Preparación del entorno:

1. Descargar el fuente de la siguiente ubicación:
2. *~~http://xp-dev.com/svn/sf-produziert/Allus/workshop\_fwk\_10/health~~*

**Dispatchers** :

Despachadores de servicios en las tecnologías WebSerivce y WCF (tcpbinfing)

**Libs**: librerías estáticas del sistema

**Reléase**: Post build de las capas del sistema

**Src**: Codigo fuente del sistema

1. Levantar el proyecto [PROY-FOLDER]\src\HealthW32.sln
2. Reference Path:s Chequee las referencias a las librerías estables para cada uno de los siguientes proyectos:
   1. Health.BE
   2. Health.DAC
   3. Health.SVC
   4. Health.Front
   5. Health.Front.Base
   6. Health.Front.MedEvents
   7. Health.Test

La carpeta de librerías se encuentra en [PROY-FOLDER]\libs

1. Compile el proyecto

Trabajo practico 1: servicios

Desarrollo de servicios de negocio o clases BusinessService<TRequest, TResponse>

El líder del proyecto asigna un el desarrollo de un servicio a cada desarrollador:

* **Servicio 1** 
  + RetrivePatientsService: retorna los pacientes de la base de datos

**Firma**: RetrivePatientsService : BusinessService<RetrivePatientsReq, RetrivePatientsRes>

**Request**: resivira parámetros de filtro com: Nombre y/o Apellido

**Response**: Retorna clase PatientList

* **Servicio** 2
  + GetPacienteService.cs: Obtiene un paciente (FullView) base de datos

**Firma**: GetPacienteService: BusinessService< GetPatientReq, GetPatientReq>

**Request**: parámetros de filtro: **Id**

**Response**: Retorna un PatientBE mas su información de Persona y Mutuales

Se debe rellenar el objeto o propiedad Persona de la instancia PatientBE

Se debe rellenar el objeto o propiedad Mutuales de la instancia PatientBE

* **Servicio 3** CrearPatientService

Crea un paciente

Verifica si existe el paciente (Si Existe lanzar Excepción Funcional)

El Nro documento XXXXX ya pertenece a otro paciente registrado

Puede que la persona exista por Nro de Docuemnto dado que puede ser un profesional del instituto de salud pero aún no este cargado al sistema como paciente. En tal caso la validación deberá permitir el registro o creación del paciente, pero no la persona. En este caso seria una simple asociación

**Firma**: CrearPatientService : BusinessService<CrearPatientReq, CrearPatientRes>

**Request**: parámetros PatientBE

**Response**: Retorna IdPersona y IdPatient creados

* + **Servicio 4** UpdatePatientService

Para esta ocacion solo actualizara las propiedades de paciente y/o persona si esta relacionada.

Solo validamos que el paciente exista si no existe lanzaremos una excepción funcional

**Firma**: UpdatePatientService : BusinessService<UpdatePatientReq, UpdatePatientRes>

**Request**: parámetros PatientBE

**Response**: Retorna DummyContract es desir no retorna nada se utiliza este tipo de objeto para respetar la Arquitectura

Trabajo practico 1: Unit Test

Por cada uno de los servicios realizaremos un breve test unitario en el proyecto Healt.Test

La documentación recomendada para esta practica es: Testing Unitario V.2.0.docx.

Se deberán tener un conocimiento previo sobre nunit .- Los test deben chequear cada funcionalidad del servicio Ejemplo:

*Test1 CreatePtientService\_Erro\_Existente*

*Test 2 CreatePtientService\_Ok*

Material de lectura

Para tener un entendimiento de patrones de la arquitectura se recomienda dar un vistazo general a los siguientes documentos

* *Arquitectura Tecnológica Servicios SOA V.2.0.doc*
* *StoreProcedures.pdf*
* *Servicios.pdf*
* *Namespaces.pdf*
* *Estandares de Codificación V.1.5.pdf*
* *Testing Unitario V.2.0.docx*