# 8.- Trabajo desarrollado

## 8.1.-Librería de datos

La librería de datos contiene la implementación de todos los tipos CEN expuestos en la introducción teórica. Cada tipo se ha implementado como una clase java, y más concretamente con características de bean, es decir:

- Todos los atributos son privados.
- Existen métodos de acceso "get" y "set" públicos para todos los atributos.
- Constructor vacío y, como mínimo, otro constructor con todos los atributos de la clase.
- Método **toString()** para la representación por pantalla.

### 8.1.1.- Tipos básicos del CEN

Se han tomado los tipos del proyecto de partida del paquete *PID.BasicTypes*, se les ha dado características de bean. También han sido revisados para que cumplan las especificaciones del CEN completamente. Entre los cambios más significativos están:

- **Numeric**: se han representado con un atributo privado de tipo Number.
- **ST**: el atributo *'character_string'* debería ser del tipo *SET<char>*, pero finalmente se ha implementado como *String*.
- **II**: el atributo *'extension'* había sido implementado de tipo *String* y cambia a tipo *ST*.
- **URL**: todos los atributos anteriormente implementados como *String* pasan a *ST*.
- **IVL_TS**: en la implementación existente se ha cambiado el tipo de los atributos 'low' y 'high' de *Timestamp* a *TS* para que sea un tipo primitivo de los del CEN.
- **CS**: los atributos *'codeValue'* y *'displayName'* habían sido implementados como *String* y se ha cambiado a *ST*.
- **CV**: los atributos *'codingSchemeName', 'codingSchemeVersion', 'originalText'* se habían implementado como *String* y son *ST*. Al heredar del anterior también se ve afectado por los cambios de antes.

Después, y junto con el resto de los tipos que se han implementado a partir de la especificación se han incluido en el paquete **basicTypes.** De los tipos implementados para este proyecto cabe destacar:

- **Char**: se ha optado por crear un atributo privado de tipo *Character* en lugar de un literal *char*, para que se implemente sobre un objeto y no un tipo simple.
- **Date**: se ha expresado con un atributo privado del tipo *java Calendar*.
- **ED**: todos los tipos son correctos excepto el tipo *binary* que se ha representado con un tipo *int*. Afecta a los atributos: *integrityCheck y data*.

Todas las clases son públicas, y heredan de la clase abstracta **DV (DataValue).** Las clases que representan a los tipos básicos compuestos, tienen como variables a tipos primitivos o incluso a otros tipos básicos compuestos.
Las clases contienen además un método:

**public boolean fromXML2Object(Mapping mapa, String documentoXML)**

que permite crear un objeto java a partir del documento XML, sin conocer la clase concreta. Esto se debe a que es un método sobrescrito por todas las clases del

paquete. Así, si se crea un objeto **DV**, del cual heredan todas las clases, se puede invocar éste método, y se creará el objeto de la clase contenida en el documento XML con todos sus campos rellenos con los datos del XML. Basa su funcionamiento en el framework de Castor y se le pasan como parámetros un objeto **org.exolab.castor.mapping.Mapping**, que especificará a Castor como traducir el documento a objeto; y un *String* con el URI del documento XML que se quiere transformar a objeto.
Por otro lado, para la implementación de los tipos que significaban un conjunto de datos se ha optado por una única clase **List**, con la siguiente estructura UML:

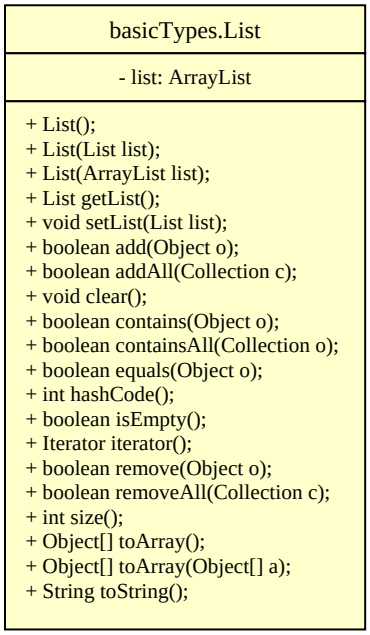


Figura 8.1: Diagrama UML de la clase basicTypes.List

Como se puede observar, se le ha dotado de una serie de métodos típicos de las clases *collection*, muy usadas en Java.

### 8.1.2.- Tipos GPICS

La implementación de los tipos GPICS, constituye una fiel representación de los diagramas de clase UML mostrados en el apartado teórico 3.3. Cada tipo complejo queda encapsulado en una clase Java, teniendo como variables a tipos simples o incluso a tipos complejos CAG.
Se han tomado los tipos que ya habían sido implementados en el proyecto de partida en el paquete PID.CAG_GPIC, se han modificado para que posean características de bean, y se han incluido junto con el resto de las clases java en el paquete **GPICS**, dentro del cual se han diferenciado por su significado 2 subpaquetes: **GPICS.CommonSubComponents** y **GPICS.CAG_GPICS.**

# 8.2.-Pasarela Java-XML mediante el uso de Castor

Una vez implementados los tipos de los cuáles se compone la librería, se pasó a estudiar su conversión a XML mediante el framework de Castor.

## 8.2.1.- Construcción del mapa

Para una correcta conversión de los tipos se optó por implementar mapas de Castor que lo guiaran en la trascripción. Se han creado tres:

- Para los tipos básicos implementados en el paquete basicTypes, se crea **basicTypes.XML.**
- Para los GPICS implementados en el paquete GPICS, se crea **GPICS.XML.**
- Para los tipos de las estructuras básicas del PID, implementados en el paquete PID.DataTypes se crea **PIDs.XML**.

Para referenciar a otro mapa de modo que podamos usar los datos en él descritos, se ha usado el campo *'include'* en cada mapa. Se consigue así dividir el contenido en mapas más pequeños según diferentes criterios (que en éste caso es la pertenencia a una norma u otra) aunque los datos estén relacionados entre sí. Por ello el mapa GPICS.XML incluye al mapa basicTypes.XML mediante:

```
<include href="basicTypes.xml"/>
```

y el mapa PIDs.XML incluye al mapa GPICS.XML (y por extensión de éste a basicTypes.XML):

```
<include href="GPICS.xml"/>
```

En la creación de cada mapa se han usado las reglas descritas en la sección teórica. Cabe destacar que la única clase que representa un conjunto de tipos de datos necesita representarse en el mapa mediante un atributo collection, por lo que su estructura dentro del mapa basicTypes.XML es:

```
<class name="basicTypes.List">
  <map-to xml="List"/>
  <field name="list" type="other" collection="arraylist"/>
</class>
```

Con el tipo igual a other en el campo list, se pretende referir a que el campo list, si bien es de tipo *ArrayList*, contendrá cualquier campo que herede de *Object*, implementándose así una lista de objetos genéricos.

## 8.2.2.- Transformación Java-XML.

Para la transformación de Java a XML se usa siempre las mismas líneas de código:

```
//Creamos el mapping y lo cargamos en su controlador.
Mapping mapa= new Mapping();

mapa.loadMapping("C:/Java/jdev1012/jdev/mywork/Libreria/Libreria/public_html/PIDs.xml");
File documentoXML=new File(URI);
FileWriter file_writer_documentoXML= new FileWriter(documentoXML);
//Creamos el elemento encargado de transformar de objeto a XML
Marshaller m= new Marshaller(file_writer_documentoXML);
```

```
m.setMapping(mapa);
m.marshal(objeto_java);
```

El mapa a cargar será el que contenga al objeto java a transformar.

### 8.2.3.- Transformación XML-Java.

Cada documento XML contendrá un único objeto java de los implementados, lo cual se corresponde con el principio del XML de que cada documento tenga un único elemento raíz. Además, para la obtención del objeto se conocerá a priori la clase del objeto contenido en el XML. Dicha clase la conocemos en este proyecto porque las interfaces para el intercambio de información están completamente definidas, de modo que se sabe en cada momento el objeto contenido en el documento XML a tratar. Así dependiendo del procedimiento en el que estemos se determinará el objeto raíz contenido en el XML, y se podrá usar el código siguiente:

```
//Creamos el mapping y le cargamos el archivo del mapa
Mapping mapa=new Mapping();
mapa.loadMapping("C:/Java/jdev1012/jdev/mywork/Libreria/Libreria/public_html/PIDs.xml");
//Documento de entrada que contiene al objeto
File documentoXML= new File(URI);
//Lector del documento XML
FileReader in = new FileReader(documentoXML);
//Se encarga de la transformacion xml a objeto
Unmarshaller un= new Unmarshaller(Clase_del_Objeto.class);
un.setMapping( mapa );
//Transformación a objeto
Clase_del_Objeto nuevoObjeto = (Clase_del_Objeto) un.unmarshal(in);
```

Clase_del_Objeto debe ser reemplazado por la clase del objeto correspondiente.

### 8.2.4.- Transformación de los tipos básicos del CEN.

Como ya se ha indicado anteriormente, cada tipo del paquete basicTypes posee un método que le permite rellenarse a sí mismo a partir de un documento XML que lo represente. Esto se debe a que dentro de un objeto concreto es evidente que sabemos de qué clase es el objeto, y debido a que todos los objetos del paquete heredan de DV, se puede crear un identificador DV que se implementará internamente como el objeto que le corresponda según el documento XML. Para el proceso de transformación del objeto a XML se ha creado una clase Conversor, que está incluida en el mismo paquete. En esta clase existen métodos que se encargan de convertir cualquier objeto del paquete basicTypes a XML y viceversa, sin necesidad de saber qué tipo es concretamente.
Los métodos implementados son:

- **public FileWriter toXML(Mapping mapa, Object objeto, String documentoXML):** Transforma cualquier objeto en un archivo XML. Se le debe pasar el mapa que contiene la especificación de la clase, el objeto a trasformar y la URI del documento XML de salida.
- **public Mapping cargarMapa(String direccion):** Crea un *Mapping* y le carga el archivo correspondiente, del cual se le pasa la dirección.
- **public StringBuffer root(FileReader file):** Establece el elemento raíz del documento XML.
- **public String toPaqueteYObjeto(StringBuffer elemento):** A partir de un *String* con el nombre del objeto, determina el paquete que lo contiene si es de la librería, sino no hace nada. Devolverá el nombre del paquete y el objeto.

- **public Object toObject(String nombreClase):** Crea un objeto a partir de su nombre.

Para ejemplificar el uso de dicha clase se ha creado **Prueba.java** dentro del paquete pruebas, que comprueba el correcto funcionamiento de la conversión Java XML del paquete basicTypes mediante un objeto *conversor*. Los documentos XML que produce la ejecución de éste código se almacenaran en documentos/basicTypes.

## 8.3.- Interfaces del PID de CORBAmed

Una vez implementada la librería y estudiado el modo de realizar la pasarela Java-XML, se ha optado por continuar con las interfaces del PID de modo que se puedan pasar las estructuras de datos (recuperadas de la base de datos del proyecto de partida) a documentos XML y viceversa. De las clases que crean la base de datos y nos permiten manejarla no se ha modificado nada, y se han incluido en el paquete **Local.PID.DataBase.**

### 8.3.1.- DataTypes de PID

Dentro de la documentación del PID se definen una serie de estructuras de datos que se usan en sus interfaces. Dichas estructuras fueron inicialmente implementadas en el proyecto de partida en el paquete *PID.ServiceClasses*, pero para el presente proyecto se les ha dotado de estructura de bean y han sido incluidas en el paquete **PID.DataTypes**. Se han implementado además aquellas estructuras que faltaban y se han modificado las que no cumplían la norma.

### 8.3.2.- Exception de PID

No se ha realizado ningún cambio en su implementación inicial, sólo han sido agrupadas en un paquete propio denominado **PID.Exception**.

### 8.3.3.- Revisión de las interfaces IdentifyPerson y ProfileAccess.

Ambas interfaces son implementadas en dos clases: **IdentifyPersonImpl** y **ProfileAccessImpl**, que se han revisado de modo que puedan usar las clases bean que representan los tipos de datos que necesiten. Se han incluido en el paquete **Local.PID.ServiceClasses**, junto con las interfaces que las definen.
Respecto a las clases que las sustentan, se han tenido que revisar completamente para que la creación de los objetos fuera correcta. Se presentan a continuación los cambios principales de cada clase:

#### 8.3.3.1.- Clase DB_CAG_GPICS

Se han modificado todos los métodos para la correcta creación de los objetos de la librería usando los métodos *get y set* para los atributos privados. Se han añadido además una serie de métodos que permiten acceder a más información de la base de datos de modo que se pueda ampliar el número de trait recuperables. Entre ellos se encuentran:

- **public CS get_administrativeGenderCode(int PersonId):** Crea un **CS** con el género de la persona asociada al identificador.

- **public CV get_CVCode(int PersonId, String nombreCV):** Crea un **CV** con un dato especificado por *nombreCV* de la persona con identificador *PersonId*. Los datos que se pueden obtener con este método pertenecen a la tabla Patient y son: *disabilityCode, maritalStatusCode, livingArrangmentCode, religeousAffiliationCode, riskCode, employmentStatusCode y ethnicGroupCode*. Todos del tipo **CV**.
- **public TS get_birthTime(int candidate):** Método que recibe un *PersonId* y devuelve la fecha de nacimiento de la persona.
- **public TS get_deceasedTime(int candidate):** Método que recibe un *PersonId* y devuelve la fecha de fallecimiento de la persona. En caso de no poseer fecha de fallecimiento, se devuelve null.
- **public BL get_deceasedInd(int candidate):** Método que recibe un *PersonId* y devuelve un booleano indicando si la persona ha fallecido.
- **public Int get_birthOrderNumber(int candidate):** Método que recibe un *PersonId* y devuelve el orden de nacimiento de la persona, para casos de parto múltiple.
- **public CV get_codeJobCodes(int PersonId, String name):** Método que recibe un *PersonId* y un nombre de dato relacionado con la condición laboral y devuelve dicho dato de tipo **CV**. Los datos relacionados con la condición laboral se obtienen de la tabla HealthcareProfessional y pueden ser: *code, jobCode, jobTitleCode.*

Así pues, uniendo estos métodos a los que ya poseía la clase, se pueden recuperar los siguientes datos:

- *addr*: conjunto de direcciones de la persona o paciente.
- *administrativeGenderCode*: género de la persona.
- *birthTime*: fecha de nacimiento de la persona.
- *birthOrderNumber*: orden en el nacimiento del paciente.
- *code*: especialidad del profesional sanitario
- *deceasedInd*: indica si el paciente ha fallecido.
- *deceasedTime*: indica la fecha de fallecimiento del paciente.
- *disabilityCode*: discapacidad asociada a un paciente.
- *employmentStatusCode*: situación laboral del paciente.
- *ethnicGroupCode*: grupo étnico al que pertenece el paciente.
- *healthcareProfessionalRole*: rol seguido por la persona como profesional sanitario.
- *id*: identificadores que se poseen de la persona o paciente.
- *jobCode*: posición en la carrera laboral del profesional sanitario
- *jobTitleCode*: título laboral del profesional sanitario.
- *languageCommunication*: conjunto de idiomas conocidos por la persona.
- *livingArrangmentCode*: calidad del alojamiento del paciente.
- *maritalStatusCode*: estado civil del paciente.
- *name*: conjunto de nombres por los que se conoce a la persona.
- *nationalityCode*: conjunto de nacionalidades que tiene el paciente.
- *patientExtendedInformation*: información extendida del paciente.
- *patientStandardInformation*: información estándar del paciente.
- *person*: información de la persona.
- *religeousAffiliationCode*: religion del paciente.
- *riskCode*: riesgos asociados al paciente.
- *telecom*: conjunto de direcciones de telecomunicación asociadas con la persona.

#### 8.3.3.2.- Clase DB_Person

Esta clase es concretamente la que da apoyo a las clases **IdentifyPersonImpl y ProfileAccessImpl**. Sirve para analizar los datos recibidos por dichas clases, realizando la lógica necesaria para determinar qué datos son los que se deben recuperar de la base de datos y accede a la clase **DB_CAG_GPIC** para obtener los objetos que se deben devolver.
La revisión de esta clase ha supuesto la reestructuración de gran parte de su código debido a que el proyecto de partida no se contemplaba la devolución de los objetos de la librería.
Los principales cambios realizados son:

- Se ha creado un conjunto de métodos que analizan los traits recibidos en el elemento *profile_selector* (objeto de la clase **TraitSelectorSeq**) del método **find_candidates**(...) de la interfaz **IdentifyPerson**. Esto se debe a que para buscar los candidatos que más se ajustan al perfil se usa únicamente la información más significativa de cada uno de los traits recibidos. Así por ejemplo, si se recibe un **EntityName**, se decompondrá en cada uno de sus elementos **EntityNamePart**, y de estos se tomará sólo el atributo *caracter_string*, de modo que los campos que finalmente se utilizan en la búsqueda son un conjunto de *String*. Además para indicar la confianza del perfil se calculará sobre el número de traits resultantes de la descomposición.
  Los métodos que se han creado son:
  - **public ArrayList TransformaTrait(Trait trait) throws UnknownTraits**: Método que recibe un Trait y lo transforma a sus datos útiles para realizar la búsqueda del identificador que más requisitos cumpla. Lanza una excepción **UnknownTraits** si recibe un trait que no se admite para hacer la búsqueda porque no se conoce. Devuelve un **ArrayList** con los traits equivalentes preparados para realizar la búsqueda en la base de datos.
  - **public ArrayList TransformaTelecom(Trait trait):** Toma trait con name *"Telecom"* y value de tipo **GPICS.CAG_GPICS.Telecom** y devuelve **ArrayList** de trait con name *"URL"* y value de tipo *String*.
  - **public ArrayList Transformatelecom(Trait trait**): Toma trait con name *"telecom"* y value de tipo **basicTypes.List** y devuelve **ArrayList** de trait con name *"URL"* y value de tipo *String*.
  - **public ArrayList TransformaURL(Trait trait):** Toma trait con name *"URL"* y value de tipo **basicTypes.URL** y devuelve **ArrayList** de trait con name *"URL"* y value de tipo *String*.
  - **public ArrayList TransformaEntityNamePart(Trait trait):** Toma trait con name *"EntityNamePart"* y value de tipo **GPICS.CAG_GPICS.EntityNamePart** y devuelve **ArrayList** de trait con name *"entityNamePart"* y value de tipo *String*.
  - **public ArrayList TransformaEntityName(Trait trait):** Toma trait con name *"EntityName"* y value de tipo **GPICS.CAG_GPICS.EntityName** y devuelve **ArrayList** de trait con name *"entityNamePart"* y value de tipo *String*.
  - **public ArrayList TransformaName(Trait trait):** Toma trait con name *"name"* y value de tipo **basicTypes.List** y devuelve **ArrayList** de trait con name *"entityNamePart"* y value de tipo *String*.
  - **public ArrayList TransformaST(Trait trait):** Toma trait con name variable y value de tipo **basicTypes.ST** y devuelve **ArrayList** de trait con mismo name y value de tipo *String*.
  - **public ArrayList TransformaPostalAddressPart(Trait trait):** Toma un trait con name *"PostalAddressPart"* y value de tipo **GPICS.CAG_GPICS.PostalAddressPart** y devuelve **ArrayList** de trait con name *"addressLine"* y value de tipo *String*.

- **public ArrayList TransformaPostalAddress(Trait trait):** Toma trait con name *"PostalAddress"* y value de tipo **GPICS.CAG_GPICS.PostalAddress** y devuelve **ArrayList** de trait con name *"addressLine"* y value de tipo *String*.
- **public ArrayList TransformaAddr(Trait trait):** Toma trait con name *"addr"* y value de tipo **basicTypes.List** y devuelve **ArrayList** de trait con name *"addressLine"* y value de tipo *String*.
- **public ArrayList TransformaCV(Trait trait):** Toma trait con name variable y value de tipo **basicTypes.CV** y devuelve **ArrayList** de trait con mismo name y value de tipo String.
- **public ArrayList TransformaListCV(Trait trait):** Toma trait con name variable y value de tipo **List<CV>** y devuelve **ArrayList** de trait con mismo name y value de tipo String.
- **public ArrayList TransformaId(Trait trait):** Toma trait con name *"id"* y value de tipo **List** y devuelve **ArrayList** de trait con name *"II"* y value de tipo *String*.
- **public ArrayList TransformaII(Trait trait):** Toma trait con name *"II"* y value de tipo **basicTypes.II** y devuelve **ArrayList** de trait con name de la autoridad certificadora (que se encuentra en el codeValue de assigningAuthorityName) y value de tipo *String*.
- **public ArrayList TransformaCode(Trait trait):** Es para especificar la categoría profesional. Toma trait con name *"code"* y value de tipo **GPICS.HealthcareProfessionalRole** y devuelve **ArrayList** de trait con name *"code"* y value de tipo *String*.
- **public ArrayList TransformaTS(Trait trait):** Toma trait con name variable y value de tipo **basicTypes.TS** y devuelve **ArrayList** de trait con mismo name y value de tipo *TimeStamp*.
- **public ArrayList TransformaCS(Trait trait):** Toma trait con name variable y value de tipo **basitTypes.CS** y devuelve **ArrayList** de trait con mismo name y value de tipo *String*.
- **public ArrayList TransformaLanguageCommunication(Trait trait):** Toma trait con name *"LanguageCommunication"* y value de tipo **GPICS.LanguageCommunication** y devuelve **ArrayList** de trait con name *"languageCode"* y value de tipo *String*.
- **public ArrayList TransformalanguageCommunication(Trait trait):** Toma trait con name *"languageCommunication"* y value de tipo **basicTypes.List** y devuelve **ArrayList** de trait con name *"languageCode"* y value de tipo *String*.

  Los métodos *Tranforma__* son los que dan apoyo al primer método, **TransformaTrait** . Toda esta descomposición se hace para continuar usando la lógica de localización de identificadores de candidatos implementada en el proyecto de partida, que queda reflejada en los métodos:
- **public Vector CandidatePersonId(TraitSelector profile_selector[], float confidence_threshold) throws InvalidWeight, UnknownTraits**: Método que recibe una secuencia de Traits, y devuelve un **Vector** con los *PersonId* de los candidatos seleccionados, ordenados en función de la probabilidad de pertenecer a la persona buscada.
- **private Vector TraitPersonId(String TraitName, Object TraitValue) throws UnknownTraits:** Método que recibe un **Trait**, y devuelve un **Vector** con los *PersonId* de los candidatos seleccionados por el servicio.
  Respecto a como se establece la prioridad de los candidatos no se ha modificado prácticamente.

➢ Se ha sobrescrito el método *FindDuplicateTraits*, de modo que existan 2:

  - **public void FindDuplicateTraits (String requested_traits[]) throws DuplicateTraits**: Método para localizar atributos duplicados. Especifico para *arrays de String*. Se compara únicamente el nombre del trait. (éste era el método ya existente)
  - **public void FindDuplicateTraits (TraitSelector requested_traits[]) throws DuplicateTraits**: Método para localizar atributos duplicados. Especifico para *array de TraitSelector*. Se compara tanto el nombre como el valor del atributo para determinar que sean iguales.
- En la creación de los objetos a devolver dentro de los trait solicitados, se ha cambiado el método *FindTrait* de modo que en el presente proyecto se accede a los métodos de la clase **DB_CAG_GPIC** y se devuelve el objeto correspondiente con todos los campos que sea posible rellenos. El método queda definido de la siguiente manera:
  - **public Trait FindTrait(Object PersonId, String TraitName, DB_CAG_GPIC newObject)**: Método que recibe un **InstanceIdentifier** y nombre de **Trait**, y devuelve el valor de **Trait**. El parámetro *newObject* se usará para acceder a los métodos de su clase.
- Para determinar los traits que se encuentran almacenados en la base de datos para un determinado identificador (método *get_traits_known* de la clase **ProfileAccess**) se ha reutilizado los métodos **Find<Nombre del trait>**. Creados en el proyecto de partida, son más rápidos que la creación completa del objeto y como no se pide la devolución del objeto, permiten una disminución del tiempo de ejecución. Por otro lado para determinar si existe información de una persona como paciente o como profesional sanitario se han creado los métodos:
  - **public boolean KnownPatient(int candidate**): Método que recibe un *PersonId* y devuelve true si pertenece a la base de datos, en la tabla Patients.
  - **public boolean KnownHealthcareProfessionalRole(int candidate):** Método que recibe un *PersonId* y devuelve true si pertenece a la base de datos en la tabla HealthcareProfessional.

  Para personas ya existía el método KnownPerson.

Por último, se muestra en el listado siguiente los traits que se pueden introducir como selectores a la hora de determinar un candidato (es decir los trait contenidos en el **profile_selector** del método *find_candidates*) Se especifican con nombre:valor

- *entityNamePart*: **ST** con un único nombre.
- *addressLine*: **ST** con un único componente de la dirección
- *URL*: **URL** con una única dirección de telecomunicaciones.
- *name*: **List** de EntityName de una persona o paciente.
- *EntityName*: **EntityName**
- *EntityNamePart*: **EntityNamePart**
- *addr*: **List** de PostalAddress de una persona o paciente.
- *PostalAddress*: **PostalAddress**.
- *PostalAddressPart*: **PostalAddressPart.**
- *telecom*: **List** de Telecom de una person o paciente.
- *Telecom*: **Telecom**
- *nationalityCode*: **List** de nacionalidades que posee un paciente.
- *languageCode*: **CV** con el código de idioma hablado por una persona o paciente.
- *disabilityCode*: **CV**
- *maritalStatusCode*: **CV**
- *livingArrangmentCode*: **CV**
- *religeousAffiliationCode*: **CV**
- *riskCode*: **CV**
- *employmentStatusCode*: **CV**
- *ethnicGroupCode*: **CV**

- *id*: **List** de los II de una person o paciente
- *II*: **II**
- *code*: **CV** con el código del profesional según su HealthcarePersonRole
- *birthTime*: **TS**
- *deceasedTime*: **TS**
- *deceasedInd*: **BL**
- *birthOrderNumber*: **Int**
- *administrativeGenderCode*: **CS**
- *LanguageCommunication*: **LanguageCommunication**
- *languageCommunication*: **List** de LanguageCommunication de una persona o paciente.

## 8.3.4.- Paquete PID.Implementation

Se han implementado dentro de éste paquete las tres clases principales que realizan de manera automática las conversiones java xml y viceversa necesarias para la ejecución del código.

### 8.3.4.1.- Clase Canonizador

Se crea esta clase con el fin de codificar los documentos XML para que a priori no contengan caracteres no válidos como pudieran ser ñ o vocales con tilde. Los datos recuperados de la base de datos están correctamente escritos en castellano, pero debido a que ni tildes ni ñ se consideran caracteres válidos, se ha optado por una codificación de los mismos. Esta canonización consiste en que cada vez que aparece uno de los caracteres no válidos se sustituirá por # seguido del carácter más equivalente, es decir: á por #a, é por #e, í por #i, ó por #o, ú por #u, ñ por #n, Á por #A, É por #É, Í por #I, Ó por #O, Ú por #Ú, Ñ por #N; además, para el caso de que en documento original existiera un símbolo #, este será duplicado, de modo que # se sustituye por ##. Este proceso se debe realizar al contrario para aquellos documentos que son recibidos por los métodos.
Para ello esta clase se compone de un constructor vacío:

- **public Canonizador()**
  y los métodos:
- **public void canoniza(File in_file, File out_file) throws IOException:** Método encargado de reescribir los mensajes codificándolos de manera que no contengan tildes ni ñ. El archivo in_file contiene el XML con caracteres no válidos, y out_file contiene el XML sin caracteres no válidos.
- **public void descanoniza(File in_file, File out_file) throws IOException**: Método encargado de reescribir los mensajes con tildes y ñ. El archivo in_file contiene el XML sin caracteres no válidos, y out_file contiene el XML con caracteres no válidos.

### 8.3.4.2.- Clase IdentifyPersonXML

Se encargará de envolver a **IdentifyPersonImpl** de modo que las entradas y las salidas sean archivos. Para ello se ha construido con un atributo privado:

```
private IdentifyPersonImpl IdPersonImpl=null;
//Implementación de la interfaz IdentifyPerson.
```

Este atributo es iniciado en el constructor:

```
public IdentifyPersonXML(DB_Manager DBM)
{
```

```
                IdPersonImpl = new IdentifyPersonImpl(DBM);
        }
```

Juntos constituyen la base para poder llamar al proceso interno de recuperación de datos.
El método que se implementa en esta clase es:

- **public void find_candidates( File profile_selector, IdState[] states_of_interest, float confidence_threshold, int secuence_max, int iterator_max, File traits_requested, File returned_secuence, File returned_iterator) throws UnknownTraits, InvalidWeight, DuplicateTraits:** Implementación del método find_candidates de modo que los parámetros de entrada y salida sean documentos XML.

Dicho método recibe 2 archivos XML codificados: *profile_selector y traits_requested* que se encontrarán en la carpeta documentos/entradas. Los decodifica creando 2 nuevos documentos localizados en la carpeta documentos/intermedios y los transforma a objetos java. Dicha transformación se puede llevar a cabo por que se conoce el objeto que contiene cada archivo, es decir se sabe a priori que el **File profile_selector** contiene un **TraitSelectorSeq** y que **traits_requested** contiene un **SpecifiedTraits** (en cualquier otro caso se producirá una excepción). Además se crean 2 objetos vacíos: un **CandidateSeq** llamado *objeto_returned_secuence* y un **vector** denominado *objeto_returned_iterator*. Estos 4 objetos junto con *states_of_interest*, *confidence_threshold*, *secuence_max*, e *iterator_max* (recibidos como parámetros) se le pasan a la llamada del método **find_candidates** del objeto **IdPersonImpl**, creado en el constructor. La ejecución de este find_candidates interno obtendrá como resultado que se rellenen los objetos *objeto_returned_secuence* y *objeto_returned_iterator* con los datos correspondientes. Así, transformamos los objetos a documentos XML (contenidos en la carpeta documentos/intermedios) y posteriormente los codificamos eliminando los caracteres no válidos obteniendo los archivos de salida que se encontrarán en la carpeta documentos/salidas.
La transformación java-xml se hace con los pasos que se expusieron en el apartado 8.2.2 y 8.2.3 del presente proyecto.
Mostramos a continuación en la figura 8.2. un esquema simple que trata de reflejar el el proceso anteriormente descrito.

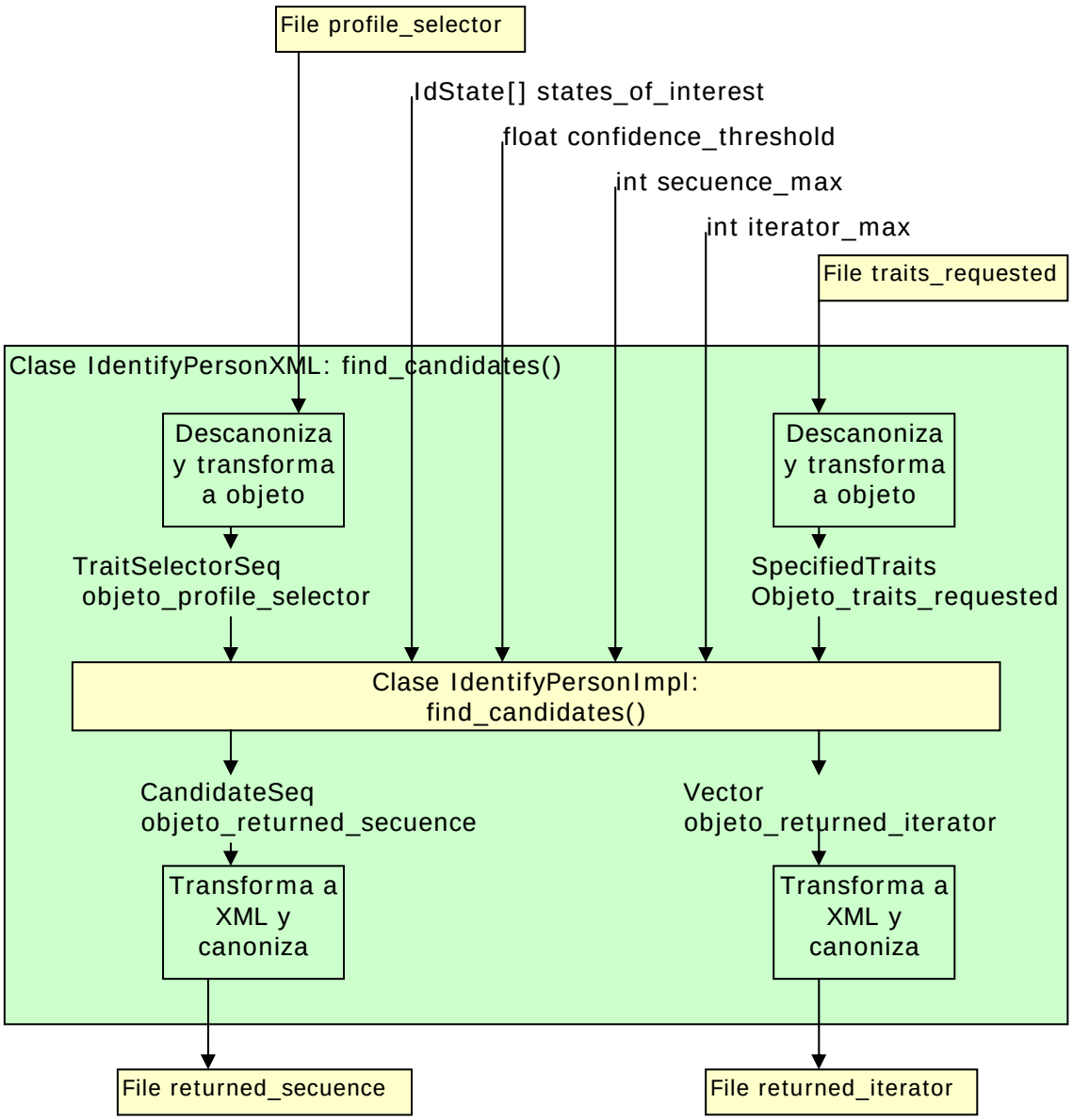


Figura 8.2: Diagrama del método find_candidates(...) de la clase IdentifyPersonXML

### 8.3.4.3.- Clase ProfileAccessXML

De forma equivalente a la clase anterior, **ProfileAccessXML** se encarga de envolver a **ProfileAccessImpl** para realizar la transformación java-xml y viceversa que sean necesarias.
Se crea a su vez con un atributo privado:

```
private ProfileAccessImpl ProAccImpl=null;
//Implementación de la interfaz ProfileAccess.
```

Este atributo es iniciado en el constructor:

```
public ProfileAccessXML(DB_Manager DBM)
{
        ProAccImpl=new ProfileAccessImpl(DBM);
}
```

Juntos constituyen la base para poder llamar al proceso interno de recuperación de datos proporcionados por la interfaz *ProfileAccess*. Los métodos que se han creado son:

- **public File get_traits_known(String PersonId) throws InvalidId**: Método que permite recuperar en un archivo los trait que se pueden solicitar de un determinado *PersonId*.
- **public File get_profile(String PersonId, File traits_requested) throws InvalidId, UnknownTraits**: Método que permite obtener un conjunto de trait especificados en el archivo **trait_requested** para un *PersonId*, devolviéndolos en un archivo.
- **public File get_profile_list(String ids[], File traits_requested) throws InvalidIds, DuplicateIds, UnknownTraits, DuplicateTraits**: Método que permite obtener un conjunto de trait especificados en el archivo **trait_requested** para todos los *PersonId* de la lista ids suministrada, devolviéndolos en un archivo.

Del mismo modo que antes, los archivos de entrada (carpeta documentos/entradas) son decodificados (carpeta documentos/intermedios) y transformados a objetos que se pasan como parámetros el método correspondiente dentro de los implementados en ProfileAccessImpl; a su vez los objetos devueltos por dichos métodos son transformados a documentos XML (carpeta documentos/intermedios) y codificados para eliminar los caracteres no válidos produciéndose los archivos de salida (carpeta documentos/salida) que se devuelven por los métodos antes listados.
En las tres figuras siguientes, 8.3, 8.4 y 8.5, mostramos de forma esquemática como actúan cada uno de los métodos de la clase ProfileAccessXML que se han implementado.

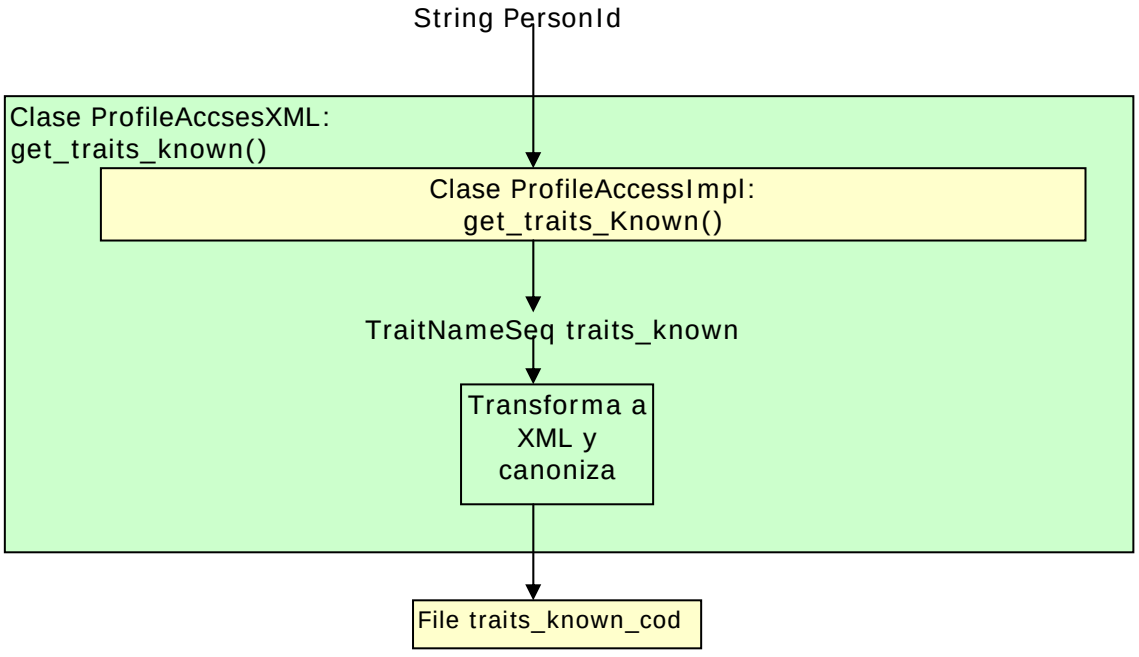


Figura 8.3: Diagrama del método get_traits_known(...) de la clase ProfileAccessXML

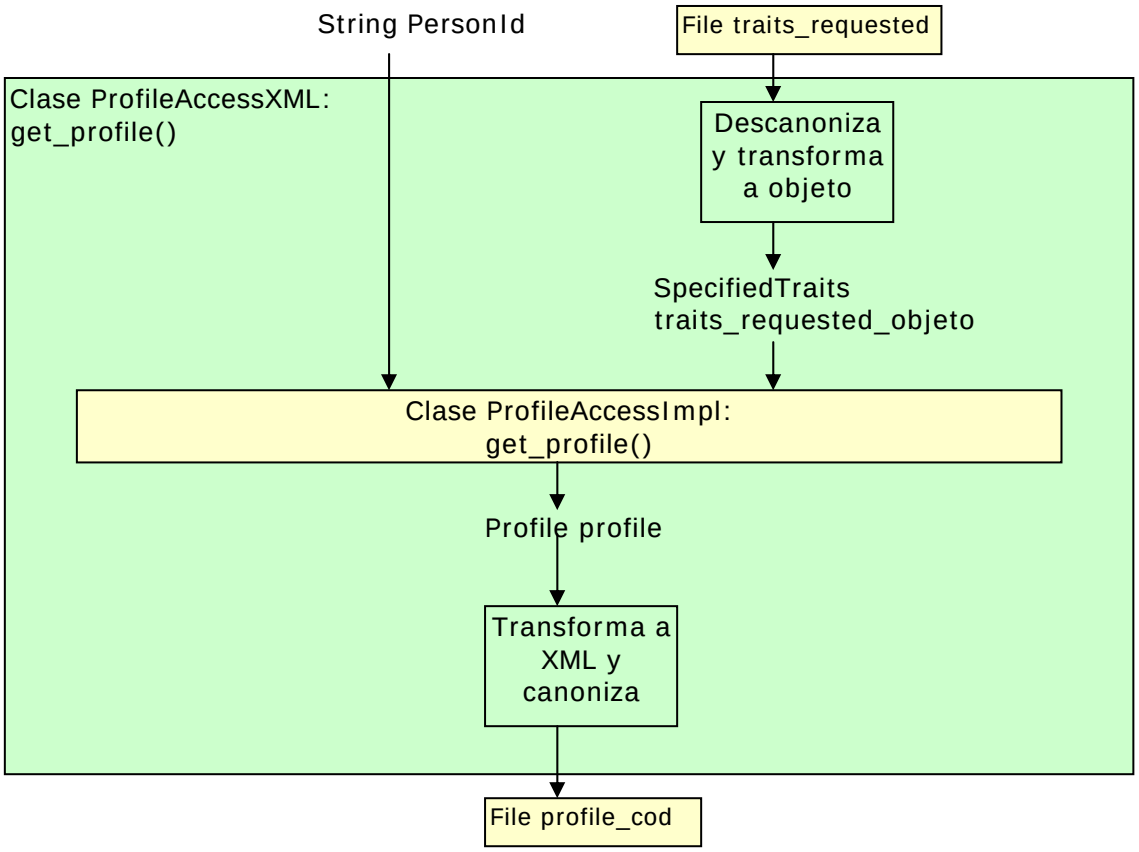


Figura 8.4: Diagrama del método get_profile(...) de la clase ProfileAccessXML

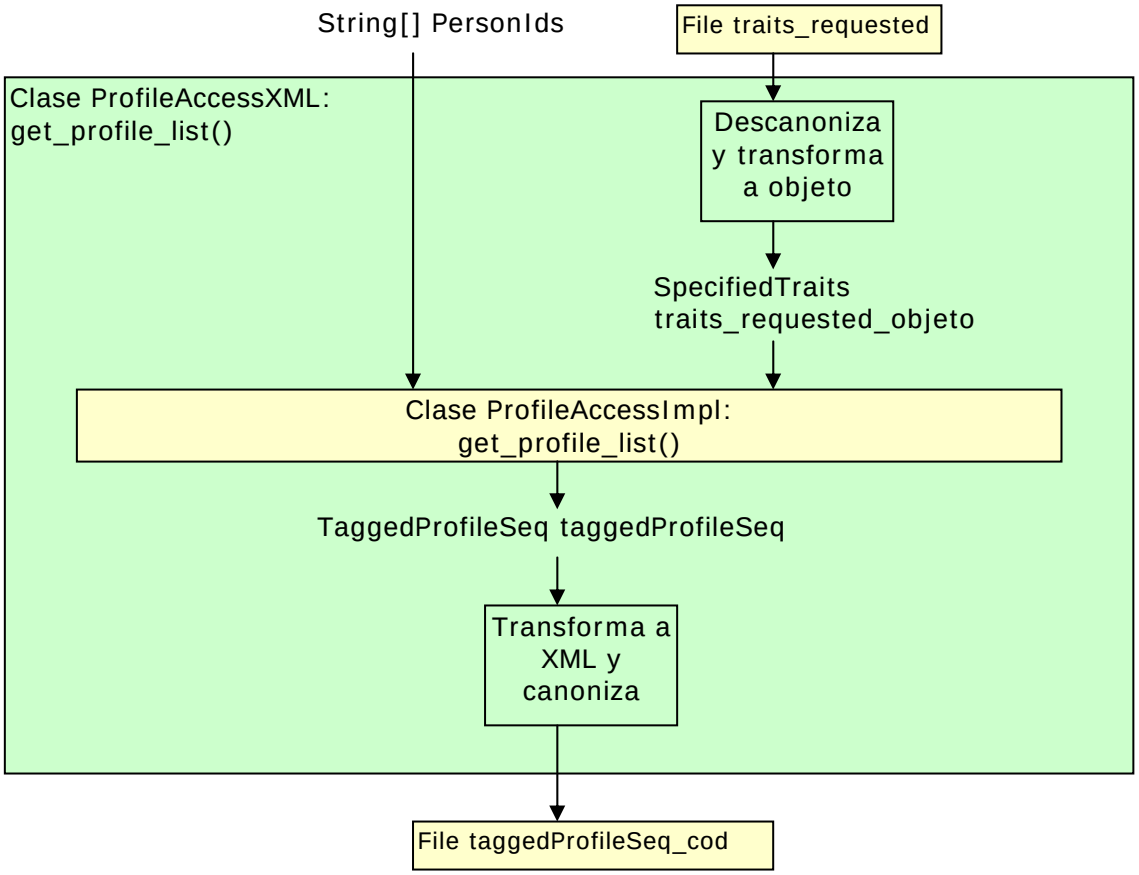


Figura 8.5: Diagrama del método get_profile_list(...) de la clase ProfileAccessXML

## 8.4.-Resumen de paquetes

Debido a que el proyecto cuenta con dos partes diferenciadas, se ha optado por exponerlas en dos carpetas diferentes. En la primera, denominada *"Librería"* se encuentran todas las clases inherentes a la librería, y en la segunda, *"Aplicacion"*, se encuentran las clases del servidor demográfico.
A continuación se muestra parte de la documentación de la aplicación generada mediante la herramienta JavaDoc, en las que se especifican todos los paquetes que han sido necesarios implemetar con sus correspondientes clases.

### 8.4.1.- Librería

**Resumen de paquetes**

| Packages | |
|---|---|
| basicTypes | Este paquete contiene las clases que encapsulan a los tipos básicos (primitivos y compuestos) del estándar Health informatics-Data types del CEN. |
| GPICS | Este paquete contiene las clases que encapsulan a los tipos del estándar prEN 14822:2003 del CEN (tipos GPIC). |
| GPICS.CAG_GPICS | Este paquete contiene las clases que encapsulan a los tipos del estándar prEN 14822:2003 del CEN (tipos CAG). |
| GPICS.CommonSubComponents | Este paquete contiene las clases que encapsulan a los tipos del estándar prEN 14822:2003 del CEN (tipos CommonSubComponent). |

Figura 8.6: Paquetes de la librería.

**Paquete basicTypes**

| Class Summary | |
|---|---|
| BL | Representa el tipo Boolean del estándar Health informatics-Data types del CEN. |
| Byte | Representa el tipo Byte del estándar Health informatics-Data types del CEN. |
| CD | Representa el tipo Concept Descriptor del estándar Health informatics-Data types del CEN. |
| CE | Representa el tipo Coded with Equivalents del estándar Health informatics-Data types del CEN. |
| Char | Representa el tipo Char del estándar Health informatics-Data types del CEN. |
| CodedOrText | Representa el tipo CodedOrText del estándar Health informatics-Data types del CEN. |

| | |
|---|---|
| **CodedText** | Representa el tipo CodedText del estándar Health informatics-Data types del CEN. |
| **Conversor** | Clase que se encarga de convertir cualquier objeto del paquete basicTypes a XML y viceversa sin necesidad de saber que tipo es. |
| **CR** | Representa el tipo Concept Role del estándar Health informatics-Data types del CEN. |
| **CS** | Representa el tipo Coded Simple Value del estándar Health informatics-Data types del CEN. |
| **CV** | Representa el tipo Coded Value del estándar Health informatics-Data types del CEN. |
| **CVQ** | Representa el tipo Coded Value with Qualifier del estándar Health informatics-Data types del CEN. |
| **Date** | Representa el tipo Date del estándar Health informatics-Data types del CEN. |
| **Double** | Representa el tipo Double del estándar Health informatics-Data types del CEN. |
| **DV** | Representa el tipo abstracto Data Value del estándar Health informatics-Data types del CEN. |
| **ED** | Representa el tipo Encapsulated Data del estándar Health informatics-Data types del CEN. |
| **EIVL** | Representa el tipo Periodic Interval of Time del estándar Health informatics-Data types del CEN. |
| **Float** | Representa el tipo Float del estándar Health informatics-Data types del CEN. |
| **II** | Representa el tipo Instance Identifier del estándar Health informatics-Data types del CEN. |
| **Int** | Representa el tipo Int del estándar Health informatics-Data types del CEN. |
| **Integer** | Representa el tipo Integer del estándar Health informatics-Data types del CEN. |
| **IVL_PQ** | Representa el tipo QuantityRange del estándar Health informatics-Data types del CEN. |
| **IVL_T** | Representa el tipo Interval del estándar Health informatics-Data types del CEN. |
| **IVL_TS** | Representa al tipo Interval Of Time del estándar Health informatics-Data types del CEN. |
| **List** | Representa el tipo List Se ha implementado para reflejar un tipo común a todas las listas de la libreria |
| **Longint** | Representa el tipo Longint del estándar Health informatics-Data types del CEN. |
| **Numeric** | Representa el tipo Numeric del estándar Health informatics-Data types del CEN. |
| **OID** | Representa el tipo ISO Object Identifier del estándar Health informatics-Data types del CEN. |
| **ORD** | Representa el tipo Ordinal del estándar Health informatics-Data |

| | |
|---|---|
| | types del CEN. |
| **PIVL** | Representa el tipo Periodic Interval of Time del estándar Health informatics-Data types del CEN. |
| **PQ** | Representa el tipo Physical Quantity del estándar Health informatics-Data types del CEN. |
| **Quantity** | Representa el tipo abstracto Quantity del estándar Health informatics-Data types del CEN. |
| **Real** | Representa el tipo Real del estándar Health informatics-Data types del CEN. |
| **RTO** | Representa el tipo Ratio del estándar Health informatics-Data types del CEN. |
| **Shortint** | Representa el tipo Longint del estándar Health informatics-Data types del CEN. |
| **ST** | Representa el tipo String del estándar Health informatics-Data types del CEN. |
| **Text** | Representa el tipo Text del estándar Health informatics-Data types del CEN. |
| **TS** | Representa el tipo Time Point del estándar Health informatics-Data types del CEN. |
| **URL** | Representa el tipo Universal Resource Locator del estándar Health informatics-Data types del CEN. |

Figura 8.7: Paquete basicTypes

**Paquete GPICS**

| Class Summary | |
|---|---|
| **AnalysableObjectInUse** | Representa el tipo AnalysableObjectInUse del estándar prEN 14822:2003 del CEN. |
| **Authorisation** | Representa el tipo Authorisation del estándar prEN 14822:2003 del CEN. |
| **AuthorisationLink** | Representa el tipo AuthorisationLink del estándar prEN 14822:2003 del CEN. |
| **CareCost** | Representa el tipo CareCost del estándar prEN 14822:2003 del CEN. |
| **CareCostLink** | Representa el tipo CareCostLink del estándar prEN 14822:2003 del CEN. |
| **CareEncounterRecipient** | Representa el tipo CareEncounterRecipient del estándar prEN 14822:2003 del CEN. |
| **CareLocation** | Representa el tipo CareLocation del estándar prEN 14822:2003 del CEN. |
| **CareLocationInUse** | Representa el tipo CareLocationInUse del estándar prEN 14822:2003 del CEN. |
| **CareServiceRecipient** | Representa el tipo CareServiceRecipient del estándar prEN 14822:2003 del CEN. |

| | |
|---|---|
| **ContactPerson** | Representa el tipo ContactPerson del estándar prEN 14822:2003 del CEN. |
| **ContactRole** | Representa el tipo ContactRole del estándar prEN 14822:2003 del CEN. |
| **Device** | Representa el tipo Device del estándar prEN 14822:2003 del CEN. |
| **DeviceInUse** | Representa el tipo DeviceInUse del estándar prEN 14822:2003 del CEN. |
| **DeviceParameter** | Representa el tipo DeviceParameter del estándar prEN 14822:2003 del CEN. |
| **DeviceParameterLink** | Representa el tipo DeviceParameterLink del estándar prEN 14822:2003 del CEN. |
| **DeviceRole** | Representa el tipo DeviceRole del estándar prEN 14822:2003 del CEN. |
| **GeographicLocation** | Representa el tipo GeographicLocation del estándar prEN 14822:2003 del CEN. |
| **GeographicLocationRole** | Representa el tipo GeographicLocationRole del estándar prEN 14822:2003 del CEN. |
| **HealthcareAgent** | Representa el tipo HealthcareAgent del estándar prEN 14822:2003 del CEN. |
| **HealthcareOrganisation** | Representa el tipo HealthcareOrganisation del estándar prEN 14822:2003 del CEN. |
| **HealthcareOrganisationRole** | Representa el tipo HealthcareOrganisationRole del estándar prEN 14822:2003 del CEN. |
| **HealthcareParty** | Representa el tipo abstracto HealthcareParty del estándar prEN 14822:2003 del CEN. |
| **HealthcareProfessional** | Representa el tipo HealthcareProfessional del estándar prEN 14822:2003 del CEN. |
| **HealthcareProfessionalRole** | Representa el tipo HealthcareProfessionalRole del estándar prEN 14822:2003 del CEN. |
| **IdentifiedDevice** | Representa el tipo IdentifiedDevice del estándar prEN 14822:2003 del CEN. |
| **IdentifiedDeviceInUse** | Representa el tipo IdentifiedDeviceInUse del estándar prEN 14822:2003 del CEN. |
| **IdentifiedDeviceRole** | Representa el tipo IdentifiedDeviceRole del estándar prEN 14822:2003 del CEN. |
| **IdentifiedHealthcareAgent** | Representa el tipo abstracto IdentifiedHealthcareAgent del estándar prEN 14822:2003 del CEN. |
| **IdentifiedHealthcareOrganisation** | Representa el tipo IdentifiedHealthcareOrganisation del estándar prEN 14822:2003 del CEN. |

| | |
|---|---|
| **IdentifiedHealthcareParty** | Representa el tipo abstracto IdentifiedHealthcareParty del estándar prEN 14822:2003 del CEN. |
| **IdentifiedHealthcareProfessional** | Representa el tipo IdentifiedHealthcareProfessional del estándar prEN 14822:2003 del CEN. |
| **IdentifiedLivingSubject** | Representa el tipo IdentifiedLivingSubject del estándar prEN 14822:2003 del CEN. |
| **IdentifiedOrganisation** | Representa el tipo IdentifiedOrganisation del estándar prEN 14822:2003 del CEN. |
| **IdentifiedOrganisationParticipation** | Representa el tipo IdentifiedOrganisationParticipation del estándar prEN 14822:2003 del CEN. |
| **IdentifiedOrganisationRole** | Representa el tipo IdentifiedOrganisationRole del estándar prEN 14822:2003 del CEN. |
| **IdentifiedParticipatingOrganisation** | Representa el tipo IdentifiedParticipatingOrganisation del estándar prEN 14822:2003 del CEN. |
| **IdentifiedParticipatingPatient** | Representa el tipo IdentifiedParticipatingPatient del estándar prEN 14822:2003 del CEN. |
| **IdentifiedParticipatingProfessional** | Representa el tipo IdentifiedParticipatingProfessional del estándar prEN 14822:2003 del CEN. |
| **IdentifiedProfessional** | Representa el tipo IdentifiedProfessional del estándar prEN 14822:2003 del CEN. |
| **IdentifiedProfessionalRole** | Representa el tipo IdentifiedProfessionalRole del estándar prEN 14822:2003 del CEN. |
| **ItemTransportation** | Representa el tipo ItemTransportation del estándar prEN 14822:2003 del CEN. |
| **LanguageCommunication** | Representa el tipo LangeageCommunication del estándar prEN 14822:2003 del CEN. |
| **LivingSubjectTransportation** | Representa el tipo LivingSubjectTransportation del estándar prEN 14822:2003 del CEN. |
| **LocationRole** | Representa el tipo LocationRole del estándar prEN 14822:2003 del CEN. |
| **NonLivingEntityTransportation** | Representa el tipo NonLivingEntityTransportation del estándar prEN 14822:2003 del CEN. |
| **Organisation** | Representa el tipo Organisation del estándar prEN 14822:2003 del CEN. |
| **OrganisationHierarchy** | Representa el tipo OrganisationHierarchy del estándar prEN 14822:2003 del CEN. |

| | |
|---|---|
| **OrganisationHierarchyRole** | Representa el tipo OrganisationHierarchyRole del estándar prEN 14822:2003 del CEN. |
| **ParticipatingHealthcareOrganisation** | Representa el tipo ParticipatingHealthcareOrganisation del estándar prEN 14822:2003 del CEN. |
| **ParticipatingHealthcareParty** | Representa el tipo ParticipatingHealthcareParty del estándar prEN 14822:2003 del CEN. |
| **ParticipatingHealthcareProfessional** | Representa el tipo ParticipatingHealthcareProfessional del estándar prEN 14822:2003 del CEN. |
| **ParticipatingLocation** | Representa el tipo ParticipatingLocation del estándar prEN 14822:2003 del CEN. |
| **ParticipatingPatient** | Representa el tipo ParticipatingPatient del estándar prEN 14822:2003 del CEN. |
| **ParticipatingPatientRelatedParty** | Representa el tipo ParticipatingPatientRelatedParty del estándar prEN 14822:2003 del CEN. |
| **ParticipatingRelatedSubjectOfCare** | Representa el tipo ParticipatingRelatedSubjectOfCare del estándar prEN 14822:2003 del CEN. |
| **ParticipatingSubjectOfCare** | Representa el tipo ParticipatingSubjectOfCare del estándar prEN 14822:2003 del CEN. |
| **PatientExtendedInformation** | Representa el tipo PatientExtendedInformation del estándar prEN 14822:2003 del CEN. |
| **PatientStandardInformation** | Representa el tipo PatientStandardInformation del estándar prEN 14822:2003 del CEN. |
| **Person** | Representa el tipo Person del estándar prEN 14822:2003 del CEN. |
| **ReferencedAgentParticipation** | Representa el tipo ReferencedAgentParticipation del estándar prEN 14822:2003 del CEN. |
| **ReferencedDeviceParticipation** | Representa el tipo ReferencedDeviceParticipation del estándar prEN 14822:2003 del CEN. |
| **ReferencedHealthcareAgent** | Representa el tipo ReferencedHealthcareAgent del estándar prEN 14822:2003 del CEN. |
| **ReferencedHealthcareDevice** | Representa el tipo ReferencedHealthcareDevice del estándar prEN 14822:2003 del CEN. |
| **ReferencedHealthcareOrganisation** | Representa el tipo ReferencedHealthcareOrganisation del estándar prEN 14822:2003 del CEN. |

| | |
|---|---|
| **ReferencedHealthcareParty** | Representa el tipo ReferencedHealthcareParty del estándar prEN 14822:2003 del CEN. |
| **ReferencedHealthcareProfessional** | Representa el tipo ReferencedHealthcareProfessional del estándar prEN 14822:2003 del CEN. |
| **ReferencedOrganisationParticipation** | Representa el tipo ReferencedOrganisationParticipation del estándar prEN 14822:2003 del CEN. |
| **ReferencedPartyParticipation** | Representa el tipo ReferencedPartyParticipation del estándar prEN 14822:2003 del CEN. |
| **ReferencedProfessionalParticipation** | Representa el tipo ReferencedProfessionalParticipation del estándar prEN 14822:2003 del CEN. |
| **ReferencedSubjectOfCare** | Representa el tipo ReferencedSubjectOfCare del estándar prEN 14822:2003 del CEN. |
| **ReferencedSubjectParticipation** | Representa el tipo ReferencedSubjectParticipation del estándar prEN 14822:2003 del CEN. |
| **ReferencedSubjectRole** | Representa el tipo ReferencedSubjectRole del estándar prEN 14822:2003 del CEN. |
| **RelatedAgentLink** | Representa el tipo RelatedAgentLink del estándar prEN 14822:2003 del CEN. |
| **RelatedDevice** | Representa el tipo RelatedDevice del estándar prEN 14822:2003 del CEN. |
| **RelatedHealthcareAgent** | Representa el tipo RelatedHealthcareAgent del estándar prEN 14822:2003 del CEN. |
| **RelatedHealthcareOrganisation** | Representa el tipo RelatedHealthcareOrganisation del estándar prEN 14822:2003 del CEN. |
| **RelatedHealthcareProfessional** | Representa el tipo RelatedHealthcareProfessional del estándar prEN 14822:2003 del CEN. |
| **RelatedIdentifiedDevice** | Representa el tipo RelatedIdentifiedDevice del estándar prEN 14822:2003 del CEN. |
| **RelatedLivingSubjectTransportation** | Representa el tipo RelatedLivingSubjectTransportation del estándar prEN 14822:2003 del CEN. |
| **RelatedOrganisation** | Representa el tipo RelatedOrganisation del estándar prEN 14822:2003 del CEN. |
| **RelatedOrganisationLink** | Representa el tipo RelatedOrganisationLink del estándar prEN 14822:2003 del CEN. |
| **RelatedOrganisationRole** | Representa el tipo |

| | |
|---|---|
| | RelatedOrganisationRole del estándar prEN 14822:2003 del CEN. |
| **RelatedParty** | Representa el tipo RelatedParty del estándar prEN 14822:2003 del CEN. |
| **RelatedPartyRole** | Representa el tipo RelatedPartyRole del estándar prEN 14822:2003 del CEN. |
| **RelatedProfessionalLink** | Representa el tipo RelatedProfessionalLink del estándar prEN 14822:2003 del CEN. |
| **RelatedSubjectOfCare** | Representa el tipo RelatedSubjectOfCare del estándar prEN 14822:2003 del CEN. |
| **RelatedSubjectOfCareParticipation** | Representa el tipo RelatedSubjectOfCareParticipation del estándar prEN 14822:2003 del CEN. |
| **RelatedSubjectOfCareRole** | Representa el tipo RelatedSubjectOfCareRole del estándar prEN 14822:2003 del CEN. |
| **RelatedTransportation** | Representa el tipo RelatedTransportation del estándar prEN 14822:2003 del CEN. |
| **ServiceAgreement** | Representa el tipo ServiceAgreement del estándar prEN 14822:2003 del CEN. |
| **ServiceAgreementLink** | Representa el tipo ServiceAgreementLink del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCare** | Representa el tipo abstracto SubjectOfCare del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCareAnimal** | Representa el tipo abstracto SubjectOfCareAnimal del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCareAnimalGroup** | Representa el tipo abstracto SubjectOfCareAnimalGroup del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCareAnimalIdentification** | Representa el tipo abstracto SubjectOfCareAnimalIdentification del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCareIdentification** | Representa el tipo abstracto SubjectOfCareIdentification del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCareParticipation** | Representa el tipo SubjectOfCareParticipation del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCarePerson** | Repredenta el tipo abstracto SubjectOfCarePerson del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCarePersonIdentification** | Representa el tipo abstracto SubjectOfCarePersonIdentification del estándar prEN 14822:2003 del CEN. |

| | |
|---|---|
| **SubjectOfCarePersonRole** | Representa el tipo SubjectOfCarePersonRole del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCareRelatedParty** | Representa el tipo SubjectOfCareRelatedParty del estándar prEN 14822:2003 del CEN. |
| **SubjectOfCareRole** | Representa el tipo SubjectOfCareRole del estándar prEN 14822:2003 del CEN. |
| **SubjectOfInvestigation** | Representa el tipo SubjectOfInvestigation del estándar prEN 14822:2003 del CEN. |
| **SubjectOfInvestigationParticipation** | Representa el tipo SubjectOfInvestigationParticipation del estándar prEN 14822:2003 del CEN. |
| **SubjectOfInvestigationRole** | Representa el tipo SubjectOfInvestigationRole del estándar prEN 14822:2003 del CEN. |
| **SubjectTransportation** | Representa el tipo SubjectTransportation del estándar prEN 14822:2003 del CEN. |

Figura 8.8: Paquete GPICS

**Paquete GPICS.CAG_GPICS**

| Class Summary | |
|---|---|
| **EntityName** | Representa el tipo EntityName del estándar prEN 14822:2003 del CEN. |
| **EntityNamePart** | Representa el tipo EntityNamePart del estándar prEN 14822:2003 del CEN. |
| **PostalAddress** | Representa el tipo PostalAddress del estándar prEN 14822:2003 del CEN. |
| **PostalAddressPart** | Representa el tipo PostalAddressPart del estándar prEN 14822:2003 del CEN. |
| **Telecom** | Representa el tipo Telecom del estándar prEN 14822:2003 del CEN. |

Figura 8.9: Paquete GPICS.CAG_GPICS

**Paquete GPICS.CommonSubComponents**

| Class Summary | |
|---|---|
| **DeviceParticipation** | Representa el tipo abstracto DeviceParticipation del estándar prEN 14822:2003 del CEN. |
| **HealthcareAgentParticipation** | Representa el tipo abstracto HealthcareAgentParticipation del estándar prEN 14822:2003 del CEN. |

| | |
|---|---|
| **HealthcareOrganisationParticipation** | Representa el tipo abstracto HealthcareOrganisationParticipation del estándar prEN 14822:2003 del CEN. |
| **HealthcareProfessionalParticipation** | Representa el tipo abstracto HealthcareProfessionalParticipation del estándar prEN 14822:2003 del CEN. |
| **NonHealthcarePersonParticipation** | Representa el tipo abstracto NonHealthcarePersonParticipation del estándar prEN 14822:2003 del CEN. |

Figura 8.10: Paquete GPICS.CommonSubComponents

## 8.4.2.- Aplicación

### Paquetes de la aplicación

| Packages | |
|---|---|
| **Local.PID.DataBase** | Este paquete contiene las clases necesarias para que el servicio pueda interactuar con la base de datos y realizar búsquedas de atributos e identificadores de persona. |
| **Local.PID.ServiceClasses** | Este paquete contiene las clases necesarias para que el servicio pueda acceder a los métodos implementados de las interfaces IdentityPerson y ProfileAccess de CORBAmed (find_candidates(), get_profile(), get_profile_list() y get_traits_known()), a partir de los cuales se realizan consultas en la base de datos para obtener identificadores de persona o valores de atributos. |
| **PID.DataTypes** | Este paquete contiene las clases que encapsulan a los tipos de datos definidos en el PID en el apartado de Data Types |
| **PID.Exception** | Este paquete contiene las clases que encapsulan a las excepciones definidas en el PID en el apartado de Data Types |
| **PID.Implementation** | Este paquete contiene las clases necesarias para que el servicio pueda realizar sus funciones de pasarela java-XML |
| **PID.Pruebas** | Paquete que contiene dos pequeños programas de prueba que simulan la función del cliente |

Figura 8.11: Paquete Local.PID.DataBase

### Paquete Local.PID.DataBase

| Class Summary | |
|---|---|
| **DB_BasicTypes** | Métodos necesarios para rellenar los campos de los tipos basicos |

| | |
|---|---|
| | (primitivos y compuestos) del estandar Health informatics-Data types del CEN. |
| **DB_CAG_GPIC** | Métodos necesarios para rellenar los campos de los tipos CAG y GPIC del estandar prEN 14822:2003 del CEN. |
| **DB_Factory** | Clase que contiene los métodos necesarios para crear las tablas de la base de datos. |
| **DB_Manager** | Clase que contiene los métodos necesarios para interactuar con la base de datos: conectarse, desconectarse, consultar o modificar tablas. |
| **DB_Person** | Clase que contiene los métodos necesarios para implementar las interfaces IdentifyPerson y ProfileAccess. |

Figura 8.12: Paquete Local.PID.DataBase

**Paquete Local.PID.ServiceClasses**

| Interface Summary | |
|---|---|
| **IdentificationComponent** | Interfaz IdentificationComponent. |
| **IdentifyPerson** | Interfaz IdentifyPerson. |
| **ProfileAccess** | Interfaz ProfileAccess. |

| Class Summary | |
|---|---|
| **IdentifyPersonImpl** | Implementación de la interfaz IdentifyPerson. |
| **ProfileAccessImpl** | Implementación de la interfaz ProfileAccess. |

Figura 8.13: Paquete Local.PID.ServiceClasses

**Paquete PID.DataTypes**

| Interface Summary | |
|---|---|
| **CandidateIterator** | Generated from IDL definition of interface "CandidateIterator" |

| Class Summary | |
|---|---|
| **Candidate** | Generated from IDL definition of struct "Candidate" |
| **CandidateIteratorImpl** | Generated from IDL definition of struct "Candidate" |
| **CandidateSeq** | Clase que representa una estructura que almacena un número indeterminado de Candidates. |
| **IdState** | Generated from IDL definition of enum "IdState" |
| **IdStateSeq** | Generated from IDL definition of struct "IdStateSeq" |
| **MergeStruct** | Generated from IDL definition of struct "MergeStruct" |
| **MergeStructSeq** | Generated from IDL definition of struct "MergeStructSeq" |
| **Profile** | Generated from IDL definition of struct "Profile" |

| | |
|---|---|
| **ProfileSeq** | Generated from IDL definition of struct "ProfileSeq" |
| **ProfileUpdate** | Generated from IDL definition of struct "ProfileUpdate" |
| **ProfileUpdateSeq** | Generated from IDL definition of struct "ProfileUpdateSeq" |
| **SpecifiedTraits** | Clase que representa una secuencia de TraitSpec. |
| **TaggedProfile** | Clase que representa a la estructura TaggedProfile. |
| **TaggedProfileSeq** | Clase que representa una secuencia de TaggedProfile. |
| **Trait** | Generated from IDL definition of struct "Trait" |
| **TraitNameSeq** | Clase que representa una secuencia de nombres de atributos. |
| **TraitSelector** | Generated from IDL definition of struct "TraitSelector" |
| **TraitSelectorSeq** | Clase que representa una secuencia de TraitSelector. |
| **TraitSeq** | Generated from IDL definition of struct "TraitSeq" |
| **TraitSpec** | Generated from IDL definition of struct "TraitSpec" |
| **TraitSpecSeq** | Generated from IDL definition of struct "TraitSpecSeq" Estructura que contiene una secuencia de Traitspec. |

Figura 8.14: Paquete PID.DataTypes

**Paquete PID.Exception**

| Exception Summary | |
|---|---|
| **DuplicateIds** | Excepción DuplicateIds, identificadores de persona repetidos. |
| **DuplicateTraits** | Excepción DuplicateTraits, atributos repetidos. |
| **InvalidId** | Excepción InvalidId, identificador de persona no válido. |
| **InvalidIds** | Excepción InvalidIds, identificadores de persona no válidos. |
| **InvalidWeight** | Excepción InvalidWeight, peso no válido. |
| **UnknownTraits** | Excepción UnknownTraits, atributo no válido. |

Figura 8.15: Paquete PID.Exception

**Paquete PID.Implementation**

| Class Summary | |
|---|---|
| **Canonizador** | Clase que se encarga de codificar las tildes y ñ de un File |
| **IdentifyPersonXML** | Clase que se encarga de transformar los documentos XML recibidos en objetos java para poder llamar a IdentifyPerson. |
| **ProfileAccessXML** | Clase que se encarga de transformar los documentos XML recibidos en objetos java para poder llamar a ProfileAccess. |

Figura 8.16: Paquete PID.Implementation

**Paquete pruebas**

| Class Summary | |
|---|---|
| **Prueba** | Programa de prueba: Comprueba el correcto funcionamiento de la conversion Java XML del paquete basicTypes mediante la clase conversor. |
| **Prueba2** | Programa de prueba: Esta clase como si de un cliente crea los documentos XML de entrada al servicio y lanza las consultas; recibe los documentos XML con los resultados obtenidos. |

Figura 8.17: Paquete PID.pruebas