TRABAJO DE TITULACIÓN:

“AGENTES INTELIGENTES PARA LA GESTIÓN DE RECURSOS DIGITALES DE APRENDIZAJE”

**MANUAL DEL PROGRAMADOR**

**smaweb**

**Versión 1.0**

|  |  |
| --- | --- |
| **DATOS GENERALES** | |
| **Código:** | 003 |
| **Versión:** | 1.0 |
| **Fecha de la versión:** | 26 de septiembre de 2019 |
| **Páginas:** |  |
| **Creado por:** | Alex Condoy |
| **Nivel de confidencialidad:** | Alto |

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | |
| **Código** | **Versión** | **Fecha** | **Responsable** |
| 003 | 1.0 | 26 de septiembre de 2019 | Alex Rubén Condoy Carrión |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CONTROL DE MODIFICACIONES** | | | | |
| **Código** | **Versión** | **Fecha** | **Responsable** | **Descripción** |
| 003 | 1.0 | 26 de septiembre de 2019 | Alex Rubén Condoy Carrion | Creación |
|  |  |  |  |  |
|  |  |  |  |  |

**TABLA DE CONTENIDO**

[1. INTRODUCCIÓN 5](#_Toc20391854)

[2. DIRIGIDO A USUARIOS 5](#_Toc20391855)

[3. OBJETIVO 5](#_Toc20391856)

[4. REQUERIMIENTOS PREVIOS. 5](#_Toc20391857)

[5. CONEXIÓN CON LA BASE DE DATOS 6](#_Toc20391858)

[6. CONEXIÓN CON OPENLINK VIRTUOSO 9](#_Toc20391859)

[7. INSTALACIÓN DEL SMA EN EL SERVIDOR WEB 14](#_Toc20391860)

[8. INGRESAR 16](#_Toc20391861)

**ÍNDICE DE FIGURAS**

[Figura 1. Arquitectura de Instalación. 5](#_Toc20391871)

[Figura 2. Creación de la BD. 6](#_Toc20391872)

[Figura 3. Script de BD. 7](#_Toc20391873)

[Figura 4. Tablas admin y usuarios. 8](#_Toc20391874)

[Figura 5. Creación del usuario para el SMA. 8](#_Toc20391875)

[Figura 6. Permisos de usuario de BD. 8](#_Toc20391876)

[Figura 7. Driver MySql en NetBeans. 9](#_Toc20391877)

[Figura 8. Conexión del proyecto con la BD. 9](#_Toc20391878)

[Figura 9. Instalación de OpenLink Virtuoso (Windows). 10](#_Toc20391879)

[Figura 10. OpenLink Virtuoso Server [vos] 10](#_Toc20391880)

[Figura 11. Panel de Administración de Virtuoso. 11](#_Toc20391881)

[Figura 12. Virtuoso Conductor. 11](#_Toc20391882)

[Figura 13. Linked Data. 11](#_Toc20391883)

[Figura 14. Creación de la Tabla (Graph) LearningObjects. 12](#_Toc20391884)

[Figura 15. Creación de la Tabla (Graph) Perfiles. 12](#_Toc20391885)

[Figura 16. Graphs de LeanringObjects y Perfiles creadas. 12](#_Toc20391886)

[Figura 17. Virtuoso JDBC 4 Driver JAR file. 13](#_Toc20391887)

[Figura 18. Dependencias de Virtuoso. 13](#_Toc20391888)

[Figura 19. Código de conexión con el servidor Virtuoso. 13](#_Toc20391889)

[Figura 20. Reglas de firewall para acceso de Virtuoso. 14](#_Toc20391890)

[Figura 21. Seleccionar sma\_web como el proyecto principal. 14](#_Toc20391891)

[Figura 22. Mensaje de confirmación al generar el archivo WAR. 15](#_Toc20391892)

[Figura 23. Archivo sma\_web.war. 15](#_Toc20391893)

[Figura 24. Descompresión del archivo WAR en webapps/ROOT. 15](#_Toc20391894)

[Figura 25. Despliegue del sistema en el servidor. 16](#_Toc20391895)

# INTRODUCCIÓN

El presente manual explica paso a paso y detalladamente la estructura y programación del proyecto, definiendo los paquetes, clases e interfaces más importantes, se exponen la estructura del proyecto, los paquetes principales, seguidos de las librerías importadas, los archivos JSP, CSS, JavaScript entre otros.

# DIRIGIDO A USUARIOS

El presente manual está destinado al personal técnico y/o investigadores que pretendan revisar la programación del Sistema Multiagente (SMA) desarrollado en el trabajo de titulación: “Agentes Inteligentes para la Gestión de Recursos Digitales de Aprendizaje”.

# OBJETIVO

Detallar la estructura y programación de sistema de gestión de objetos de aprendizaje **SmaWeb** (<http://smaweb.azurewebsites.net/>), el mismo que permite realizar búsquedas de objetos de aprendizaje en base al perfil inteligente de un estudiante.

# ESTRUCTURA DEL PROYECTO

El Proyecto, nombrado **sma\_web,** está estructurado como lo indica la Figura 1; compuesto por cinco carpetas principales:

* La carpeta **Web Pages** que contiene los archivos JSP, HTML, JavaScript y CSS, los mismos que forman la interfaz de usuario.
* La carpeta **Remote Files**, que contiene algunos archivos externos como Bootstrap y JQuery.
* La carpeta **Source Packages**, contiene los paquetes principales del proyecto como el sistema multiagente, los servlets, el paquete modelo y el paquete controlador.
* La carpeta de **Libraries** contiene todas las librerías externas importadas.
* Y finalmente la carpeta **Configuration Files** que contiene los archivos de configuración.

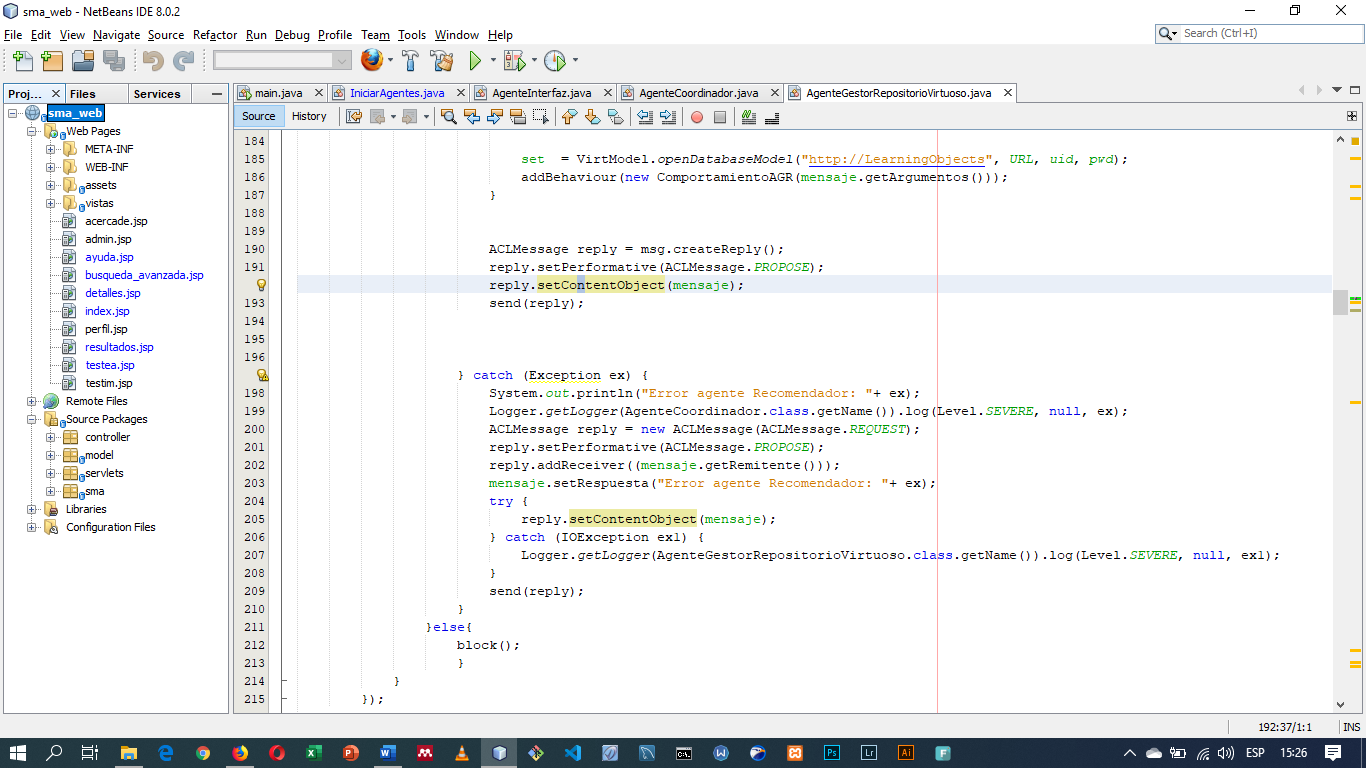


Figura 1. Estructura del Proyecto.

# WEB PAGES

El directorio Web Pages contiene los Archivos JSP principales, una carpeta de vistas que generalmente son reutilizadas en los archivos principales, la carpeta de assets que contiene los archivos CSS, JS, imágenes y dos carpetas de configuración de las vistas.

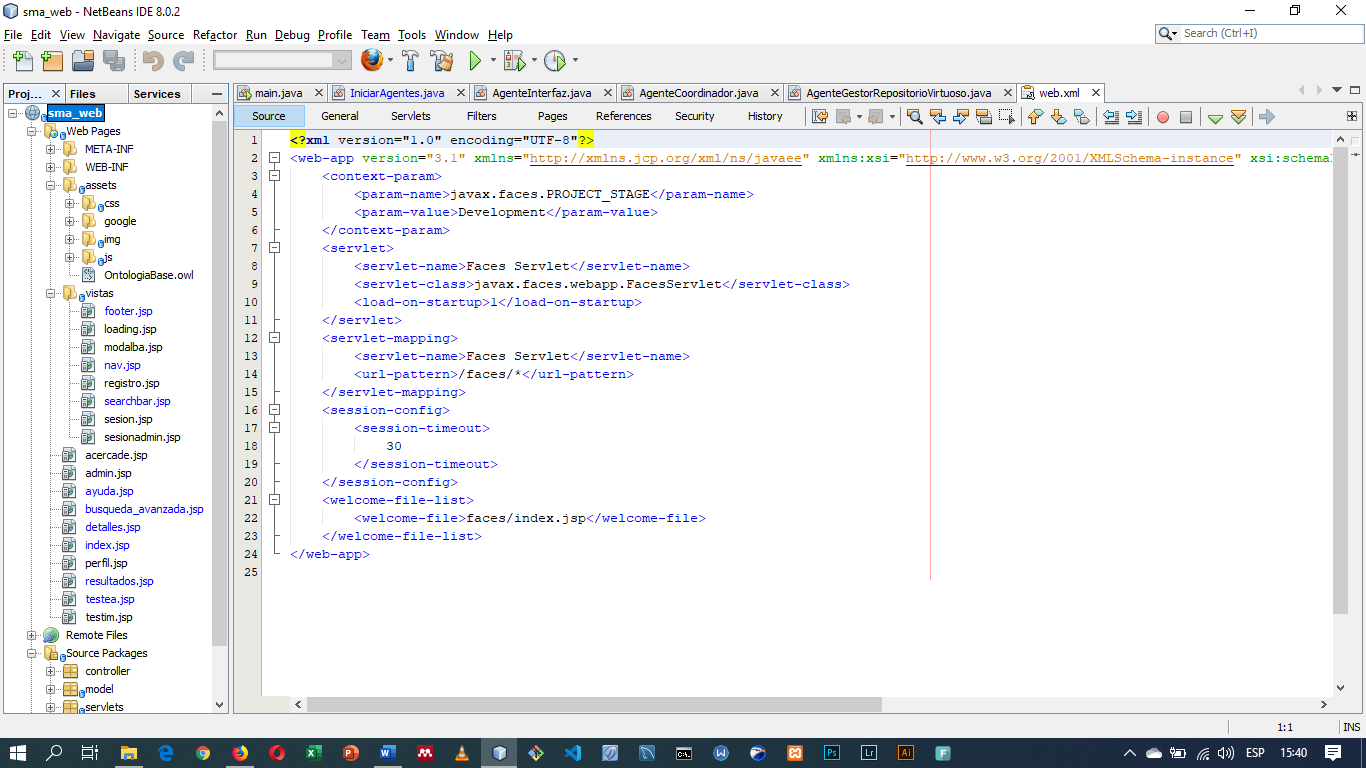


Figura 2. Estructura de Web Pages.

Los archivos ubicados en el fichero Web Pages, constituyen las paginas conforman el sistema, están conformadas por la pagina index, acerca de, ayuda, perfil, testim, testea, nusqueda avanzada, las paginas de resultados y una pagina de administrador.

## META-INF y WEB-INF

Contiene los archivos de configuración del proyecto, el archivo Web.xml permite definir la pagina de inicio del proyecto además de la configuración de servlets.

**WEB-INF >Web.xml**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd">      <context-param>          <param-name>javax.faces.PROJECT\_STAGE</param-name>          <param-value>Development</param-value>      </context-param>      <servlet>          <servlet-name>Faces Servlet</servlet-name>          <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>          <load-on-startup>1</load-on-startup>      </servlet>      <servlet-mapping>          <servlet-name>Faces Servlet</servlet-name>          <url-pattern>/faces/\*</url-pattern>      </servlet-mapping>      <session-config>          <session-timeout>              30          </session-timeout>      </session-config>      <welcome-file-list>          <welcome-file>faces/index.jsp</welcome-file>      </welcome-file-list>  </web-app> |

Figura 3. Web.xml

**META-INF > context.xml**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <Context antiJARLocking="true" path=""/> |

Figura 4. context.xml

## ASSETS

La carpeta CSS contiene los estilos para cada archivo JSP; además contiene las imágenes que muestra el sistema y los archivos JavaScript.

## VISTAS

Este directorio contiene vistas que son reutilizadas en los archivos JSP, el archivo footer.jsp contiene el código del pie de página, el archivo loading contiene un mensaje de espera, modalba.jsp contiene un mensaje modal, nav.jsp muestra la barra de navegación, registro.jsp contiene una ventana modal de registro, searchbar.jsp contiene una barra de búsqueda utilizada en la página de resultados y de detalle, sesión.jsp y sesionadmin.jsp contiene ventanas para el inicio de sesión del usuario y administrador.

# REMOTE FILES

Contiene las librerías importadas mediante dirección HTTP; se muestran las librerías Bootstrap y sus dependencias; y la librería JQuery; Las cuales son incluidas en los archivos JSP.

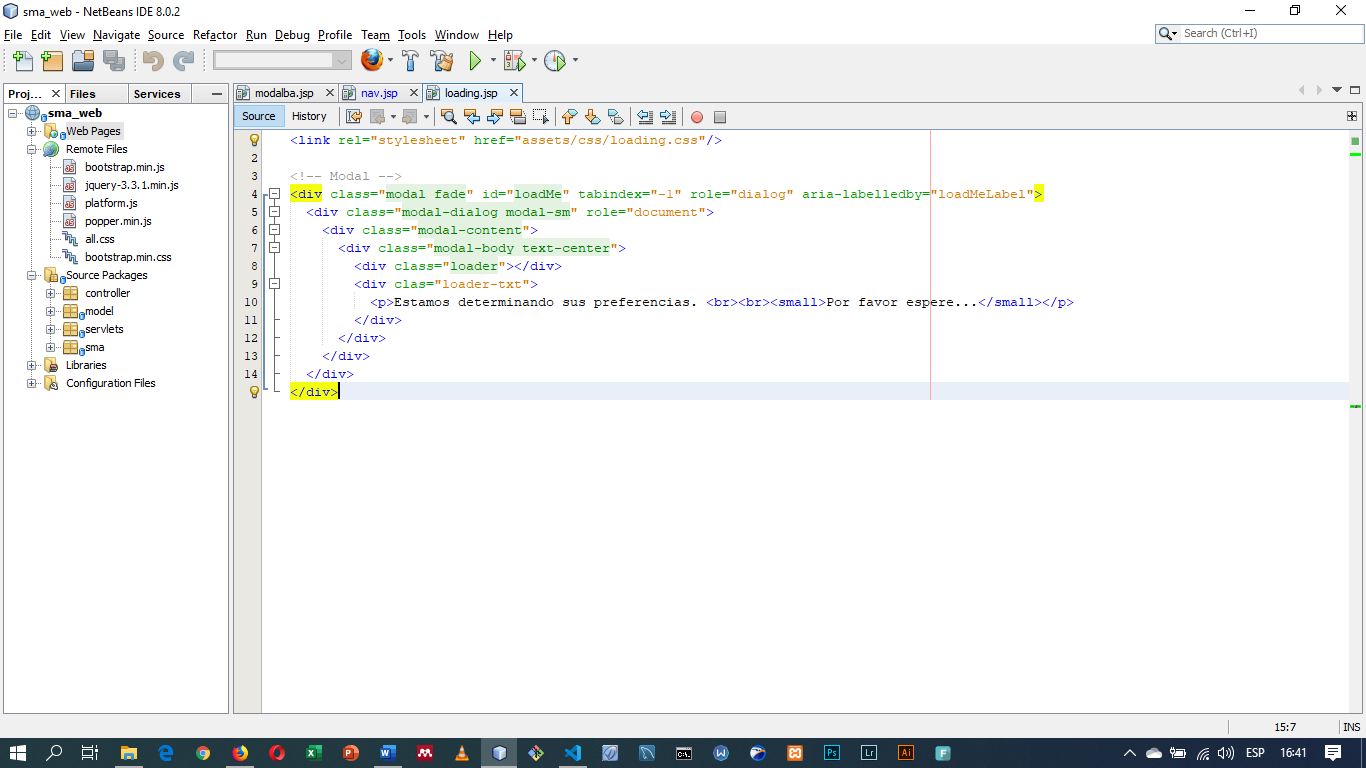


Figura 5. librerías externas importadas por HTTP.

# SOURCE PACKAGES

Este fichero contiene los paquetes principales del proyecto; el cual ha sido dividido en 4, Si bien se ha tratado de trabajar con un paradigma Modelo-Vista-Controlador (MVC), no se lo ha cumplido totalmente, debido a la necesidad de integrar el SMA.

Las Vistas ya han sido expuestas en el fichero Web Pages; en source packages se encuentra el paquete model y controller, además el paquete servlets es considerado como modelo, pero está separado por su función única de contenedor de servlets.

El fichero también contiene el SMA, programado dentro del paquete sma, como lo muestra la Figura 6.

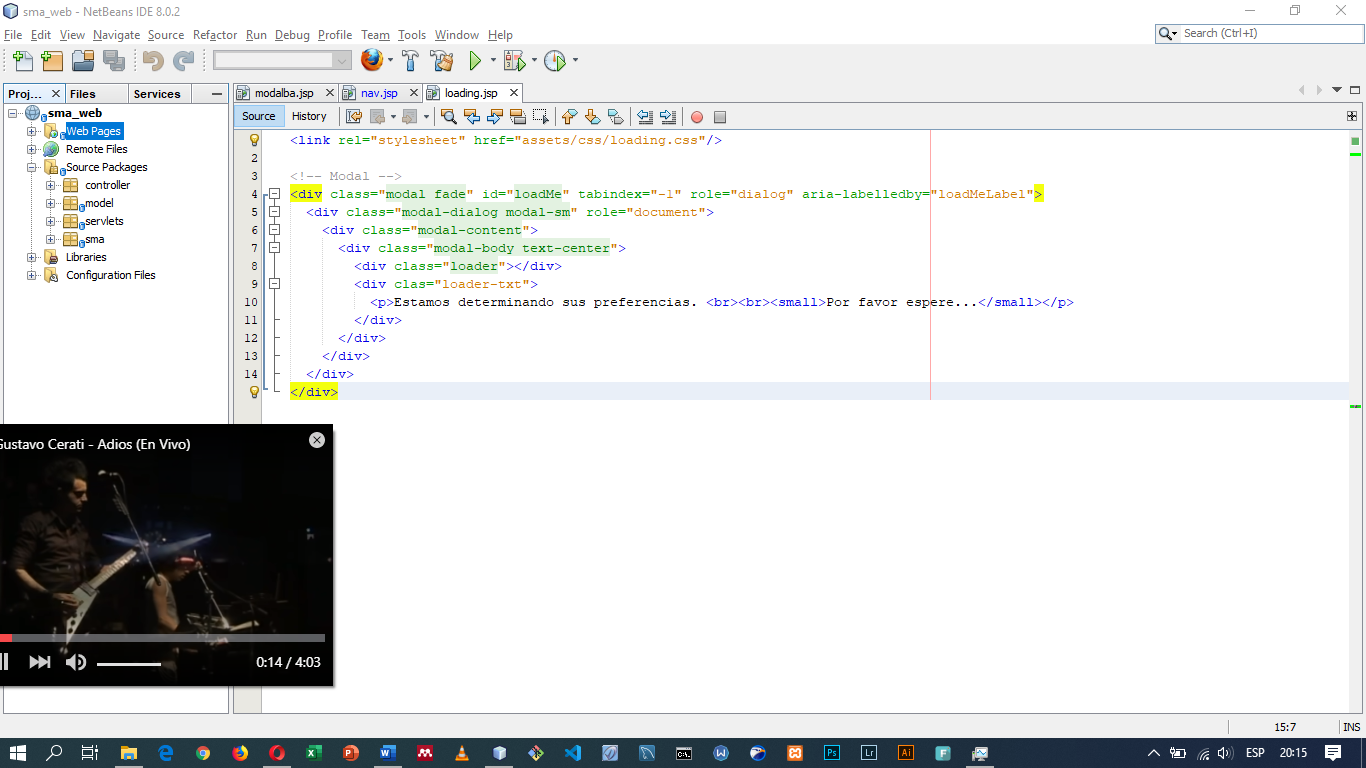


Figura 6. Source Packages

## CONTROLLER

Dentro de este paquete únicamente se programaron dos clases que sirven de controladores; el resto se detallan en el paquete servlets.

**Acentos.java**

Clase que se encarga de transformar cadenas con acentos, en cadenas que no los contengan; para realizar las búsquedas, tanto con y sin acentos de una misma palabra.

|  |
| --- |
| package controller;  public class Acentos {     public static String eliminarAcentos(String ky){         String cadena =ky;         cadena = cadena.replace('á','a');         cadena = cadena.replace('é','e');         cadena = cadena.replace('í','i');         cadena = cadena.replace('ó','o');         cadena = cadena.replace('ú','u');          cadena = cadena.replace('Á','A');         cadena = cadena.replace('É','E');         cadena = cadena.replace('Í','I');         cadena = cadena.replace('Ó','O');         cadena = cadena.replace('Ú','U');         return cadena;     } } |

Figura 7. Acentos.java

**Mensaje.java**

Es una de las clases más importantes del controlador. Constituye el modelo de comunicación de los agentes inteligentes, esta clase contiene los métodos necesarios para que los Agentes inteligentes del SMA intercambien información y las interfaces presenten los mensajes de respuesta.

|  |
| --- |
| package controller;  import jade.core.AID;  public class Mensaje implements java.io.Serializable{      private String mensaje = new String("");      private Object obj;      private Object respuesta;      private String user;      private AID remitente;      public void setMensaje (String mensaje){          this.mensaje = mensaje;      }      public String getMensaje (){          return mensaje;      }      public void setArgumentos(java.lang.Object obj){          this.obj = obj;      }      public Object getArgumentos(){          return obj;      }       public void setRespuesta(java.lang.Object resp){          this.respuesta = resp;      }      public Object getRespuesta(){          return respuesta;      }      public void setUsuario(String user){          this.user = user;      }      public String getUsuario(){          return user;      }      public void setRemitente(AID r){          this.remitente=r;      }      public AID getRemitente(){          return remitente;      }  } |

Figura 8. mensaje.java

## MODEL

El modelo contiene los métodos que interactúan directamente con la base de datos MySql y OpenLink Virtuoso, además contiene uno de los métodos que interactúa con los servicios de Google para el inicio de sesión (oauth2).

**IdTokenVerifierAndParser.java**

Servicio de Google que permite realizar autenticación con Gmail.

|  |
| --- |
| package model;  import com.google.api.client.googleapis.auth.oauth2.GoogleIdToken;  import com.google.api.client.googleapis.auth.oauth2.GoogleIdTokenVerifier;  import com.google.api.client.http.javanet.NetHttpTransport;  import com.google.api.client.json.jackson.JacksonFactory;  public class IdTokenVerifierAndParser {      private static final String GOOGLE\_CLIENT\_ID = "clienttoken.apps.googleusercontent.com";      public static GoogleIdToken.Payload getPayload (String tokenString) throws Exception {          JacksonFactory jacksonFactory = new JacksonFactory();          GoogleIdTokenVerifier googleIdTokenVerifier =                              new GoogleIdTokenVerifier(new NetHttpTransport(), jacksonFactory);          GoogleIdToken token = GoogleIdToken.parse(jacksonFactory, tokenString);          if (googleIdTokenVerifier.verify(token)) {              GoogleIdToken.Payload payload = token.getPayload();              if (!GOOGLE\_CLIENT\_ID.equals(payload.getAudience())) {                  throw new IllegalArgumentException("Audience mismatch");              } else if (!GOOGLE\_CLIENT\_ID.equals(payload.getAuthorizedParty())) {                  throw new IllegalArgumentException("Client ID mismatch");              }              return payload;          } else {              throw new IllegalArgumentException("id token cannot be verified");          }      }  } |

Figura 9. IdTokenVerifierAndParser.java

**Virtuoso.java**

Clase que interactúa directamente con la instancia de Virtuoso en la maquina virtual, se encarga de insertar, modificar, y crear individuos en la ontología.

|  |
| --- |
| private static final String URL = "jdbc:virtuoso://104.210.144.119:1111";      private static final String uid = "dba";      private static final String pwd = "dba";      VirtModel model=null;      String bd ;      public void conectar(String bd){           model = VirtModel.openDatabaseModel(bd, URL, uid, pwd);           this.bd=bd;      }      public void desconectar(){           model.close();      }      public boolean SetEstilos(int [] array,String user){          . . .      }  . . . . |

Figura 10. Extracto Virtuoso.java

## SERVLETS

Este fichero es la segunda parte de Controller, contiene toda la lógica que comunica las interfaces con el modelo y con el SMA.

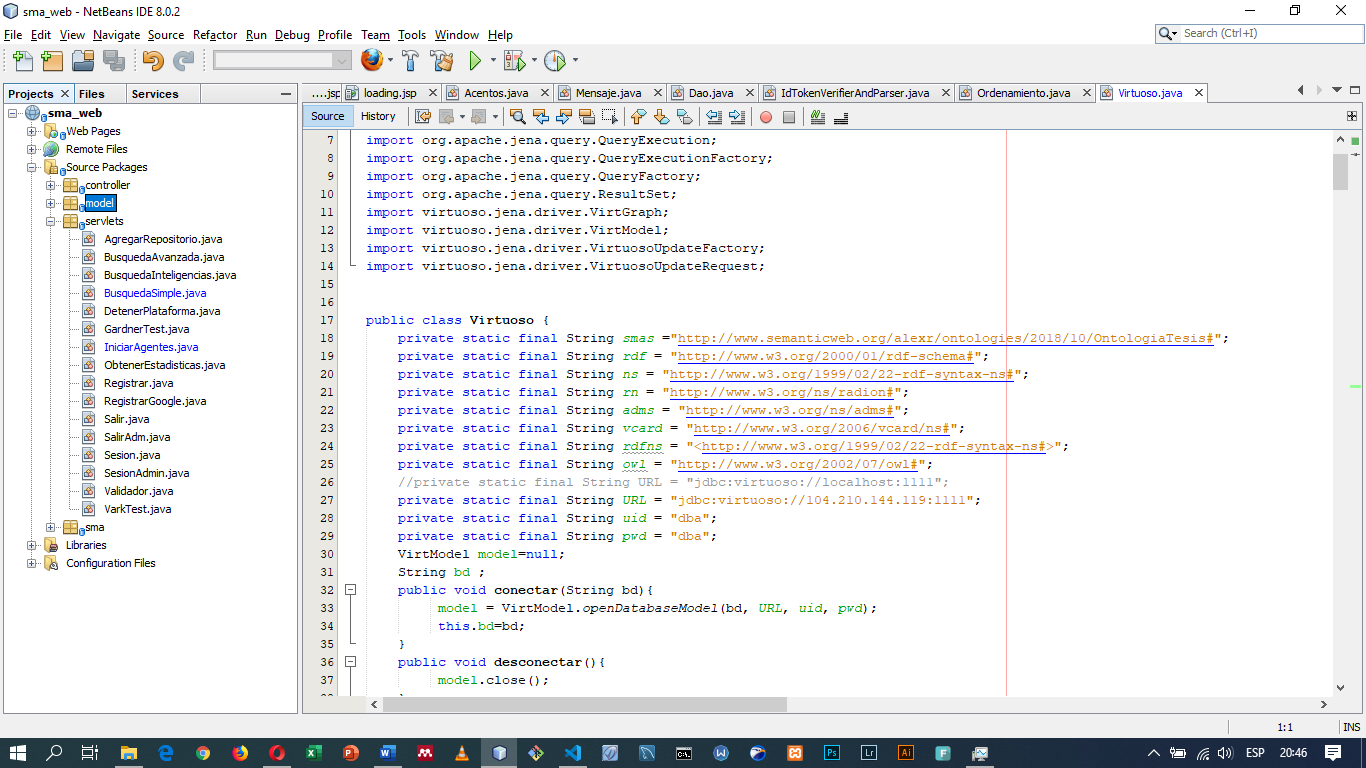


Figura 11. Servlets del Sistema.

Para cada formulario existe un servlet que procesa su petición. Se tiene servlets para búsqueda, Sesión, Registro, y formularios en general.

## SMA

Este paquete contiene el sistema multiagente, conformado por 5 agentes inteligentes como lo muestra la Figura 12.

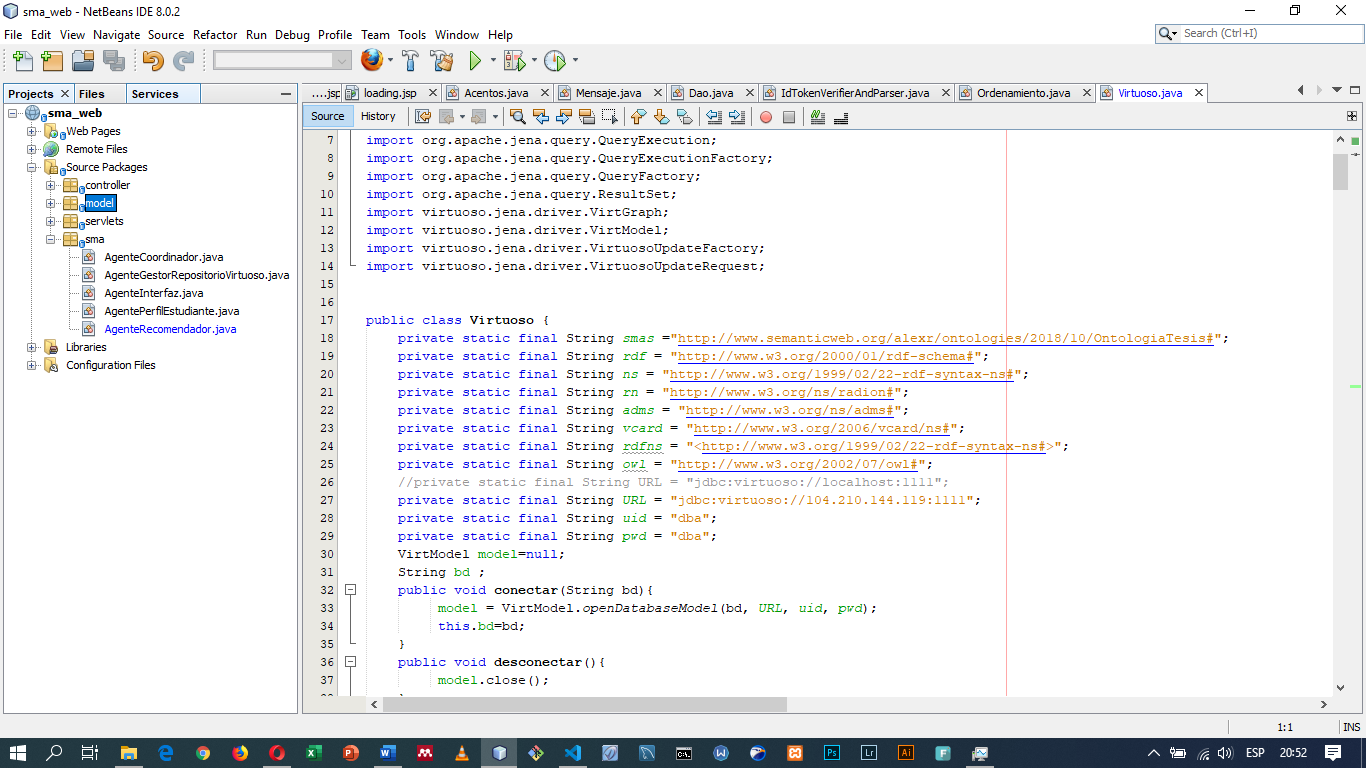


Figura 12. Agentes del SMA.

La estructura de los agentes es bastante parecida y su funcionamiento esta explicado en la memoria del trabajo de titulación.

A continuación, se muestra la programación de uno de los agentes y se presente una breve explicación.

**AgenteCoordinador.java**

El agente coordinador es el encargado de inicializar tres de los agentes inteligentes, para lo cual hereda de la clase **Agent** de JADE, con lo cual debe implementar el método **setup** que se ejecuta una vez que se crea el agente, este método recibe un mensaje de inicialización (**IA**) y procede con la creación e inicialización de cada agente. Una vez el agente termina su ejecución, llama al método **TakeDown** que finaliza el agente.

|  |
| --- |
| package sma;  import jade.core.Agent;  import jade.core.behaviours.OneShotBehaviour;  import jade.wrapper.AgentContainer;  import jade.wrapper.AgentController;  public class AgenteCoordinador extends Agent {      @Override      protected void setup(){           System.out.println("Hola"+getAID().getName()+"soy coordinador");            Object [] args = getArguments();          if(args != null && args.length==1){              addBehaviour( new OneShotBehaviour() {                  @Override                  public void action() {                      if(args[0].equals("IA")){                          try{                               AgentContainer c = getContainerController();                                  AgentController controlador;                                  controlador = c.createNewAgent("AgenteEstudiante",AgentePerfilEstudiante.class.getName(),null);                                  controlador.start();                                  controlador=c.createNewAgent("AgenteRecomendador",AgenteRecomendador.class.getName(),null);                                  controlador.start();                                  controlador=c.createNewAgent("AGR",AgenteGestorRepositorioVirtuoso.class.getName(),null);                                  controlador.start();                            }catch(Exception e){                              System.out.println("No se pudo");                          }                      }                  }              });          }      }        @Override      protected void takeDown() {          System.out.println("Finaliza el "                  + this.getLocalName());          System.out.println("---------");          this.doDelete();      }  } |

Figura 13. AgenteCoordinador.java

# LIBRARIES

El proyecto incluye una serie de librerias para su funcionamiento; seguidamente se describe su funcionamiento y características.

## JADE

JADE es un framework para el desarrollo de agente inteligentes; en el presente proyecto se usó la versión 4.5.0.

## JENA

Apache Jena es un framework libre para construir aplicaciones basadas en la Web semántica, en el smaweb se utilizo para realizar todo el manejo de los datos de la ontología. Su versión utilizada fue la 3.9.0

## JAVA-JSON

Es una librería que permitió manejar un archivo JSON descargado los repositorios de objetos de aprendizaje, esta librería es utilizada por el agente gestor de repositorios. Su versión es la 1.6.0.

## MySQL JDBC Driver

Es un driver que permite conectar JAVA con MySQL, esto lo realiza el modelo para la conexión y manipulación de la base de datos. La versión utilizada es la 5.1.23.

## OAUTH2

Es un marco de autorización utilizado por la API de Google que permite que una aplicación de terceros pueda utilizar los servicios de autenticación de Google, la versión usada es: 2.0.

## GOOGLE API

Se utilizó la API de autenticación de Google, tanto para el registro de usuarios como para el inicio de sesión de los mismos.

## VIRT-JENA

Es un proveedor de almacenamiento de modelos de gráficos nativos totalmente operativo para Jena Framework, que permite que las aplicaciones web semánticas escritas utilizando los marcos Jena RDF para consultar directamente la tienda Virtuoso RDF Quad Store. La versión utilizada es la 3.0.

## VIRT-JDBC4

Es una API para la ejecución de sentencias SPARQL, esto lo realiza el modelo para la conexión y manipulación de la base de datos Virtuoso. La versión utilizada es la 4.0.

## JDK

La versión del Java Development Kit usada, es la 1.8