PRÁCTICA DE ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS: MASHUP

UNIVERSIDADE DA CORUÑA

Máster en Informática

Realizada por:
Jesús Ángel Pérez-Roca Fernández (infjpf02)

ÍNDICE

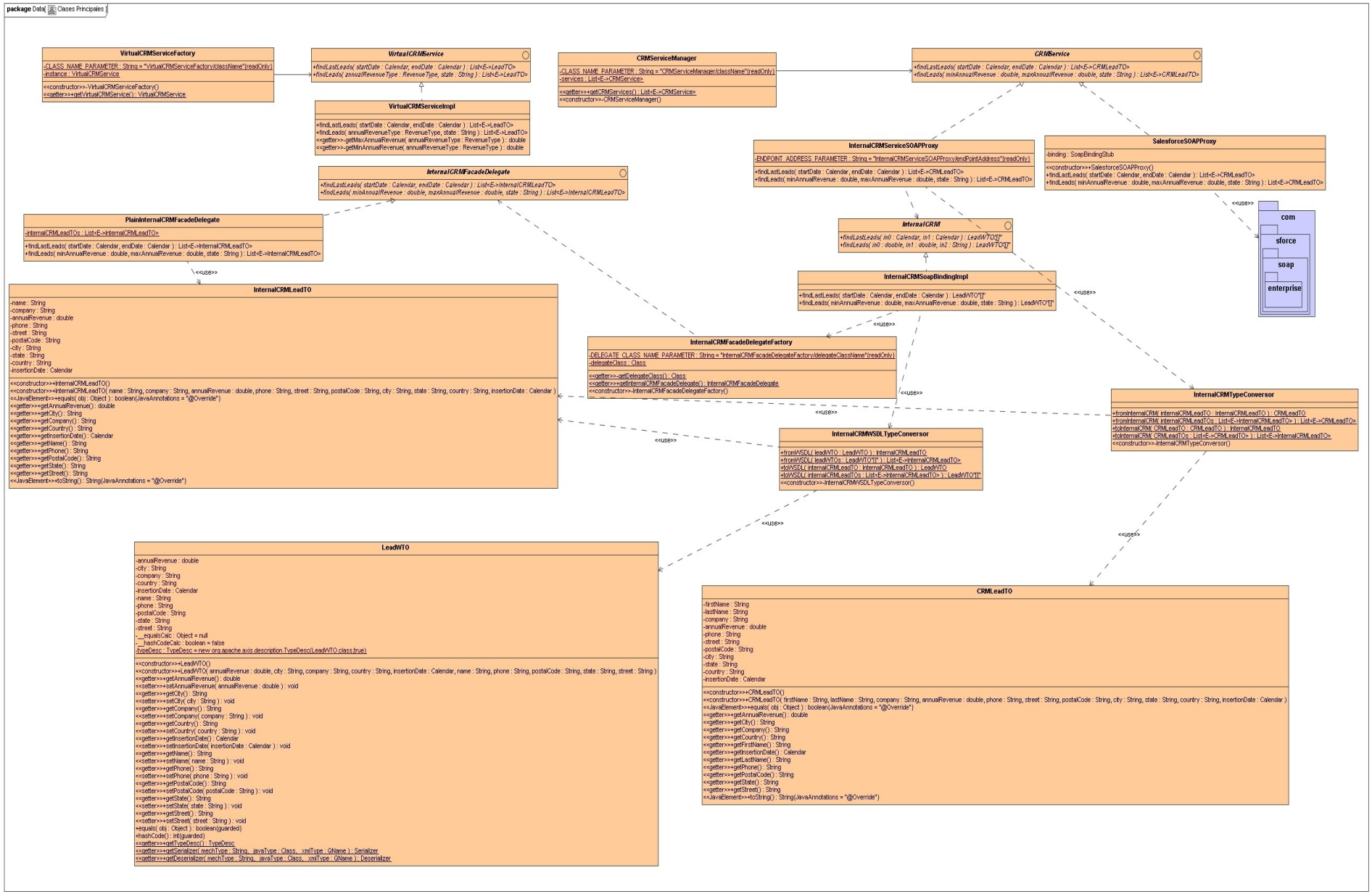
1. Introducción

Además de implementarse las partes obligatorias (Servicio de búsqueda del CRM interno, Servicio de geolocalización de Google Maps y Servicio RSS), también se ha implementado una de las dos partes optativas, concretamente el Servicio de búsqueda de Salesforce.

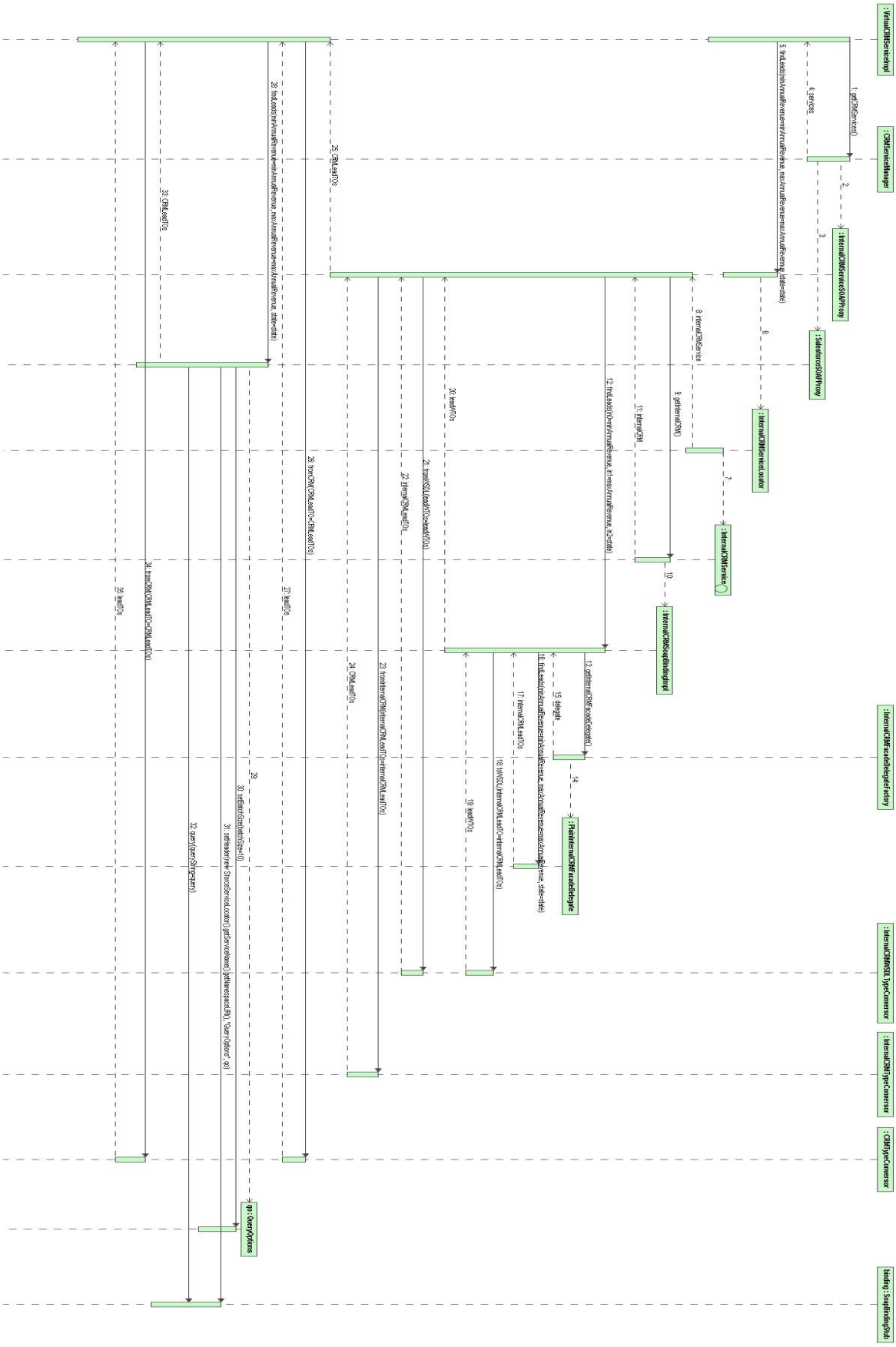
2. Diseño

2.1. Arquitectura global

A continuación se muestra un diagrama con las principales clases que intervienen a la hora de responder a una llamada a la operación *findLeads* del interfaz *VirtualCRMService*, que luego se irán detallando por separado en otros diagramas.


package Data(Clases Principales)
VirtualCRMServiceFactory
-CLASS_NAME_PARAMETER : String = "VirtualCRMServiceFactory/className"(readOnly)
-instance : VirtualCRMService
<<constructor>>-VirtualCRMServiceFactory()
<<getter>>+getVirtualCRMService() : VirtualCRMService
VirtualCRMService
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->LeadTO>
+findLeads(annualRevenueType : RevenueType, state : String) : List<E->LeadTO>
VirtualCRMServiceImpl
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->LeadTO>
+findLeads(annualRevenueType : RevenueType, state : String) : List<E->LeadTO>
<<getter>>-getMaxAnnualRevenue(annualRevenueType : RevenueType) : double
<<getter>>-getMinAnnualRevenue(annualRevenueType : RevenueType) : double
CRMServiceManager
-CLASS_NAME_PARAMETER : String = "CRMServiceManager/className"(readOnly)
-services : List<E->CRMService>
<<getter>>+getCRMServices() : List<E->CRMService>
<<constructor>>-CRMServiceManager()
CRMService
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->CRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->CRMLeadTO>
InternalCRMServiceSOAPProxy
-ENDPOINT_ADDRESS_PARAMETER : String = "InternalCRMServiceSOAPProxy/endPointAddress"(readOnly)
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->CRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->CRMLeadTO>
SalesforceSOAPProxy
-binding : SoapBindingStub
<<constructor>>+SalesforceSOAPProxy()
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->CRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->CRMLeadTO>
<<use>>
com
sforce
soap
enterprise
InternalCRMFacadeDelegate
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->InternalCRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->InternalCRMLeadTO>
PlainInternalCRMFacadeDelegate
-internalCRMLeadTOs : List<E->InternalCRMLeadTO>
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->InternalCRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->InternalCRMLeadTO>
<<use>>
InternalCRM
+findLastLeads(in0 : Calendar, in1 : Calendar) : LeadWTO[]*
+findLeads(in0 : double, in1 : double, in2 : String) : LeadWTO[]*
InternalCRMSoapBindingImpl
+findLastLeads(startDate : Calendar, endDate : Calendar) : LeadWTO[]*
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : LeadWTO[]*
<<use>>
InternalCRMLeadTO
-name : String
-company : String
-annualRevenue : double
-phone : String
-street : String
-postalCode : String
-city : String
-state : String
-country : String
-insertionDate : Calendar
<<constructor>>+InternalCRMLeadTO()
<<constructor>>+InternalCRMLeadTO(name : String, company : String, annualRevenue : double, phone : String, street : String, postalCode : String, city : String, state : String, country : String, insertionDate : Calendar)
<<JavaElement>>+equals(obj : Object) : boolean(JavaAnnotations = "@Override")
<<getter>>+getAnnualRevenue() : double
<<getter>>+getCity() : String
<<getter>>+getCompany() : String
<<getter>>+getCountry() : String
<<getter>>+getInsertionDate() : Calendar
<<getter>>+getName() : String
<<getter>>+getPhone() : String
<<getter>>+getPostalCode() : String
<<getter>>+getState() : String
<<getter>>+getStreet() : String
<<JavaElement>>+toString() : String(JavaAnnotations = "@Override")
InternalCRMFacadeDelegateFactory
-DELEGATE_CLASS_NAME_PARAMETER : String = "InternalCRMFacadeDelegateFactory/delegateClassName"(readOnly)
-delegateClass : Class
<<getter>>-getDelegateClass() : Class
<<getter>>+getInternalCRMFacadeDelegate() : InternalCRMFacadeDelegate
<<constructor>>-InternalCRMFacadeDelegateFactory()
<<use>>
<<use>>
<<use>>
InternalCRMTypeConversor
+fromInternalCRM(internalCRMLeadTO : InternalCRMLeadTO) : CRMLeadTO
+fromInternalCRM(internalCRMLeadTOs : List<E->InternalCRMLeadTO>) : List<E->CRMLeadTO>
+toInternalCRM(CRMLeadTO : CRMLeadTO) : InternalCRMLeadTO
+toInternalCRM(CRMLeadTOs : List<E->CRMLeadTO>) : List<E->InternalCRMLeadTO>
<<constructor>>-InternalCRMTypeConversor()
InternalCRMWSDLTypeConversor
+fromWSDL(leadWTO : LeadWTO) : InternalCRMLeadTO
+fromWSDL(leadWTOs : LeadWTO[]*) : List<E->InternalCRMLeadTO>
+toWSDL(internalCRMLeadTO : InternalCRMLeadTO) : LeadWTO
+toWSDL(internalCRMLeadTOs : List<E->InternalCRMLeadTO>) : LeadWTO[]*
<<constructor>>-InternalCRMWSDLTypeConversor()
<<use>>
<<use>>
LeadWTO
-annualRevenue : double
-city : String
-company : String
-country : String
-insertionDate : Calendar
-name : String
-phone : String
-postalCode : String
-state : String
-street : String
-__equalsCalc : Object = null
-__hashCodeCalc : boolean = false
-typeDesc : TypeDesc = new org.apache.axis.description.TypeDesc(LeadWTO.class,true)
<<constructor>>+LeadWTO()
<<constructor>>+LeadWTO(annualRevenue : double, city : String, company : String, country : String, insertionDate : Calendar, name : String, phone : String, postalCode : String, state : String, street : String)
<<getter>>+getAnnualRevenue() : double
<<setter>>+setAnnualRevenue(annualRevenue : double) : void
<<getter>>+getCity() : String
<<setter>>+setCity(city : String) : void
<<getter>>+getCompany() : String
<<setter>>+setCompany(company : String) : void
<<getter>>+getCountry() : String
<<setter>>+setCountry(country : String) : void
<<getter>>+getInsertionDate() : Calendar
<<setter>>+setInsertionDate(insertionDate : Calendar) : void
<<getter>>+getName() : String
<<setter>>+setName(name : String) : void
<<getter>>+getPhone() : String
<<setter>>+setPhone(phone : String) : void
<<getter>>+getPostalCode() : String
<<setter>>+setPostalCode(postalCode : String) : void
<<getter>>+getState() : String
<<setter>>+setState(state : String) : void
<<getter>>+getStreet() : String
<<setter>>+setStreet(street : String) : void
+equals(obj : Object) : boolean(guarded)
+hashCode() : int(guarded)
<<getter>>+getTypeDesc() : TypeDesc
<<getter>>+getSerializer(mechType : String, _javaType : Class, _xmlType : QName) : Serializer
<<getter>>+getDeserializer(mechType : String, _javaType : Class, _xmlType : QName) : Deserializer
CRMLeadTO
-firstName : String
-lastName : String
-company : String
-annualRevenue : double
-phone : String
-street : String
-postalCode : String
-city : String
-state : String
-country : String
-insertionDate : Calendar
<<constructor>>+CRMLeadTO()
<<constructor>>+CRMLeadTO(firstName : String, lastName : String, company : String, annualRevenue : double, phone : String, street : String, postalCode : String, city : String, state : String, country : String, insertionDate : Calendar)
<<JavaElement>>+equals(obj : Object) : boolean(JavaAnnotations = "@Override")
<<getter>>+getAnnualRevenue() : double
<<getter>>+getCity() : String
<<getter>>+getCompany() : String
<<getter>>+getCountry() : String
<<getter>>+getFirstName() : String
<<getter>>+getInsertionDate() : Calendar
<<getter>>+getLastName() : String
<<getter>>+getPhone() : String
<<getter>>+getPostalCode() : String
<<getter>>+getState() : String
<<getter>>+getStreet() : String
<<JavaElement>>+toString() : String(JavaAnnotations = "@Override")

Las clases anteriores interactúan según el siguiente diagrama de secuencia:



En primer lugar, la implementación del interfaz *VirtualCRMService*, en este caso la clase *VirtualCRMServiceImpl*, le pide a la clase *CRMServiceManager* todas las clases que implementan el interfaz *CRMService*.

El *CRMServiceManager* busca en el fichero *ConfigurationParameters.properties* el nombre de dichas clases, las instancia y se las devuelve a la clase *VirtualCRMServiceImpl*.

Ahora, la clase *VirtualCRMServiceImpl* puede ir llamando a cada una de las clases que le han devuelto e ir añadiendo en una lista, que luego devolverá, los resultados devueltos por la invocación de la operación *findLeads* sobre cada una de estas clases.

Primero, en este caso, llama a la operación *findLeads* sobre la clase *InternalCRMServiceSOAPProxy*. Esta clase, primero crea un *InternalCRMServiceLocator*, que crea a su vez un *InternalCRMService*, sobre el cual se invoca la operación *getInternalCRM*, que devuelve una instancia, en este caso, de la clase *InternalCRMSoapBindingImpl*.

Ahora, se puede invocar sobre esta clase la operación *findLeads*. Como el CRM interno está implementado con una fachada, primero hay que conseguir una instancia de dicha fachada, lo cual se consigue llamando a la factoría *InternalCRMFacadeDelegateFactory*, que devuelve una implementación del interfaz *InternalCRMFacadeDelegate*. En este caso, se devuelve una instancia de la clase *PlainInternalCRMFacadeDelegate*, sobre la que ya se invoca la operación *findLeads*.

Esta invocación devuelve una lista de objetos de la clase *InternalCRMLeadTO*. Como se necesitan objetos de la clase *LeadWTO*, hay que hacer una conversión para pasar de *InternalCRMLeadTO* a *LeadWTO*, usando para ello la clase *InternalCRMWSDLTypeConversor* y dicha lista de objetos es devuelta a la clase *InternalCRMServiceSOAPProxy*.

Como las implementaciones del interfaz *CRMService* usan objetos de la clase *CRMLeadTO*, hay que hacer un par de conversiones: primero una para volver a convertir los objetos *LeadWTO* a *InternalCRMLeadTO*, usando para ello la clase *InternalCRMWSDLTypeConversor* y luego otra conversión para pasar de *InternalCRMLeadTO* a *CRMLeadTO*, usando para ello la clase *InternalCRMTypeConversor*. Una vez convertidos los objetos ya se pueden devolver a la clase *VirtualCRMServiceImpl*, la cual, tras convertirlos a *LeadTO*, usando la clase *CRMTypeConversor*, los añade a la lista de objetos de tipo *LeadTO*, que será la que se devuelva finalmente.

En ese momento, habría que hacer una invocación sobre la clase que implementa el acceso a Salesforce, en este caso la clase *SalesforceSOAPProxy.*

Esta clase usa varias de las clases del paquete *com.sforce.soap.enterprise* para conseguir construir los objetos con los datos de los clientes.

En primer lugar, en el constructor se crea un objeto *binding* de la clase *SoapBindingStub* y se establece diversas propiedades sobre él, como el ID de la sesión y las opciones de la *query* (tales como el número de resultados que devuelve de cada vez).

En la operación *findLeads*, se construye la *query* y se consigue un objeto de la clase *QueryResult*, que contiene una serie de objetos de la clase *Lead*, que tienen que ser convertidos a objetos de la clase *CRMLeadTO*, que son los que se devuelven a la clase *VirtualCRMServiceImpl*.

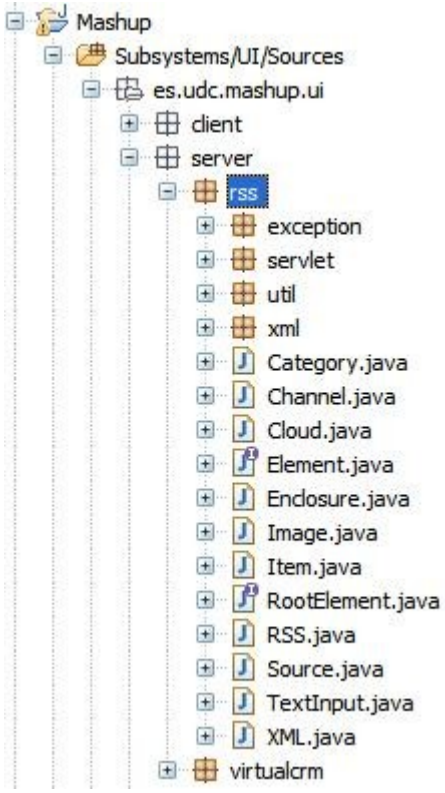
Otra vez, hay que hacer una conversión de objetos de la clase *CRMLeadTO* a objetos de la clase *LeadTO*, usando la clase *CRMTypeConversor*, para luego añadirlos a los objetos obtenidos en la invocación anterior.

En este caso, sólo se añaden los objetos al final y no se comprueba si existen duplicados y tampoco se ordenan de ninguna forma.

2.2. Subsistema UI

En este subsistema, además de la interfaz gráfica que ya venía implementada, se ha implementado el servicio de RSS.

La estructura de paquetes de este subsistema es la siguiente:



En el paquete *exception* se encuentran una serie de clases que representan las diversas excepciones que se pueden dar al crear un objeto *XML*.

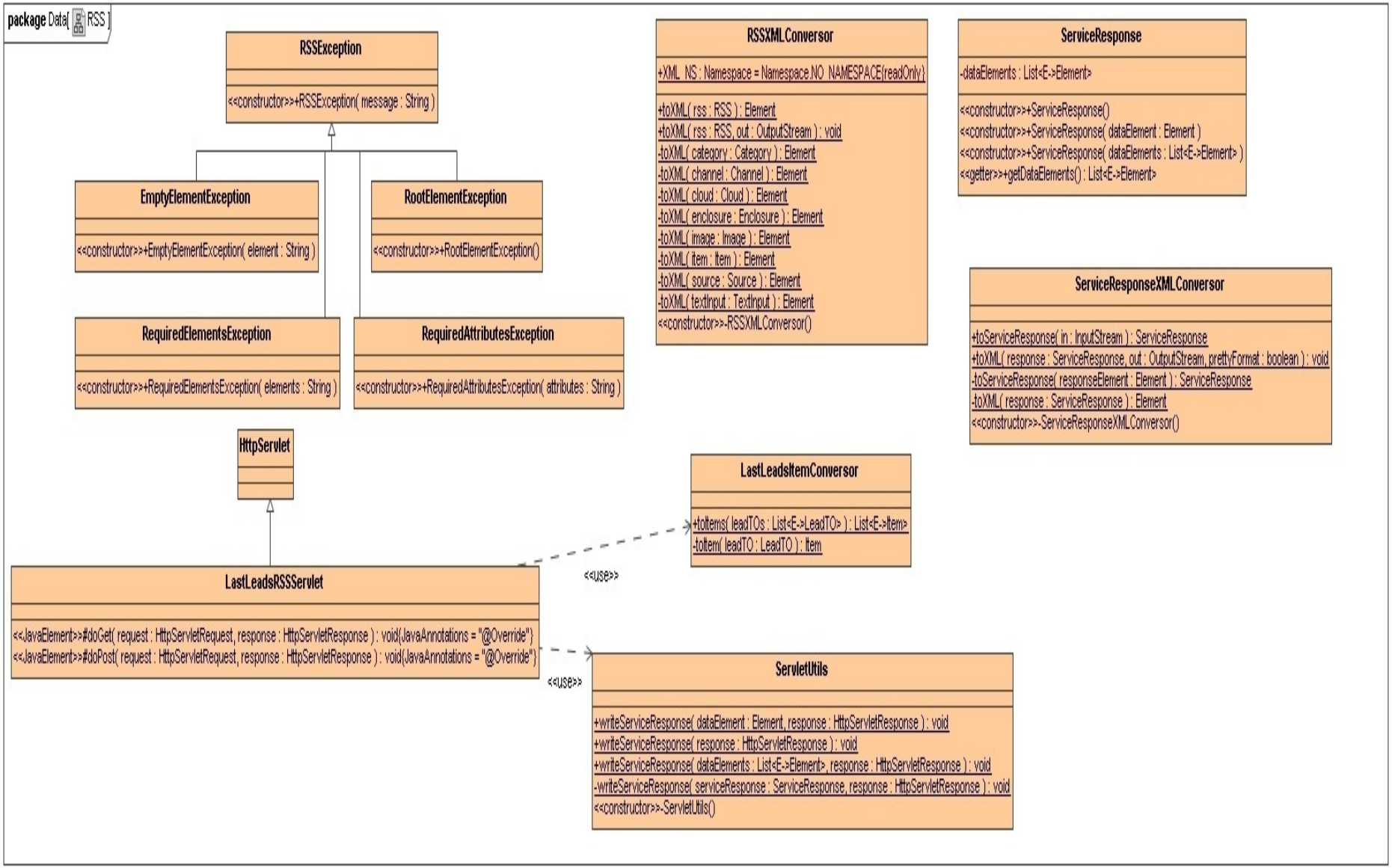
En el paquete *servlet* se encuentra el servlet que responde a la invocación de la operación *findLastLeads* y también se encuentra la clase *ServletUtils*, que se utiliza para escribir en la *response*.

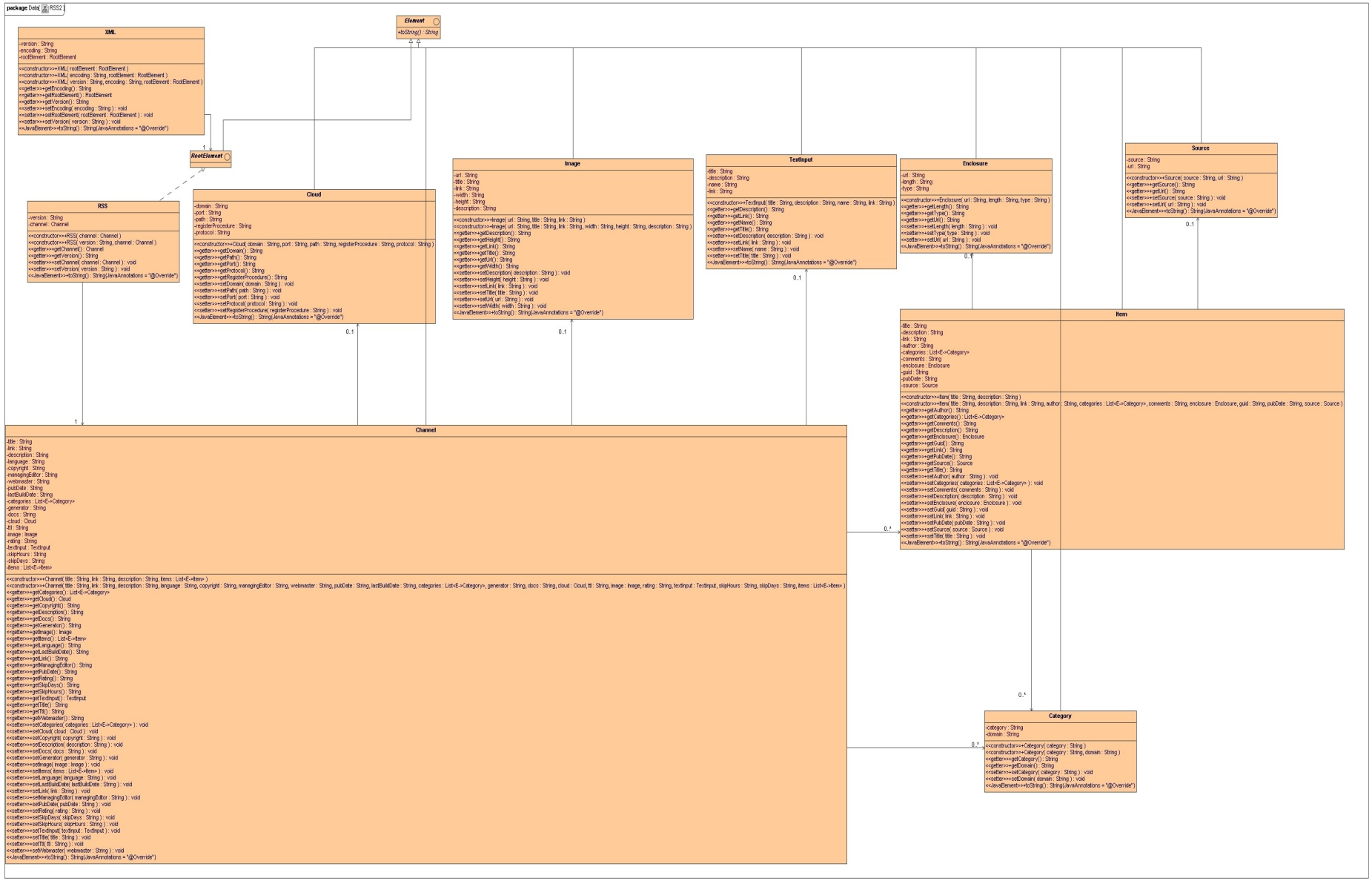
En el paquete *util* se encuentra la clase *LastLeadsItemConversor* que transforma una lista de objetos de la clase *LeadTO* en una lista de objetos de la clase *Item*.

En el paquete *xml* se encuentran tres clases que sirven para, usando JDOM, transformar un objeto de la clase *RSS* a un objeto de la clase *Element*, para que luego pueda ser escrito por la clase *ServletUtils* en la *response*.

Por último, las demás clases de este subsistema, implementan la jerarquía de los documentos XML (y en concreto la de los documentos RSS).

A continuación se muestra dos diagramas de clase con las clases de este subsistema.

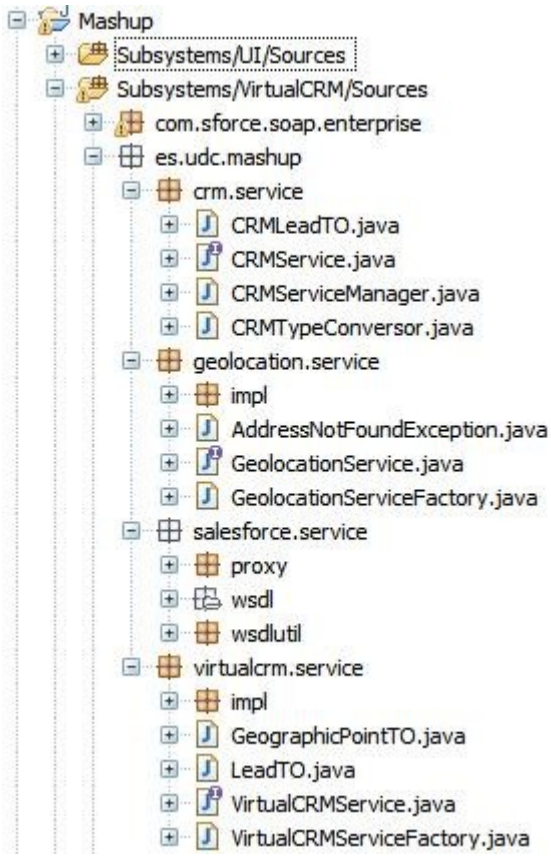

package Data[RSS]
RSSException
<<constructor>>+RSSException(message : String)
EmptyElementException
<<constructor>>+EmptyElementException(element : String)
RootElementException
<<constructor>>+RootElementException()
RequiredElementsException
<<constructor>>+RequiredElementsException(elements : String)
RequiredAttributesException
<<constructor>>+RequiredAttributesException(attributes : String)
HttpServlet
LastLeadsRSSServlet
<<JavaElement>>#doGet(request : HttpServletRequest, response : HttpServletResponse) : void{JavaAnnotations = "@Override"}
<<JavaElement>>#doPost(request : HttpServletRequest, response : HttpServletResponse) : void{JavaAnnotations = "@Override"}
<<use>>
LastLeadsItemConversor
+toItems(leadTOs : List<E->LeadTO>) : List<E->Item>
-toItem(leadTO : LeadTO) : Item
<<use>>
ServletUtils
+writeServiceResponse(dataElement : Element, response : HttpServletResponse) : void
+writeServiceResponse(response : HttpServletResponse) : void
+writeServiceResponse(dataElements : List<E->Element>, response : HttpServletResponse) : void
-writeServiceResponse(serviceResponse : ServiceResponse, response : HttpServletResponse) : void
<<constructor>>-ServletUtils()
RSSXMLConversor
+XML_NS : Namespace = Namespace.NO_NAMESPACE{readOnly}
+toXML(rss : RSS) : Element
+toXML(rss : RSS, out : OutputStream) : void
-toXML(category : Category) : Element
-toXML(channel : Channel) : Element
-toXML(cloud : Cloud) : Element
-toXML(enclosure : Enclosure) : Element
-toXML(image : Image) : Element
-toXML(item : Item) : Element
-toXML(source : Source) : Element
-toXML(textInput : TextInput) : Element
<<constructor>>-RSSXMLConversor()
ServiceResponse
-dataElements : List<E->Element>
<<constructor>>+ServiceResponse()
<<constructor>>+ServiceResponse(dataElement : Element)
<<constructor>>+ServiceResponse(dataElements : List<E->Element>)
<<getter>>+getDataElements() : List<E->Element>
ServiceResponseXMLConversor
+toServiceResponse(in : InputStream) : ServiceResponse
+toXML(response : ServiceResponse, out : OutputStream, prettyFormat : boolean) : void
-toServiceResponse(responseElement : Element) : ServiceResponse
-toXML(response : ServiceResponse) : Element
<<constructor>>-ServiceResponseXMLConversor()


package Data[RSS2]
Element
+toString() : String
XML
-version : String
-encoding : String
-rootElement : RootElement
<<constructor>>+XML(rootElement : RootElement)
<<constructor>>+XML(encoding : String, rootElement : RootElement)
<<constructor>>+XML(version : String, encoding : String, rootElement : RootElement)
<<getter>>+getEncoding() : String
<<getter>>+getRootElement() : RootElement
<<getter>>+getVersion() : String
<<setter>>+setEncoding(encoding : String) : void
<<setter>>+setRootElement(rootElement : RootElement) : void
<<setter>>+setVersion(version : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
RootElement
RSS
-version : String
-channel : Channel
<<constructor>>+RSS(channel : Channel)
<<constructor>>+RSS(version : String, channel : Channel)
<<getter>>+getChannel() : Channel
<<getter>>+getVersion() : String
<<setter>>+setChannel(channel : Channel) : void
<<setter>>+setVersion(version : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
Cloud
-domain : String
-port : String
-path : String
-registerProcedure : String
-protocol : String
<<constructor>>+Cloud(domain : String, port : String, path : String, registerProcedure : String, protocol : String)
<<getter>>+getDomain() : String
<<getter>>+getPath() : String
<<getter>>+getPort() : String
<<getter>>+getProtocol() : String
<<getter>>+getRegisterProcedure() : String
<<setter>>+setDomain(domain : String) : void
<<setter>>+setPath(path : String) : void
<<setter>>+setPort(port : String) : void
<<setter>>+setProtocol(protocol : String) : void
<<setter>>+setRegisterProcedure(registerProcedure : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
Image
-url : String
-title : String
-link : String
-width : String
-height : String
-description : String
<<constructor>>+Image(url : String, title : String, link : String)
<<constructor>>+Image(url : String, title : String, link : String, width : String, height : String, description : String)
<<getter>>+getDescription() : String
<<getter>>+getHeight() : String
<<getter>>+getLink() : String
<<getter>>+getTitle() : String
<<getter>>+getUrl() : String
<<getter>>+getWidth() : String
<<setter>>+setDescription(description : String) : void
<<setter>>+setHeight(height : String) : void
<<setter>>+setLink(link : String) : void
<<setter>>+setTitle(title : String) : void
<<setter>>+setUrl(url : String) : void
<<setter>>+setWidth(width : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
TextInput
-title : String
-description : String
-name : String
-link : String
<<constructor>>+TextInput(title : String, description : String, name : String, link : String)
<<getter>>+getDescription() : String
<<getter>>+getLink() : String
<<getter>>+getName() : String
<<getter>>+getTitle() : String
<<setter>>+setDescription(description : String) : void
<<setter>>+setLink(link : String) : void
<<setter>>+setName(name : String) : void
<<setter>>+setTitle(title : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
Enclosure
-url : String
-length : String
-type : String
<<constructor>>+Enclosure(url : String, length : String, type : String)
<<getter>>+getLength() : String
<<getter>>+getType() : String
<<getter>>+getUrl() : String
<<setter>>+setLength(length : String) : void
<<setter>>+setType(type : String) : void
<<setter>>+setUrl(url : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
Source
-source : String
-url : String
<<constructor>>+Source(source : String, url : String)
<<getter>>+getSource() : String
<<getter>>+getUrl() : String
<<setter>>+setSource(source : String) : void
<<setter>>+setUrl(url : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
Item
-title : String
-description : String
-link : String
-author : String
-categories : List<E->Category>
-comments : String
-enclosure : Enclosure
-guid : String
-pubDate : String
-source : Source
<<constructor>>+Item(title : String, description : String)
<<constructor>>+Item(title : String, description : String, link : String, author : String, categories : List<E->Category>, comments : String, enclosure : Enclosure, guid : String, pubDate : String, source : Source)
<<getter>>+getAuthor() : String
<<getter>>+getCategories() : List<E->Category>
<<getter>>+getComments() : String
<<getter>>+getDescription() : String
<<getter>>+getEnclosure() : Enclosure
<<getter>>+getGuid() : String
<<getter>>+getLink() : String
<<getter>>+getPubDate() : String
<<getter>>+getSource() : Source
<<getter>>+getTitle() : String
<<setter>>+setAuthor(author : String) : void
<<setter>>+setCategories(categories : List<E->Category>) : void
<<setter>>+setComments(comments : String) : void
<<setter>>+setDescription(description : String) : void
<<setter>>+setEnclosure(enclosure : Enclosure) : void
<<setter>>+setGuid(guid : String) : void
<<setter>>+setLink(link : String) : void
<<setter>>+setPubDate(pubDate : String) : void
<<setter>>+setSource(source : Source) : void
<<setter>>+setTitle(title : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
Channel
-title : String
-link : String
-description : String
-language : String
-copyright : String
-managingEditor : String
-webmaster : String
-pubDate : String
-lastBuildDate : String
-categories : List<E->Category>
-generator : String
-docs : String
-cloud : Cloud
-ttl : String
-image : Image
-rating : String
-textInput : TextInput
-skipHours : String
-skipDays : String
-items : List<E->Item>
<<constructor>>+Channel(title : String, link : String, description : String, items : List<E->Item>)
<<constructor>>+Channel(title : String, link : String, description : String, language : String, copyright : String, managingEditor : String, webmaster : String, pubDate : String, lastBuildDate : String, categories : List<E->Category>, generator : String, docs : String, cloud : Cloud, ttl : String, image : Image, rating : String, textInput : TextInput, skipHours : String, skipDays : String, items : List<E->Item>)
<<getter>>+getCategories() : List<E->Category>
<<getter>>+getCloud() : Cloud
<<getter>>+getCopyright() : String
<<getter>>+getDescription() : String
<<getter>>+getDocs() : String
<<getter>>+getGenerator() : String
<<getter>>+getImage() : Image
<<getter>>+getItems() : List<E->Item>
<<getter>>+getLanguage() : String
<<getter>>+getLastBuildDate() : String
<<getter>>+getLink() : String
<<getter>>+getManagingEditor() : String
<<getter>>+getPubDate() : String
<<getter>>+getRating() : String
<<getter>>+getSkipDays() : String
<<getter>>+getSkipHours() : String
<<getter>>+getTextInput() : TextInput
<<getter>>+getTitle() : String
<<getter>>+getTtl() : String
<<getter>>+getWebmaster() : String
<<setter>>+setCategories(categories : List<E->Category>) : void
<<setter>>+setCloud(cloud : Cloud) : void
<<setter>>+setCopyright(copyright : String) : void
<<setter>>+setDescription(description : String) : void
<<setter>>+setDocs(docs : String) : void
<<setter>>+setGenerator(generator : String) : void
<<setter>>+setImage(image : Image) : void
<<setter>>+setItems(items : List<E->Item>) : void
<<setter>>+setLanguage(language : String) : void
<<setter>>+setLastBuildDate(lastBuildDate : String) : void
<<setter>>+setLink(link : String) : void
<<setter>>+setManagingEditor(managingEditor : String) : void
<<setter>>+setPubDate(pubDate : String) : void
<<setter>>+setRating(rating : String) : void
<<setter>>+setSkipDays(skipDays : String) : void
<<setter>>+setSkipHours(skipHours : String) : void
<<setter>>+setTextInput(textInput : TextInput) : void
<<setter>>+setTitle(title : String) : void
<<setter>>+setTtl(ttl : String) : void
<<setter>>+setWebmaster(webmaster : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
Category
-category : String
-domain : String
<<constructor>>+Category(category : String)
<<constructor>>+Category(category : String, domain : String)
<<getter>>+getCategory() : String
<<getter>>+getDomain() : String
<<setter>>+setCategory(category : String) : void
<<setter>>+setDomain(domain : String) : void
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
1
0..1
0..*

2.3. Subsistema VirtualCRM

En este subsistema, se implementan tanto el servicio de búsqueda de Salesforce, como el servicio de geolocalización de Google Maps, además de implementar una abstracción para ocultar el uso de distintos CRMs.

La estructura de paquetes de este subsistema se muestra a continuación:



En el paquete *com.sforce.soap.enterprise* se encuentran las clases generadas a partir del WSDL de Salesforce.

En el paquete *crm.service* se encuentran las clases encargadas de ocultar la utilización de distintos CRMs. La clase *CRMServiceManager* es la encargada de buscar en el fichero *ConfigurationParameters.properties* los nombres de las distintas clases que implementan el interfaz *CRMService* y, una vez instanciadas, las devuelve a la implementación del interfaz *VirtualCRMService.* Además, también hay una clase (*CRMTypeConversor)* que convierte los objetos *CRMLeadTO* (usados por los distintas implementaciones de CRMs) en objetos *LeadTO* (usados por la implementación del *VirtualCRMService)*.

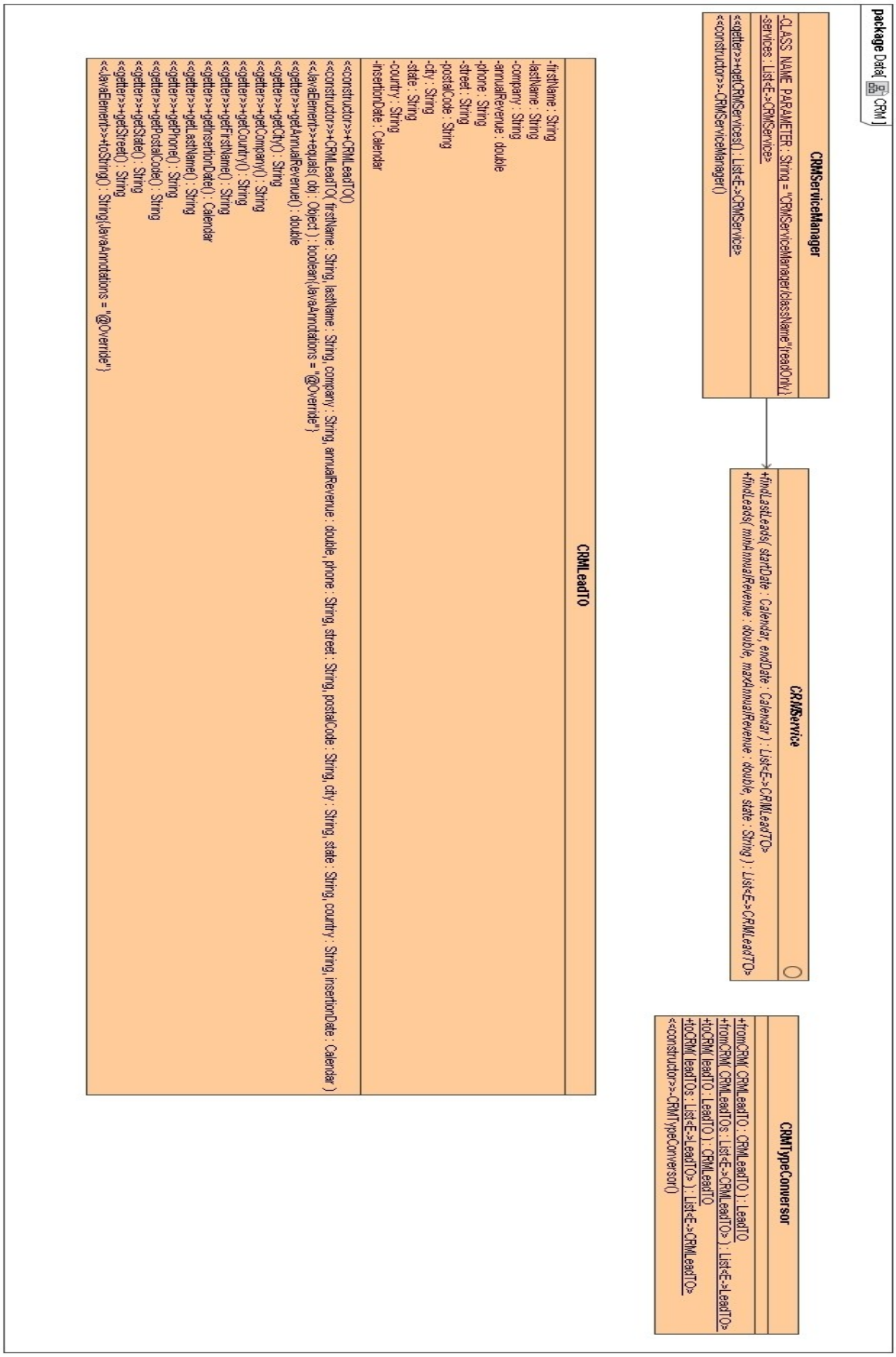
En el paquete *geolocation.service* se encuentra el servicio de geolocalización, que oculta el servicio concreto usado gracias a un interfaz y a una factoría. La implementación concreta del servicio de geolocalización de Google Maps se encuentra en el paquete *impl*.

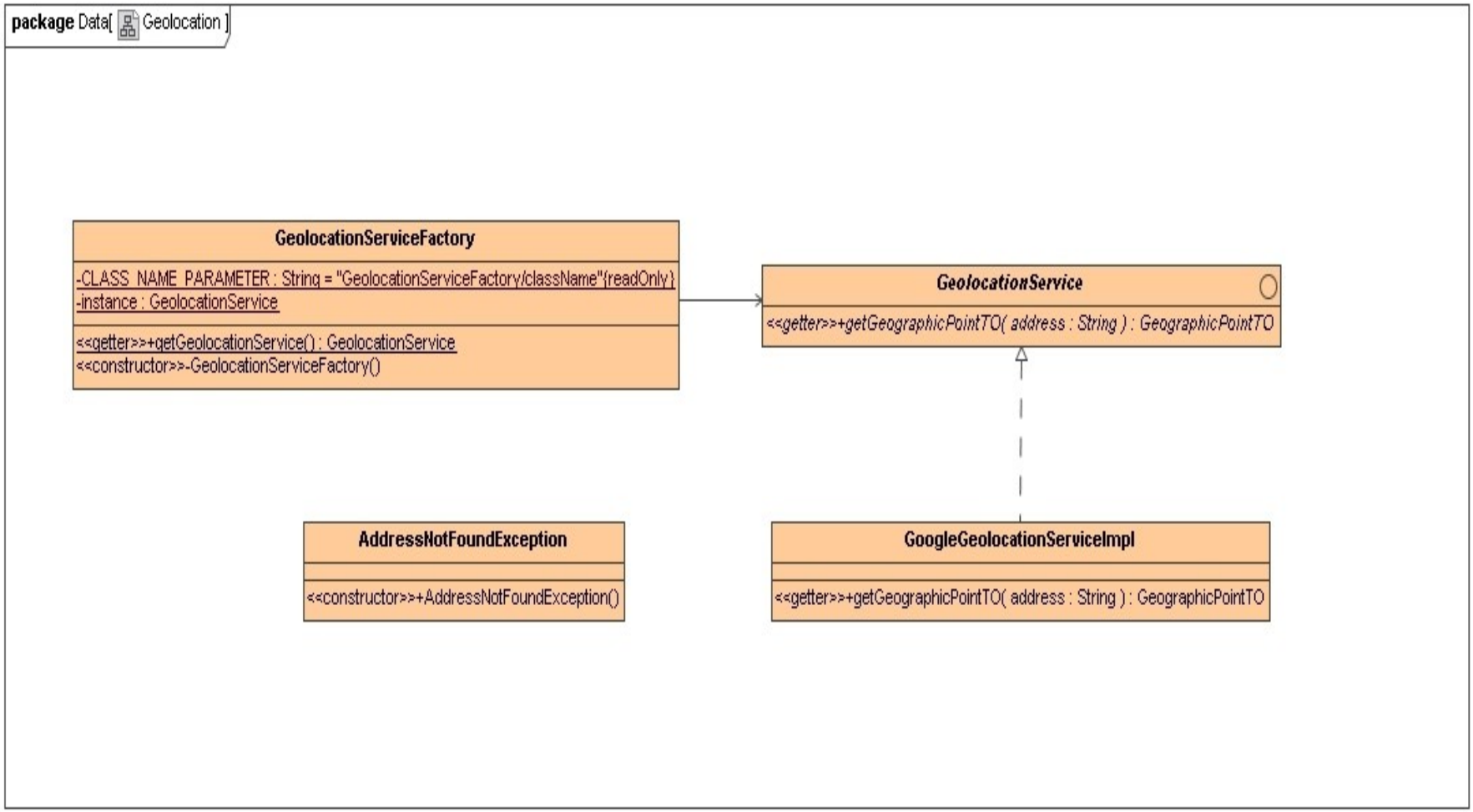
En el paquete *salesforce.service* se encuentra la implementación del servicio de búsqueda de Salesforce. En el paquete *proxy* se encuentra la implementación del interfaz *CRMService*. En el paquete *wsdl* se encuentra el fichero WSDL a partir del cual se generan las clases necesarias para usar el servicio de Salesforce (estas clases se generan en el paquete *com.sforce.soap.enterprise*). Por último, también hay una clase que ayuda a convertir los objetos *Lead* (usados por Salesforce) en objetos *LeadTO* (usados por la implementación del *VirtualCRMService)*.

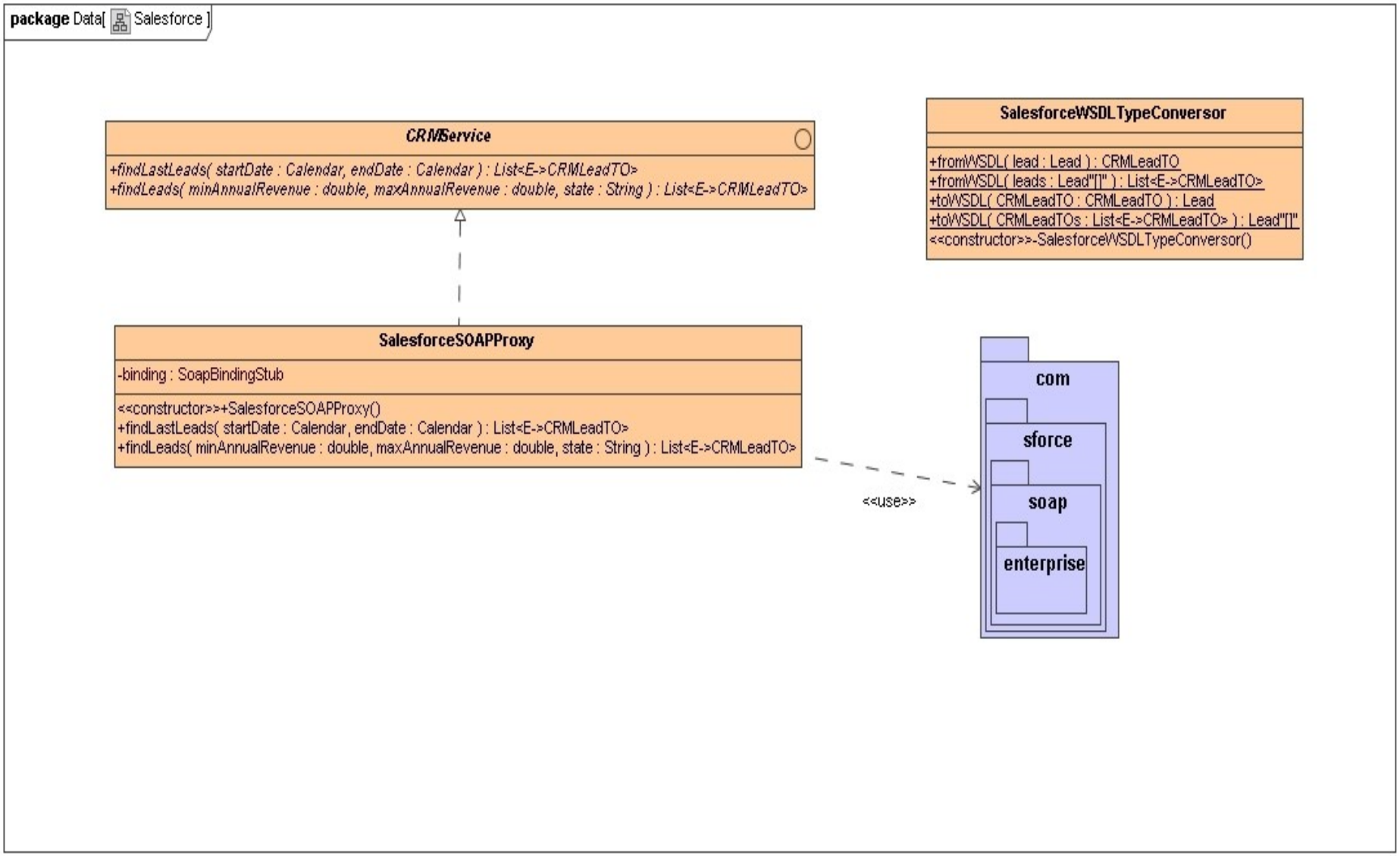
En el paquete *virtualcrm.service* se modificaron la clase *VirtualCRMService* (añadiendo una operación para encontrar los clientes insertados entre dos fechas) y la clase *LeadTO* añadiendo un

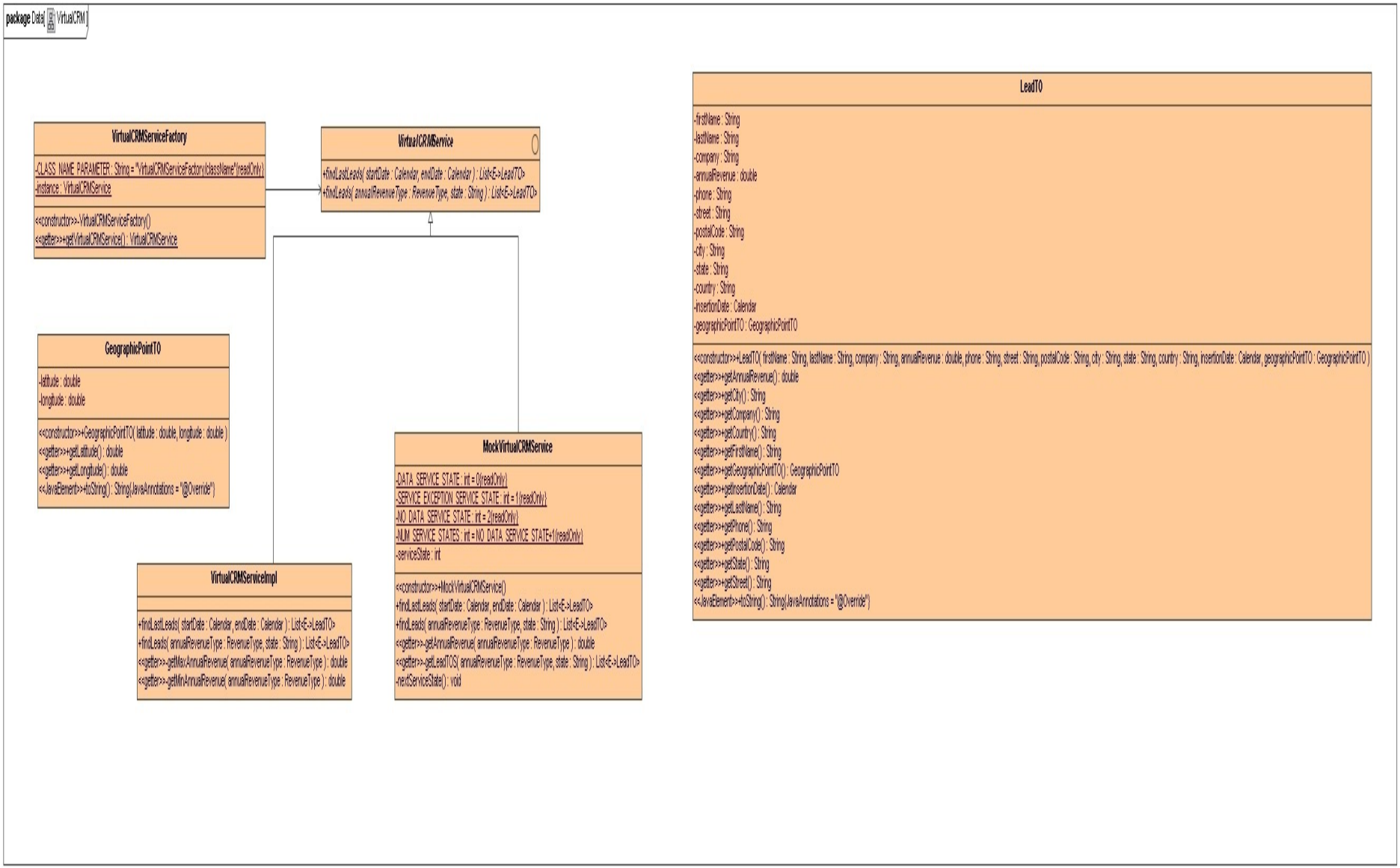
atributo de tipo *Calendar* para poder implementar el servicio RSS. Además, en el paquete *impl* se proporciona una implementación del interfaz *VirtualCRMService* (*VirtualCRMServiceImpl*).

A continuación se muestran varios diagramas con las clases de este subsistema:




package Data[Geolocation]
GeolocationServiceFactory
-CLASS_NAME_PARAMETER : String = "GeolocationServiceFactory/className"{readOnly}
-instance : GeolocationService
<<getter>>+getGeolocationService() : GeolocationService
<<constructor>>-GeolocationServiceFactory()
GeolocationService
<<getter>>+getGeographicPointTO(address : String) : GeographicPointTO
AddressNotFoundException
<<constructor>>+AddressNotFoundException()
GoogleGeolocationServiceImpl
<<getter>>+getGeographicPointTO(address : String) : GeographicPointTO

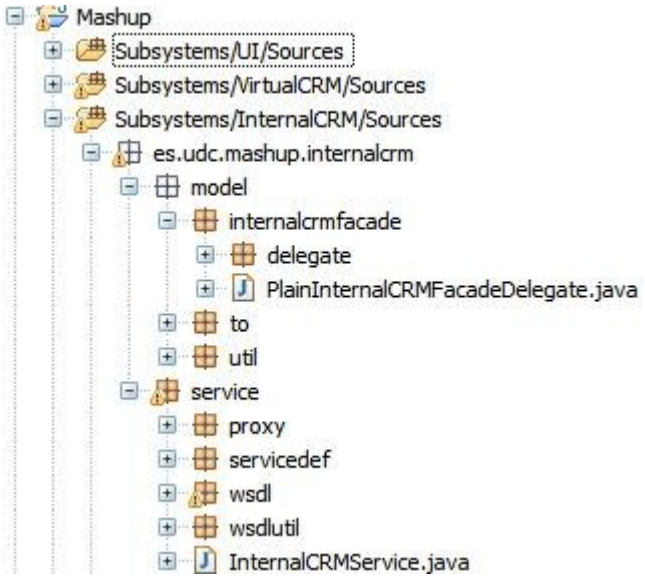

package Data[Salesforce]
CRMService
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->CRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->CRMLeadTO>
SalesforceWSDLTypeConversor
+fromWSDL(lead : Lead) : CRMLeadTO
+fromWSDL(leads : Lead"[]") : List<E->CRMLeadTO>
+toWSDL(CRMLeadTO : CRMLeadTO) : Lead
+toWSDL(CRMLeadTOs : List<E->CRMLeadTO>) : Lead"[]"
<<constructor>>-SalesforceWSDLTypeConversor()
SalesforceSOAPProxy
-binding : SoapBindingStub
<<constructor>>+SalesforceSOAPProxy()
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->CRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->CRMLeadTO>
<<use>>
com
sforce
soap
enterprise


package Data [VirtualCRM]
VirtualCRMServiceFactory
-CLASS_NAME_PARAMETER : String = "VirtualCRMServiceFactoryclassName"(readOnly)
-instance : VirtualCRMService
<<constructor>>-VirtualCRMServiceFactory()
<<getter>>+getVirtualCRMService() : VirtualCRMService
VirtualCRMService
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->LeadTO>
+findLeads(annualRevenueType : RevenueType, state : String) : List<E->LeadTO>
LeadTO
-firstName : String
-lastName : String
-company : String
-annualRevenue : double
-phone : String
-street : String
-postalCode : String
-city : String
-state : String
-country : String
-insertionDate : Calendar
-geographicPointTO : GeographicPointTO
<<constructor>>+LeadTO(firstName : String, lastName : String, company : String, annualRevenue : double, phone : String, street : String, postalCode : String, city : String, state : String, country : String, insertionDate : Calendar, geographicPointTO : GeographicPointTO)
<<getter>>+getAnnualRevenue() : double
<<getter>>+getCity() : String
<<getter>>+getCompany() : String
<<getter>>+getCountry() : String
<<getter>>+getFirstName() : String
<<getter>>+getGeographicPointTO() : GeographicPointTO
<<getter>>+getInsertionDate() : Calendar
<<getter>>+getLastName() : String
<<getter>>+getPhone() : String
<<getter>>+getPostalCode() : String
<<getter>>+getState() : String
<<getter>>+getStreet() : String
<<JavaElement>>+toString() : String(JavaAnnotations = "@Override")
GeographicPointTO
-latitude : double
-longitude : double
<<constructor>>+GeographicPointTO(latitude : double, longitude : double)
<<getter>>+getLatitude() : double
<<getter>>+getLongitude() : double
<<JavaElement>>+toString() : String(JavaAnnotations = "@Override")
MockVirtualCRMService
-DATA_SERVICE_STATE : int = 0(readOnly)
-SERVICE_EXCEPTION_SERVICE_STATE : int = 1(readOnly)
-NO_DATA_SERVICE_STATE : int = 2(readOnly)
-NUM_SERVICE_STATES : int = NO_DATA_SERVICE_STATE+1(readOnly)
-serviceState : int
<<constructor>>+MockVirtualCRMService()
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->LeadTO>
+findLeads(annualRevenueType : RevenueType, state : String) : List<E->LeadTO>
<<getter>>-getAnnualRevenue(annualRevenueType : RevenueType) : double
<<getter>>-getLeadTOS(annualRevenueType : RevenueType, state : String) : List<E->LeadTO>
-nextServiceState() : void
VirtualCRMServiceImpl
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->LeadTO>
+findLeads(annualRevenueType : RevenueType, state : String) : List<E->LeadTO>
<<getter>>-getMaxAnnualRevenue(annualRevenueType : RevenueType) : double
<<getter>>-getMinAnnualRevenue(annualRevenueType : RevenueType) : double

2.4. Subsistema InternalCRM

En este subsistema se implementa de forma bastante simple un CRM interno. Por una parte se implementa la parte del servidor (con una fachada que atiende a las invocaciones de las operaciones) y por otro la parte del cliente (con la definición de un servicio web que utiliza SOAP).

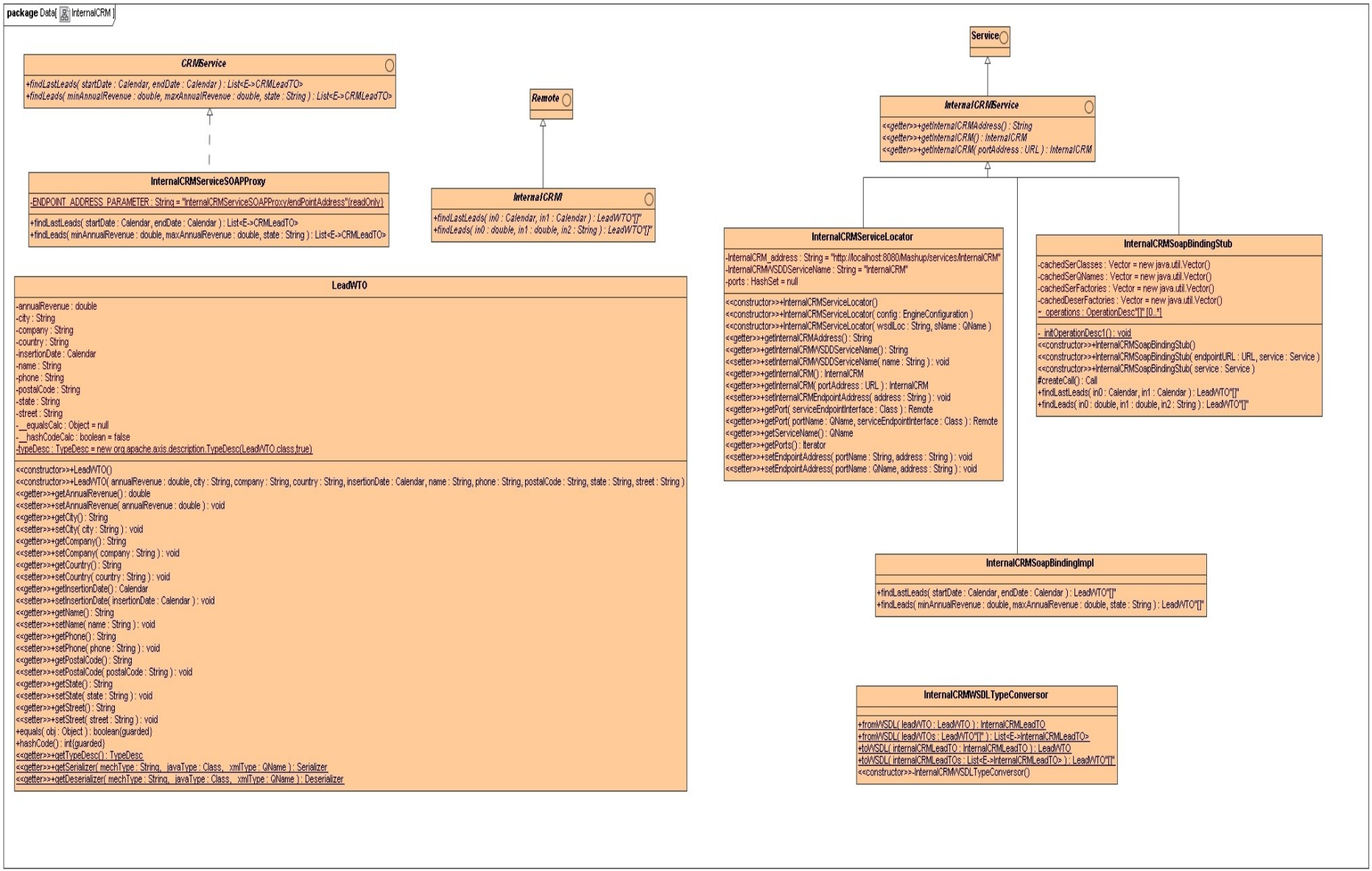
La estructura de paquetes de este subsistema se muestra a continuación:

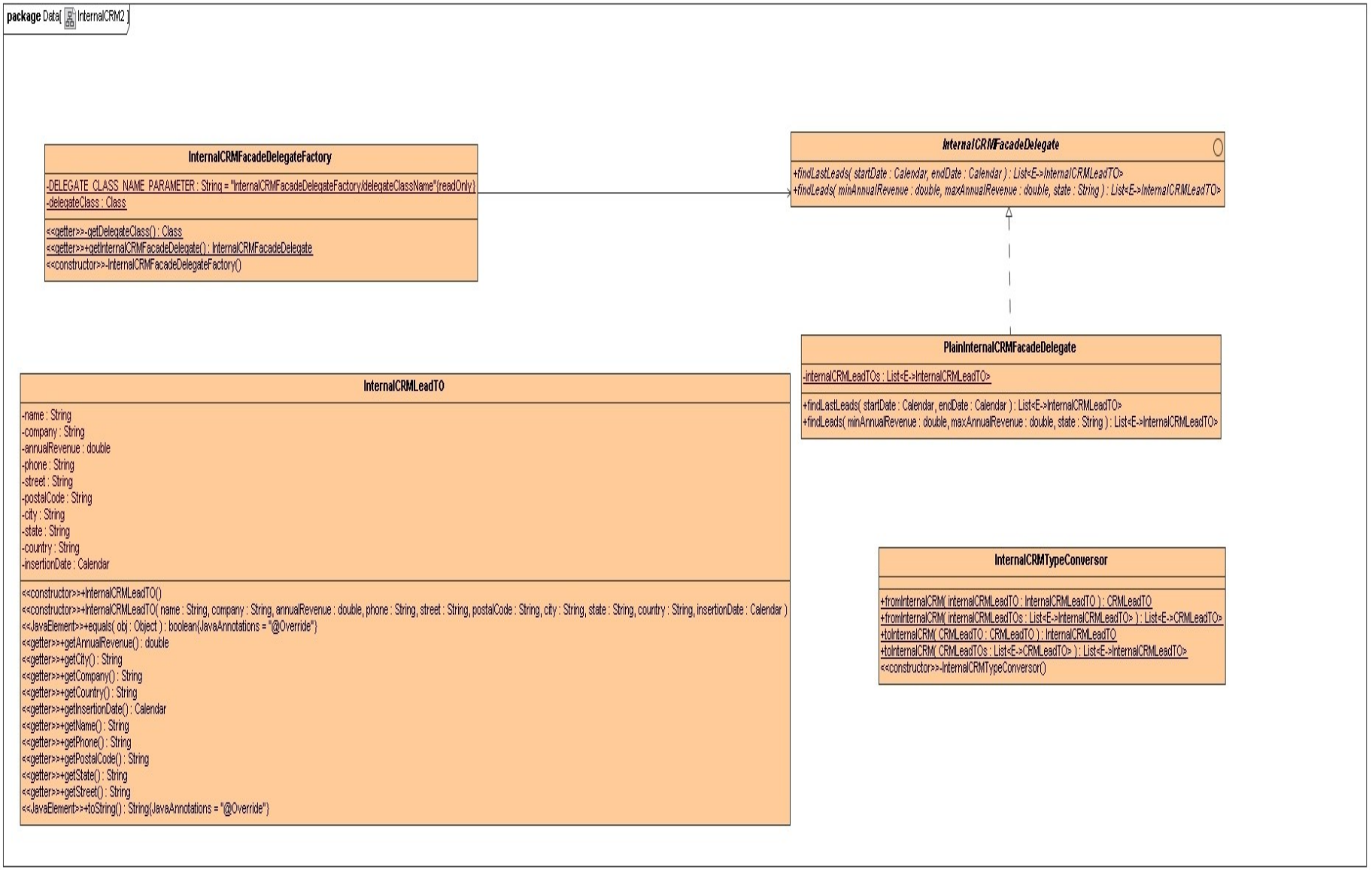


En el paquete *model* se implementa la capa modelo del CRM interno. Para ello se usa una fachada que se encuentra en el paquete *internalcrmfacade*, que usando los patrones de diseño de *Facade y Bussiness Delegate* oculta la lógica de negocio. En el paquete *to* se encuentra el *Transfer Object* utilizado por esta fachada para representar los datos de los clientes incluidos en el CRM. Y en el paquete *util* se encuentra una clase que permite convertir dichos objetos en objetos de la clase *CRMLeadTO*, que son utilizados por la implementación del interfaz *CRMService*. Por último, decir que la implementación de este CRM interno es muy simple, ya que los clientes se crean dentro del código de la clase *PlainInternalCRMFacadeDelegate* y no se leen desde un fichero o desde una base de datos, como sería lógico en una aplicación real.

En el paquete *service* se implementa el servicio web del CRM interno. En el paquete *proxy* se encuentra la clase que implementa el interfaz *CRMService*. En el paquete *servicedef* se encuentran las dos clases necesarias para crear el WSDL con la definición del servicio web. Estas clases son un interfaz y un *Transfer Object*. En el paquete *wsdl* se encuentran las clases generadas a partir del fichero WSDL. Destacar que la clase *InternalCRMSoapBindingImpl* ha sido modificada y se la ha insertado el código que hace posible que devuelva los resultados deseados, llamando para ello a las operaciones de la fachada del CRM interno. Por último, en el paquete *wsdlutil* se encuentra una clase que permite pasar de los *Transfer Objects* usados por el servicio web (*LeadWTO)* a los *Transfer Objects* usados por el CRM interno (*InternalCRMLeadTO)*.

A continuación se muestran dos diagramas con las clases que conforman este subsistema:


package Data[InternalCRM]
CRMService
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->CRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->CRMLeadTO>
InternalCRMServiceSOAPProxy
-ENDPOINT_ADDRESS_PARAMETER : String = "InternalCRMServiceSOAPProxy.endPointAddress"{readOnly}
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->CRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->CRMLeadTO>
Remote
InternalCRM
+findLastLeads(in0 : Calendar, in1 : Calendar) : LeadWTO"[]"
+findLeads(in0 : double, in1 : double, in2 : String) : LeadWTO"[]"
Service
InternalCRMService
<<getter>>+getInternalCRMAddress() : String
<<getter>>+getInternalCRM() : InternalCRM
<<getter>>+getInternalCRM(portAddress : URL) : InternalCRM
InternalCRMServiceLocator
-InternalCRM_address : String = "http://localhost:8080/Mashup/services/InternalCRM"
-InternalCRMWSDDServiceName : String = "InternalCRM"
-ports : HashSet = null
<<constructor>>+InternalCRMServiceLocator()
<<constructor>>+InternalCRMServiceLocator(config : EngineConfiguration)
<<constructor>>+InternalCRMServiceLocator(wsdlLoc : String, sName : QName)
<<getter>>+getInternalCRMAddress() : String
<<getter>>+getInternalCRMWSDDServiceName() : String
<<setter>>+setInternalCRMWSDDServiceName(name : String) : void
<<getter>>+getInternalCRM() : InternalCRM
<<getter>>+getInternalCRM(portAddress : URL) : InternalCRM
<<setter>>+setInternalCRMEndpointAddress(address : String) : void
<<getter>>+getPort(serviceEndpointInterface : Class) : Remote
<<getter>>+getPort(portName : QName, serviceEndpointInterface : Class) : Remote
<<getter>>+getServiceName() : QName
<<getter>>+getPorts() : Iterator
<<setter>>+setEndpointAddress(portName : String, address : String) : void
<<setter>>+setEndpointAddress(portName : QName, address : String) : void
InternalCRMSoapBindingStub
-cachedSerClasses : Vector = new java.util.Vector()
-cachedSerQNames : Vector = new java.util.Vector()
-cachedSerFactories : Vector = new java.util.Vector()
-cachedDeserFactories : Vector = new java.util.Vector()
~ operations : OperationDesc"[]" [0..*]
- initOperationDesc1() : void
<<constructor>>+InternalCRMSoapBindingStub()
<<constructor>>+InternalCRMSoapBindingStub(endpointURL : URL, service : Service)
<<constructor>>+InternalCRMSoapBindingStub(service : Service)
#createCall() : Call
+findLastLeads(in0 : Calendar, in1 : Calendar) : LeadWTO"[]"
+findLeads(in0 : double, in1 : double, in2 : String) : LeadWTO"[]"
LeadWTO
-annualRevenue : double
-city : String
-company : String
-country : String
-insertionDate : Calendar
-name : String
-phone : String
-postalCode : String
-state : String
-street : String
-__equalsCalc : Object = null
-__hashCodeCalc : boolean = false
-typeDesc : TypeDesc = new org.apache.axis.description.TypeDesc(LeadWTO.class,true)
<<constructor>>+LeadWTO()
<<constructor>>+LeadWTO(annualRevenue : double, city : String, company : String, country : String, insertionDate : Calendar, name : String, phone : String, postalCode : String, state : String, street : String)
<<getter>>+getAnnualRevenue() : double
<<setter>>+setAnnualRevenue(annualRevenue : double) : void
<<getter>>+getCity() : String
<<setter>>+setCity(city : String) : void
<<getter>>+getCompany() : String
<<setter>>+setCompany(company : String) : void
<<getter>>+getCountry() : String
<<setter>>+setCountry(country : String) : void
<<getter>>+getInsertionDate() : Calendar
<<setter>>+setInsertionDate(insertionDate : Calendar) : void
<<getter>>+getName() : String
<<setter>>+setName(name : String) : void
<<getter>>+getPhone() : String
<<setter>>+setPhone(phone : String) : void
<<getter>>+getPostalCode() : String
<<setter>>+setPostalCode(postalCode : String) : void
<<getter>>+getState() : String
<<setter>>+setState(state : String) : void
<<getter>>+getStreet() : String
<<setter>>+setStreet(street : String) : void
+equals(obj : Object) : boolean{guarded}
+hashCode() : int{guarded}
<<getter>>+getTypeDesc() : TypeDesc
<<getter>>+getSerializer(mechType : String, javaType : Class, xmlType : QName) : Serializer
<<getter>>+getDeserializer(mechType : String, javaType : Class, xmlType : QName) : Deserializer
InternalCRMSoapBindingImpl
+findLastLeads(startDate : Calendar, endDate : Calendar) : LeadWTO"[]"
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : LeadWTO"[]"
InternalCRMWSDLTypeConversor
+fromWSDL(leadWTO : LeadWTO) : InternalCRMLeadTO
+fromWSDL(leadWTOs : LeadWTO"[]") : List<E->InternalCRMLeadTO>
+toWSDL(internalCRMLeadTO : InternalCRMLeadTO) : LeadWTO
+toWSDL(internalCRMLeadTOs : List<E->InternalCRMLeadTO>) : LeadWTO"[]"
<<constructor>>-InternalCRMWSDLTypeConversor()


package Data[InternalCRM2]
InternalCRMFacadeDelegateFactory
-DELEGATE_CLASS_NAME_PARAMETER : String = "InternalCRMFacadeDelegateFactory/delegateClassName"{readOnly}
-delegateClass : Class
<<getter>>-getDelegateClass() : Class
<<getter>>+getInternalCRMFacadeDelegate() : InternalCRMFacadeDelegate
<<constructor>>-InternalCRMFacadeDelegateFactory()
InternalCRMFacadeDelegate
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->InternalCRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->InternalCRMLeadTO>
PlainInternalCRMFacadeDelegate
-internalCRMLeadTOs : List<E->InternalCRMLeadTO>
+findLastLeads(startDate : Calendar, endDate : Calendar) : List<E->InternalCRMLeadTO>
+findLeads(minAnnualRevenue : double, maxAnnualRevenue : double, state : String) : List<E->InternalCRMLeadTO>
InternalCRMLeadTO
-name : String
-company : String
-annualRevenue : double
-phone : String
-street : String
-postalCode : String
-city : String
-state : String
-country : String
-insertionDate : Calendar
<<constructor>>+InternalCRMLeadTO()
<<constructor>>+InternalCRMLeadTO(name : String, company : String, annualRevenue : double, phone : String, street : String, postalCode : String, city : String, state : String, country : String, insertionDate : Calendar)
<<JavaElement>>+equals(obj : Object) : boolean{JavaAnnotations = "@Override"}
<<getter>>+getAnnualRevenue() : double
<<getter>>+getCity() : String
<<getter>>+getCompany() : String
<<getter>>+getCountry() : String
<<getter>>+getInsertionDate() : Calendar
<<getter>>+getName() : String
<<getter>>+getPhone() : String
<<getter>>+getPostalCode() : String
<<getter>>+getState() : String
<<getter>>+getStreet() : String
<<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}
InternalCRMTypeConversor
+fromInternalCRM(internalCRMLeadTO : InternalCRMLeadTO) : CRMLeadTO
+fromInternalCRM(internalCRMLeadTOs : List<E->InternalCRMLeadTO>) : List<E->CRMLeadTO>
+toInternalCRM(CRMLeadTO : CRMLeadTO) : InternalCRMLeadTO
+toInternalCRM(CRMLeadTOs : List<E->CRMLeadTO>) : List<E->InternalCRMLeadTO>
<<constructor>>-InternalCRMTypeConversor()

3. Compilación e instalación de la aplicación

Las intrucciones para la compilación y correcta instalación de la aplicación se detallan en el fichero *README* incluido en la distribución. El contenido de este fichero se muestra a continuación:

```
********** Mashup v1.1 **********

                Jesus Angel Perez-Roca Fernandez
                    djalma_fd@yahoo.es
                          June 2007

Contents
--------

1. Software requirements
2. Building from the source code
3. Execution on Tomcat
4. Execution from the GWT Shell

1. Software requirements
   ---------------------

   * An implementation of J2SE 5.0 or superior.

     I have used Sun's JDK 1.6.0.

   * ant 1.6.5.

     To manage the project.

   * A Java EE compliant Web application server.

     I have used Jakarta Tomcat 6.0 and tested it succesfully on Jakarta Tomcat
5.5.17 too.

   * WS-JavaExamples 1.0's Util library.

   * Google Web Toolkit (GWT) 1.2.22.

   * Google Maps GWT 1.5.2 (http://sourceforge.net/projects/gwt).

   Probably it should work with higher versions of the above packages (or with
   minor changes).

2. Building from the source code
   -----------------------------

   Extract zip file (if you are running Windows) or tar.gz (if you are running
Linux)

   Before compiling, you may need/want to adapt the following files:

   + Subsystems/CommonProperties.xml. Go to "Development environment" section
     and adapt the paths specified in "xxx.home" properties.

   + PropertiesConfiguration/ConfigurationParameters.properties. The class
     implementing the stateful VirtualCRMService interface is specified in
     "VirtualCRMServiceFactory/className" property. This distribution includes a
     mock implementation to test three typical cases (returning results,
     throwing a runtime exception, and returning an empty list of results)
     in cyclic order. The class implementing the GeolocationService interface is
     specified in "GeolocationServiceFactory/className" property. The classes
     implementing the CRMService interface are specified in
     "GeolocationServiceFactory/className" property, adding a number who
```

```
indicate
     its sequence number (e.g. GeolocationServiceFactory/className1,
     GeolocationServiceFactory/className2, GeolocationServiceFactory/className3,
     and so on). When removing one of this properties, make sure there is no gap
     in the sequence numbers (e.g. if there are many GeolocationService
     implementations and you remove the second one, you have to rename the
     property GeolocationServiceFactory/className3 to
GeolocationServiceFactory/className2,
     GeolocationServiceFactory/className4 to
GeolocationServiceFactory/className3, and so on,
     or you can simply change the last of the implementations and set
     GeolocationServiceFactory/className2 as its name).

   + Subsystems/UI/Sources/es/udc/mashup/ui/public/TopPanel.html. It includes a
     valid Google Maps key for http://localhost:8080 in the Google Maps
     JavaScript. Update if necessary.

   cd Subsystems
   ant all

3. Execution on Tomcat
   -------------------

   Start Tomcat

   cd Subsystems/InternalCRM/Sources
   ant deployws

4. Execution from the GWT Shell
   ----------------------------

   ***This is only necessary if developing the client-side code (GWT Java code
      to be translated to JavaScript).***

   Subsystems/UI/Sources/es/udc/mashup/ui/public/TopPanel.html includes a
   commented Google Maps JavaScript with a valid key for http://localhost:8888.
   Comment the previous script and uncommnet this.

4.1. ant task.

   Execute "rungwtshell" ant task in Subsystems/UI/Sources/build.xml

4.2. Debugging the client-side code from an IDE.

   For convenience, I have included MashupUITopPanel.launch in the root
   directory of the source distribution. If you use Eclipse and name the
   project "Mashup", you will automatically see the "MashupUITopPanel"
   application. You can use it for debugging the client-side code.
   "MashupUITopPanel" executes the same command as the "rungwtshell" ant task
   (not the ant task, but the same as the ant task). Do the same for other IDEs.
```

4. Problemas conocidos

No se ha observado ningún problema referente a la aplicación durante las diversas pruebas, lo cual no quiere decir que no exista ningún problema.