11.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:App11"
```



```

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d"
Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid RenderTransformOrigin="0.463,0.564">
    <TextBox x:Name="textBoxart1" HorizontalAlignment="Left"
Margin="264,21,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"/>
    <TextBox x:Name="textBoxart2" HorizontalAlignment="Left"
Margin="263.787,86.721,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
RenderTransformOrigin="0.5,0.5" UseLayoutRounding="False"
d:LayoutRounding="Auto">
        <TextBox.RenderTransform>
            <CompositeTransform Rotation="0.226"/>
        </TextBox.RenderTransform>
    </TextBox>
    <Button x:Name="button" Content="Calcular Total"
HorizontalAlignment="Left" Margin="3,469,0,0" VerticalAlignment="Top"
RenderTransformOrigin="0.506,1.265" Click="button_Click"/>
    <TextBox x:Name="textBoxTot" HorizontalAlignment="Left"
Margin="190.079,478.399,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
RenderTransformOrigin="0.5,0.5" UseLayoutRounding="False"
d:LayoutRounding="Auto">
        <TextBox.RenderTransform>
            <CompositeTransform Rotation="0.226"/>
        </TextBox.RenderTransform>
    </TextBox>
    <TextBlock x:Name="textBlockart1" HorizontalAlignment="Left"
Margin="56,34,0,0" TextWrapping="Wrap" Text="ART1" VerticalAlignment="Top"
Height="20" Width="74" FontSize="14"/>
    <TextBlock x:Name="textBlockart2" HorizontalAlignment="Left"
Margin="56,103,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
SelectionChanged="textBlock_Copy_SelectionChanged" Text="ART2" FontSize="14"/>
    <TextBlock x:Name="textBlockart3" HorizontalAlignment="Left"
Margin="56,166,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
SelectionChanged="textBlock_Copy_SelectionChanged" FontSize="14">
        <Run Text="ART"/>
        <Run Text="3"/>
    </TextBlock>
    <TextBox x:Name="textBoxart3" HorizontalAlignment="Left"
Margin="263.787,151.599,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
RenderTransformOrigin="0.5,0.5" UseLayoutRounding="False"
d:LayoutRounding="Auto">
        <TextBox.RenderTransform>
            <CompositeTransform Rotation="0.226"/>
        </TextBox.RenderTransform>
    </TextBox>
    <TextBlock x:Name="textBlock_pre_art1" HorizontalAlignment="Left"
Margin="146,34,0,0" TextWrapping="Wrap" Text="20" VerticalAlignment="Top"
Height="20" Width="74" FontSize="14"/>
    <TextBlock x:Name="textBlock_pre_art2" HorizontalAlignment="Left"
Margin="146,103,0,0" TextWrapping="Wrap" Text="50" VerticalAlignment="Top"
Height="20" Width="74" FontSize="14"/>
    <TextBlock x:Name="textBlock_pre_art3" HorizontalAlignment="Left"
Margin="146,166,0,0" TextWrapping="Wrap" Text="30" VerticalAlignment="Top"
Height="20" Width="74" FontSize="14"/>
    <TextBlock x:Name="textBlock_pre_art1_Copy2" HorizontalAlignment="Left"
Margin="137,166,0,0" TextWrapping="Wrap" Text="$" VerticalAlignment="Top"
Height="20" Width="74" FontSize="14"/>
    <TextBlock x:Name="textBlock_pre_art1_Copy" HorizontalAlignment="Left"
Margin="137,103,0,0" TextWrapping="Wrap" Text="$" VerticalAlignment="Top"
Height="20" Width="74" FontSize="14"/>

```

```

        <TextBlock x:Name="textBlock_pre_art1_Copy1" HorizontalAlignment="Left"
Margin="137,34,0,0" TextWrapping="Wrap" Text="$" VerticalAlignment="Top"
Height="20" Width="74" FontSize="14"/>
    </Grid>
</Page>

```

MAINPAGE.XAML.CS

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
// La plantilla de elemento Página en blanco está documentada en
http://go.microsoft.com/fwlink/?LinkId=391641

```

```

namespace App1
{
    /// <summary>
    /// Página vacía que se puede usar de forma independiente o
    a la que se puede navegar dentro de un objeto Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();

            this.NavigationCacheMode =
NavigationCacheMode.Required;
        }

        int cant1, cant2, cant3, pre1, pre2, pre3;
        double resulttot;
        int operacionart1, operacionart2, operacionart3;

        /// <summary>

```

```

        /// Se invoca cuando esta página se va a mostrar en un
objeto Frame.
        /// </summary>
        /// <param name="e">Datos de evento que describen cómo
se llegó a esta página.
        /// Este parámetro se usa normalmente para configurar la
página.</param>
        protected override void
OnNavigatedTo(NavigationEventArgs e)
        {
            // TODO: Preparar la página que se va a mostrar
aquí.

            // TODO: Si la aplicación contiene varias páginas,
asegúrese de
            // controlar el botón para retroceder del hardware
registrándose en el
            // evento
Windows.Phone.UI.Input.HardwareButtons.BackPressed.
            // Si usa NavigationHelper, que se proporciona en
algunas plantillas,
            // el evento se controla automáticamente.
        }

        private void comboBox1_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
        }

        private void button_Click(object sender, RoutedEventArgs
e)
        {
            cant1 = Convert.ToInt32(textBoxart1.Text);
            cant2 = Convert.ToInt32(textBoxart2.Text);
            cant3 = Convert.ToInt32(textBoxart3.Text);
            pre1 = Convert.ToInt32(textBlock_pre_art1.Text);
            pre2 = Convert.ToInt32(textBlock_pre_art2.Text);
            pre3 = Convert.ToInt32(textBlock_pre_art3.Text);

            operacionart1 = pre1 * cant1;
            operacionart2 = pre2 * cant2;
            operacionart3 = pre3 * cant3;

            resulttot = operacionart1 + operacionart2 +
operacionart3;

            textBoxTot.Text = resulttot.ToString();

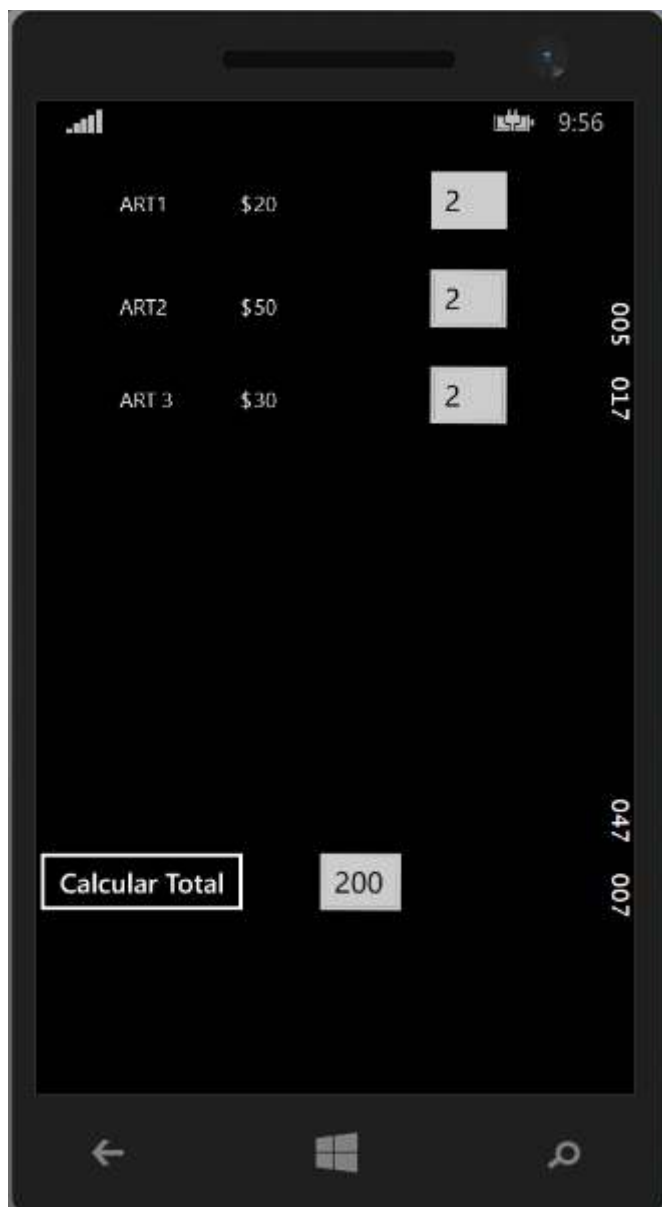
```

```

    }
}
}

```

Ejemplo:



Trabajo práctico CORBA completo

Contenido

1.	Presentación.....	3
2.	Servicios CORBA.....	3
2.1	Servicio de Nombres.....	3
2.2	Servicio de Control de Concurrencia.....	5
2.3	Servicio de Ciclo de Vida.....	6
2.4	Servicio de eventos.....	7
2.5	Servicio de negociación.....	8
2.6	Servicio de seguridad.....	8
2.7	Servicio de Relación.....	9
2.8	Servicio de Persistencia.....	10
2.9	Servicio de Transacción.....	10
2.10	Servicio de Externalización.....	10
2.11	Servicio de Licencia.....	11
2.12	Servicio de Consulta.....	11
2.13	Servicio de Propiedad.....	12
2.14	Servicio de Tiempo.....	12
	Presentación.....	13

1. Presentación

En el presente trabajo veremos en profundidad los diversos Servicios que componen al estándar CORBA (Common Object Request Broker Architecture). Para esto detallaremos esquemas de los diseños de cada servicio e incorporaremos ejemplos prácticos.

2. Servicios CORBA

Los servicios de CORBA definen un conjunto de servicios de bajo nivel que permiten a los Objetos de Aplicación comunicarse de una forma estándar.

Para obtener la referencia a uno de estos servicios (por ejemplo el Servicio de Nombres) se utiliza:

```
org.omg.CORBA.Object objeto = orb.resolve_initial_references("NameService");
NamingContext ns = NamingContextHelper.narrow(objeto);
```

2.1 Servicio de Nombres

Asocia nombres a objetos dentro de contextos de nombres.

Asocia nombres a objetos dentro de contextos de nombres.

Asocia nombres a objetos dentro de contextos de nombres.

Asocia nombres a objetos dentro de contextos de nombres.

Asocia nombres a objetos dentro de contextos de nombres

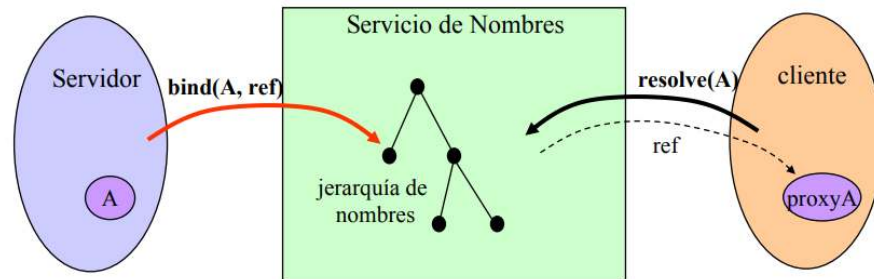
Los Servicios de Nombres distribuidos se estructuran en grafos de contextos de nombres (como estructuras de directorios en sistemas de ficheros), facilitando la administración y la escalabilidad

- Cada objeto tiene un único ID (o referencia)
- Opcionalmente se pueden asociar uno o más nombres a una referencia
- Siempre se puede definir un nombre relativo a su contexto

El Servicio de Nombres guarda pares <**nombre, referencia a un objeto**>

El Servicio de Nombres es usado por cliente y servidor

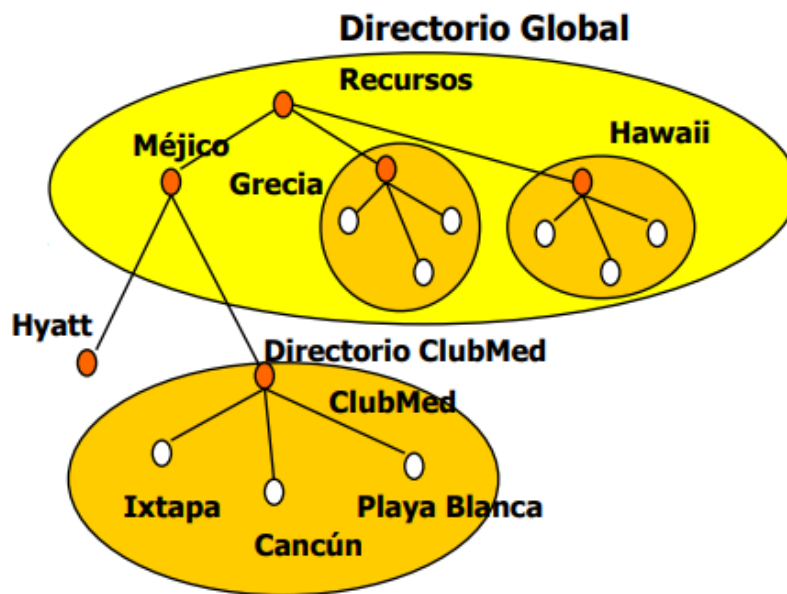
- El servidor asocia (bind) en el Servicio de Nombres una referencia a objeto con un nombre
- El cliente puede pedirle al Servicio de Nombres que a partir de un nombre le dé (resuelve) una referencia a un objeto CORBA



Los nombres están organizados jerárquicamente

Un nodo en la jerarquía puede ser:

- Un contexto de nombres
- Un nombre (necesariamente nodo hoja)



Ejemplo Interfaz NamingContext

```

interface NamingContext {
    NamingContext new_context();
    void destroy() raises (NotEmpty);
    Object resolve (in Name n) raises (NotFound, CannotProceed, InvalidName);
    void list (in unsigned long how_many, out BindingList b,
              out BindingIterator bi);
    void unbind(in Name n)
        raises (NotFound, CannotProceed, InvalidName);
    void bind(in Name n, in Object obj)
        raises (NotFound, CannotProceed, InvalidName, AlreadyBound);
    void rebind(in Name n, in Object obj)
        raises (NotFound, CannotProceed, InvalidName);
    void bind_context(in Name n, in NamingContext nc)
        raises (NotFound, CannotProceed, InvalidName, AlreadyBound);
    void rebind_context(in Name n, in NamingContext nc)
        raises (NotFound, CannotProceed, InvalidName);
    void bind_new_context(in Name n)
        raises (NotFound, CannotProceed, InvalidName, AlreadyBound);
};

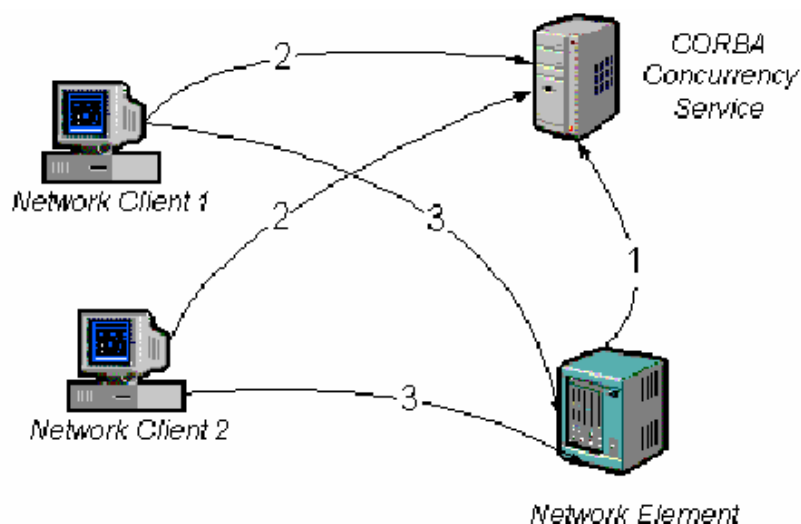
```

2.2 Servicio de Control de Concurrency

El servicio de control de concurrency provee las interfaces para fijar y liberar bloques que permitan a clientes múltiples coordinar su acceso a recursos compartidos.

Un cliente del servicio de control de concurrency puede escoger la forma de adquirir los bloqueos en una de dos formas:

- A favor de una transacción: El servicio se encarga de liberar los bloqueos cuando la transacción termina o aborta.
- A favor de un cliente sin transacciones: La responsabilidad de liberar los bloqueos recae sobre el cliente.



El servicio de control de concurrency define un lockset que es un conjunto de bloqueos asociado con un recurso simple. A su vez define un coordinador de bloqueos que gestiona la liberación de los locksets relacionados.

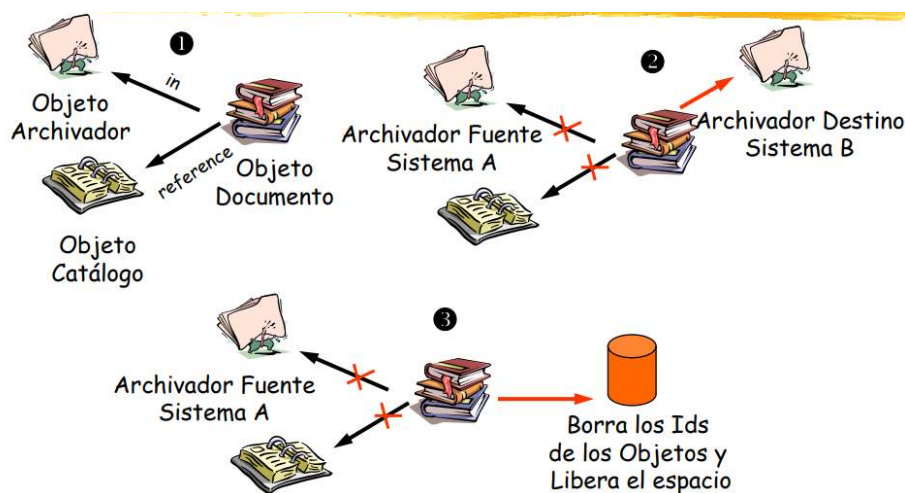
Las interfaces para gestionar el control de concurrency son:

- LocksetFactory: permite crear locksets.
- Lockset: permite adquirir y liberar bloqueos.
- TransactionalLockset: soporta los mismos métodos que Lockset. La diferencia es que se debe pasar un parámetro que identifica a la transacción.
- LockCoordinator: OTS invoca su método para liberar todos los bloqueos mantenidos por una transacción cuando aborta o se ejecuta.

2.3 Servicio de Ciclo de Vida

El servicio de ciclo de vida de objetos provee operaciones para crear, copiar, mover y borrar objetos. Todas estas operaciones pueden así soportar asociaciones entre grupos de objetos relacionados. Esto incluye relaciones de referencia y contenido. El servicio de ciclo de vida provee interfaces que son derivadas del servicio de relaciones.

Para crear un nuevo objeto un cliente en principio debe encontrar una fábrica de objetos, ejecutar una petición de creación, y recibir de regreso una referencia a un objeto.



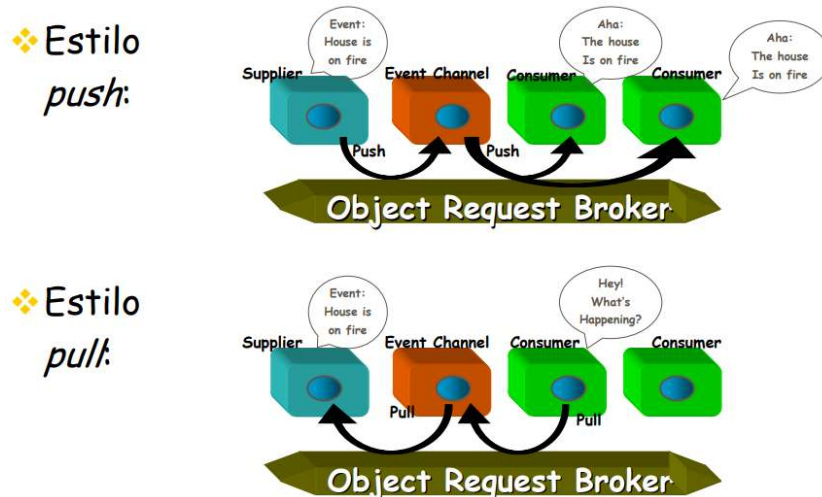
Para definir una relación, primero se debe asociar un papel con un objeto y entonces crear las relaciones entre estos papeles.

2.4 Servicio de eventos

El servicio de eventos permite a los objetos registrar y eliminar dinámicamente sus eventos específicos de interés. Un evento es una ocurrencia dentro de un objeto específico que puede ser de interés para uno o más objetos. Una notificación es un mensaje que un objeto envía a los grupos de interés para informales que un evento específico ha ocurrido sin conocerle, ya que esto es normalmente gestionado por el servicio de eventos.

El servicio de eventos mantiene la comunicación entre objetos. El servicio define dos papeles para los objetos: el de proveedores y el de consumidores.

Hay dos modelos para la comunicación de eventos de datos: empujar (push) y tirar (pull). En el modelo empujar el proveedor de eventos toma la iniciativa e inicia la transferencia de datos a los consumidores. En el modelo tirar, el consumidor toma la iniciativa y requiere los datos de eventos de un proveedor.

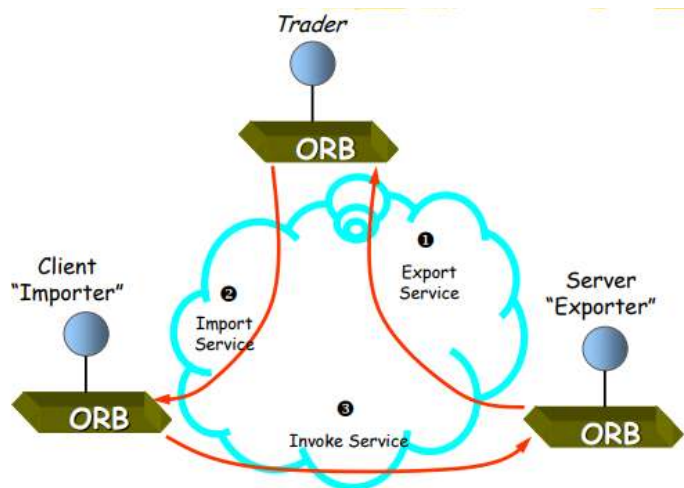


El servicio de eventos es extremadamente útil ya que provee las bases para la creación de un nuevo género de aplicaciones de objetos distribuidos.

2.5 Servicio de negociación

El objeto servicio de negociación almacena una referencia a un objeto y una descripción del tipo de servicio que se ofrece.

Los clientes se ponen en contacto con el servicio de negociación para averiguar los servicios listados y preguntar por un servicio por su tipo. El servicio de negociación encontrará el más apropiado basado en el contexto del servicio requerido y la oferta de sus proveedores. Los servicios de negociación de diferentes dominios pueden crear federaciones.

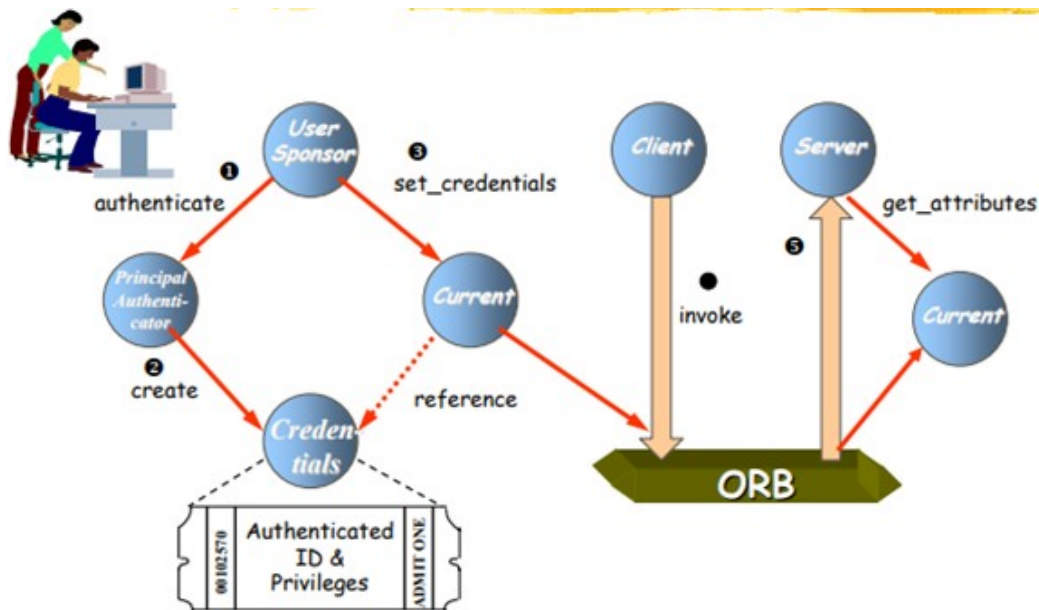


2.6 Servicio de seguridad

Un usuario es autenticado una vez por el ORB y le es dado un papel, selecciona derechos y seguridad. De la autenticación se genera un único identificador de verificación que no puede ser cambiado por un objeto; solo el servidor de autenticación puede cambiarlo.

Una vez que los clientes son autenticados, los objetos del servidor son responsables de verificar que operaciones le son permitidas desempeñar a los clientes en la información que ellos tratan de acceder. Los servidores usan listas de control de accesos (ACLs) y una derivación llamada listas de capacidad para controlar el acceso a los usuarios.

Los servicios de auditoria permiten a los administradores del sistema monitorizar los eventos en el ORB, incluyendo intentos de acceso así como que servidores u objetos están siendo usados.



2.7 Servicio de Relación

Permite a los componentes y objetos que no saben nada uno de otro estar relacionados. Permite hacer esto sin cambiar los objetos existentes o requerir que añadan nuevas interfaces, es decir permite crear relaciones dinámicas.

Los tipos de relaciones son:

- Relación de propiedad (ownership)
- Relación de contención (containment)
- Relación de referencia (reference)
- Relación de autoría (authoring)
- Relación de empleo (employment)

2.8 Servicio de Persistencia

El servicio de persistencia de objetos permite a los objetos persistir más allá de la aplicación que creó el objeto o el cliente que lo usa. Permite al estado de un objeto ser salvado en un almacenamiento persistente y restaurarlo en caso de necesitarlo.

2.9 Servicio de Transacción

Semántica transaccional distribuida, con mecanismos de two-phase commit.

2.10 Servicio de Externalización

Define interfaces para exteriorizar un objeto a un flujo e interiorizar un objeto de un flujo. Un flujo es un área que mantiene datos que puede estar en memoria, en un fichero de disco, o a través de una red. Se exterioriza un objeto cuando se requiere regresarlo a la vida en su nuevo destino. Este proceso de dos pasos permite exportar el objeto fuera de su entorno en el ORB.

Los flujos llegan a ser un medio de importación / exportación para los objetos. Permiten copiar y mover objetos y dejan también pasar objetos por valor en un parámetro.

Las interfaces del servicio de exteriorización son:

- Stream exterioriza e interioriza un objeto.
- StreamFactory: crea un flujo.
- FileStreamFactory: crea un flujo basado en un fichero.
- StreamableFactory: crea un objeto que puede interiorizar un flujo.
- StreamIO lee y escribe los contenidos del flujo.
- Streamable: exterioriza e interioriza un objeto a y desde un flujo.

2.11 Servicio de Licencia

Permite el control de la propiedad intelectual.

Pueden definirse políticas de licencia basadas en el tiempo valores o basadas en el consumidor.

Todas las licencias deben tener inicio / duración y fecha de caducidad. Es posible también asignar algunas licencias a usuarios específicos, colecciones de usuarios y

organizaciones. Finalmente el servicio de licencias debe ser un recurso del servidor muy seguro.

Este servicio consiste de dos interfaces que proveen todas las operaciones que un componente necesita para licenciarse / protegerse a si mismo. Ellas son:

- **LicenseServiceManager** sirve para localizar servicios de licencias que implementan políticas específicas.
- **ProducerSpecificLicenseService**: provee tres operaciones que hacen todo el trabajo. Iniciar el uso de la licencia, verificar su estado y terminar el uso de la misma.

2.12 Servicio de Consulta

Este servicio permite encontrar objetos cuyos atributos coincidan con el criterio de búsqueda que se especificó en una consulta. Debe notarse que las consultas no tienen acceso al estado interno del objeto.

Una consulta CORBA hace más que encontrar objetos; permite también manipular una colección de ellos. El servicio de consulta trata a la colección misma como si fuera un objeto y define operaciones que permiten manipular y navegar en una colección. De igual forma permite añadir y eliminar miembros de la colección.

El servicio de consultas provee tres interfaces que permiten manipular los resultados de una consulta, estas son:

- **CollectionFactory**: crea una instancia de una colección vacía.
- **Collection**: define operaciones que permiten añadir, reemplazar y eliminar miembros de una colección.
- **Iterator**: define tres operaciones que permiten invertir una colección.

2.13 Servicio de Propiedad

El servicio de propiedad permite asociar dinámicamente atributos con componentes empaquetados. Se pueden definir estos atributos dinámicos (o propiedades) en tiempo de ejecución sin usar una IDL y luego asociarlos con un objeto que ya exista. Una vez que se definen estas propiedades, se les puede nombrar, fijar y obtener sus valores, determinar sus modos de acceso y eliminarlos.

El servicio de propiedad define cuatro modos de propiedad mutuamente exclusivos: normal, de solo lectura, normal compuesto y de solo lectura compuesto. De igual forma consta de seis interfaces, las dos principales son las siguientes:

- **PropertySet**: define diez operaciones que permite definir, borrar, enumerar, y chequear la existencia de propiedades.
- **PropertySetDef** define ocho operaciones que permiten controlar y modificar los modos de operación.

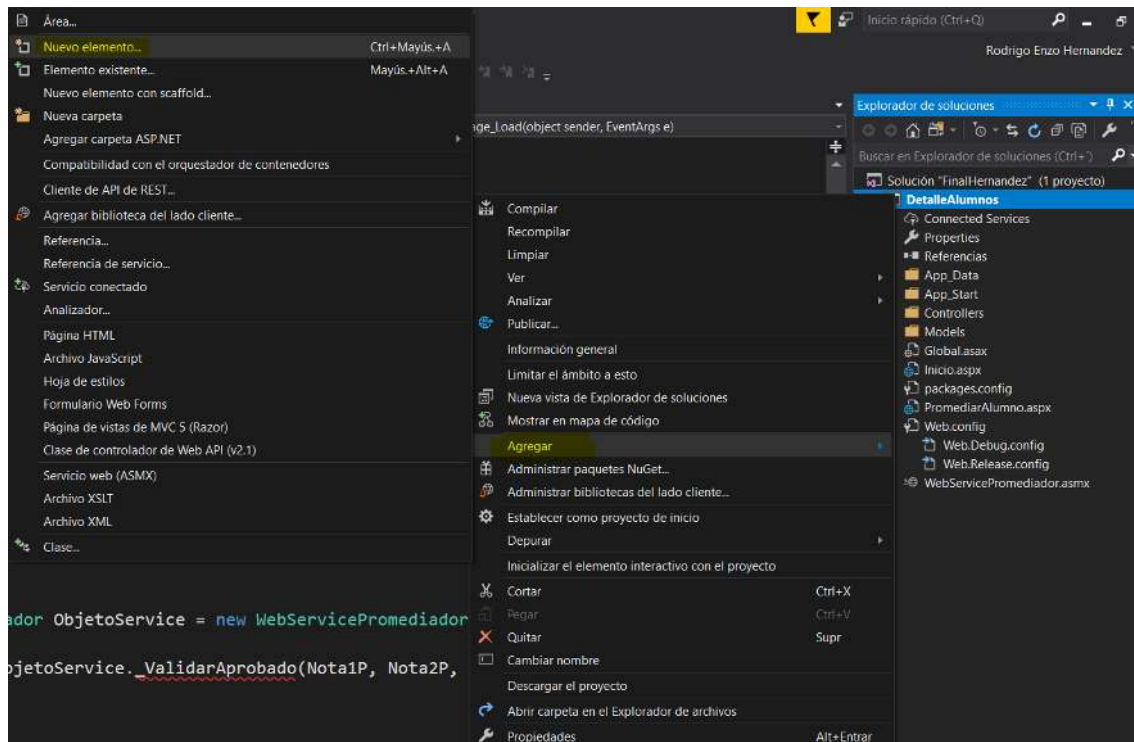
2.14 Servicio de Tiempo

Este servicio permite al ORB mantener sincronizados los relojes de distintas maquinas distribuidas en una red con la finalidad de mantener la nocion del tiempo y poder ejecutar eventos que tendrán que ocurrir en un sistema de objetos distribuido en un orden especifico.

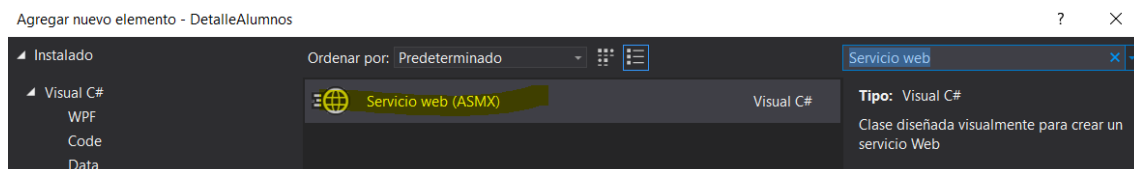
Resumen explicativo como referenciar un WebService

- 1- Click derecho en el proyecto

2- Agregar → Nuevo elemento



3- Buscamos “Servicio Web (ASMX)”

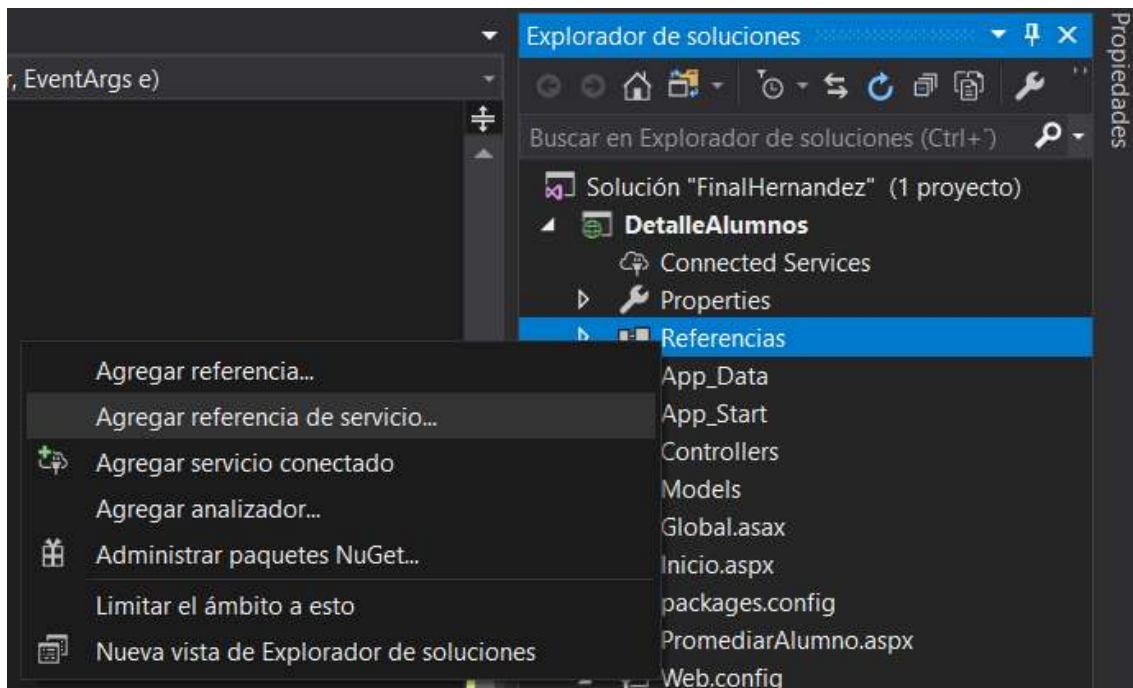


4- Lo agregamos

5- **IMPORTANTE. Compilamos la solución. Sino no vamos a poder referenciar.**

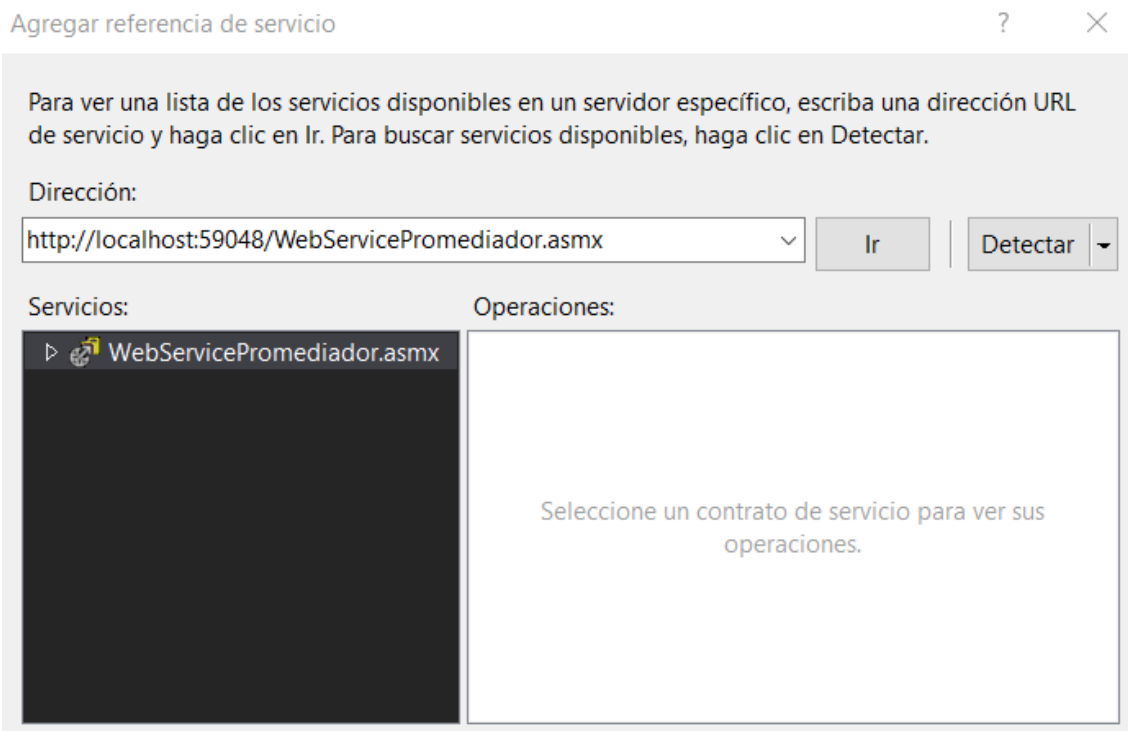
6- Para referenciarlo: Hacemos click derecho en “Referencias”

7- Agregar referencia de servicio



8- Tocamos "Detectar"

9- Nos aparece el WebService creado



10- Cambiamos el nombre de la referencia

11- Damos aceptar

12- Lo instanciamos desde donde lo queremos llamar:

```
WebService ObjetoService = new WebService ();
```