5 Programación

Este capítulo sirve como guía de información de la programación de los entregables a desarrollar:

* Transacciones API Transportes (véase en el apartado 5.1)
* Integración (véase en el apartado 5.2)
* Formulario de pruebas (véase en el apartado 5.3)

Se adjuntan como anexo los ficheros correspondientes a estos tres paquetes de software desarrollados.

La solución de la integración contiene también el proyecto de tests unitarios XUNIT (véase en el aparatado 5.4).

5.1 Transacciones de la API de Transportes

En el apartado 5.1.1 de este capítulo, se indica el lenguaje de programación y librerías utilizadas para desarrollar las nuevas transacciones de la API de Transportes tal y como se han representado en el diseño expuesto en el capítulo 4.1 de esta memoria, y en base a los requisitos funcionales y no funcionales enumerados en el capítulo 3.

A continuación, en el apartado 5.2 se cita el código programado y se enumeran los ficheros desarrollados que se adjuntarán como anexo a esta memoria.

En el apartado 5.3 se muestra el formulario de pruebas desarrollado y un ejemplo de uso. Finalmente, en el apartado 5.4 contenido dentro de este capítulo, se explican los escenarios de test desarrollados y su importancia para la mantenibilidad del código.

5.1.1 Lenguaje/tecnología

El lenguaje de programación con el que se programan las nuevas transacciones de la API de Transportes ya citadas en anteriores capítulos: DMR, RMR y EMR, así como los cambios necesarios introducidos en los elementos ya existentes en otras transacciones, es Visual Basic.

Se ha escogido este lenguaje porque el resto de transacciones que ya incluye la API está desarrollado con este mismo lenguaje. Sin embargo, una alternativa válida para desarrollar la misma API desde 0 en otro lenguaje puede ser perfectamente C#. Ofrece las mismas funcionalidades que ya se han utilizado para desarrollar la API de Transportes y además permite crear los ficheros necesarios para subirse en los frontales del servicio wsdl que existe en el sistema de TravelgateX.

Visual Basic (o C#) es necesario para que el cliente pueda atacar a las integraciones por .dll a través del entorno de test que tiene TravelgateX para que éste realice pruebas antes de subir los cambios de la API en el entorno de pre-producción o producción.

Al ser un lenguaje del paradigma de programación “Programación Orientada a Objetos” (POO), se construyen las clases necesarias para la creación de las nuevas llamadas. Todas las clases, elementos y atributos que se detallan en el apartado siguiente (5.1.2) son públicos.

Las tres llamadas desarrolladas son clases serializables, por lo tanto no se pueden heredar. Además las tres deben heredar de las clases padre TransportationBaseRQ para las peticiones y TransportationBaseRS para las respuestas.

Los elementos que incluyen las clases son del tipo XmlElementMultiLanguage y se indican los ElementName para cada idioma (español e inglés). Cuando un elemento tenga el mismo nombre para todos los idiomas con los que se vaya a poder utilizar la API, es importante definirlos con el tipo XmlElement.

Los atributos son del tipo XmlAttributeMultiLanguage y, al igual que los elementos, se indica el nombre del atributo en los dos idiomas.

Los elementos que sean raíz de una transacción xml deben definirse del tipo XmlRootMultiLanguage.

Para la programación de las clases, elementos y atributos, son suficientes las librerías del sistema (Microsoft) XML y librerías internas de la API de Transportes, ya que se van instanciar clases que deben ser heredadas por las nuevas peticiones y respuestas desarrolladas.

5.1.2 Descripción de las transacciones programadas

En este capítulo se detallan los elementos y atributos que contienen las nuevas llamadas de la API de Transportes que se usarán en la integración de Amadeus para modificar reservas. Los elementos y atributos expuestos en las tablas *Tabla 23*, *Tabla 24* y *Tabla 25*, no representan el código en si (programado en Visual Basic), sino la definición (cardinalidad y descripción) de cada uno de ellos. Éste formato es que se ha utilizado para documentar los cambios realizado en la API y que, como se ha establecido en el alcance del proyecto, es uno de los entregables que se han de presentar.

En esta memoria se describen solamente los elementos más importantes de cada transacción. El código de cada una de las transacciones se adjunta como anexo. (¿como pongo el anexo?). En el código, cada uno de los elementos cuenta con una descripción /\*comentario\*/.

Tabla 23: DMR

|  |  |  |
| --- | --- | --- |
| Elemento | Cardinalidad | Significado |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Tabla 24: RMR

|  |  |  |
| --- | --- | --- |
| Elemento | Cardinalidad | Significado |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Tabla 25: EMR

|  |  |  |
| --- | --- | --- |
| Elemento | Cardinalidad | Significado |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

5.2 Integración

En este capítulo se explica el lenguaje y la tecnología utilizada para programar la integración y se enumeran los ficheros desarrollados.

5.2.1 Lenguaje/tecnología

5.2.1 Integración (ficheros)

En principio aquí solo tengo que poner el anexo al código.

5.3 Formulario de pruebas

En el sub-apartado 5.3.1 se describe el lenguaje y tecnología utilizada para programar el formulario de pruebas que se muestra con un ejemplo de uso a través de un vídeo en el apartado 5.3.2.

5.3.1 Lenguaje/tecnología

5.3.2 Ejemplo de uso (vídeo)

Anexo al vídeo

5.4 Programación de Tests Unitarios XUNIT