Sistema multiagentes para búsqueda de recursos de Datos Abiertos enlazados indexados

Camilo Alejandro Valencia Martínez
kmilov@gmail.com
3185853811

Noviembre 2017

Ingeniería de Software I, Maestría en Ciencias de la Información y las Comunicaciones, Universidad Distrital Francisco José de Caldas

Tabla de contenido

1 Descripción de la necesidad **2**
1.1 Antecedentes . . . 2
1.2 Necesidad . . . 3

2 Alcance **3**

3 Análisis de requerimientos **3**
3.1 Requerimientos Funcionales . . . 3
3.2 Requerimientos no funcionales . . . 4

4 Metodología **4**

5 Aplicación de la metodología **5**
5.1 Planeación . . . 5
5.2 Análisis . . . 7
5.2.1 Casos de uso . . . 7
5.2.2 Identificación de agentes . . . 7
5.2.3 Identificación de responsabilidades . . . 8
5.2.4 Identificación de conocidos . . . 8
5.2.5 Refinamiento de agentes . . . 9
5.2.6 Información de despliegue de los agentes . . . 11
5.3 Diseño . . . 11
5.3.1 División/Unión/Cambio nombre agentes . . . 11
5.3.2 Especificación de la interacción . . . 11
5.3.3 Definición de protocolos de interacción Ad hoc . . . 11
5.3.4 Definición de protocolos de interacción Ad hoc . . . 11

5.3.5 Plantillas de mensajes 12
5.3.6 Descripción a ser registrada/consultada en páginas amarillas. 13
5.3.7 Interacción Agentes – Recursos 13
5.3.8 Interacción Agentes – Usuarios 14
5.3.9 Comportamientos internos agentes 14
5.3.10 Definiendo una ontología 15
5.4 Diseño integración con servicio web 15
5.4.1 Diseño de base de datos 15

6 Implementación **16**
6.1 Restaurar base de datos 16
6.2 Compilar librerías agentes 16
6.3 Iniciar contenedor de agentes 17
6.4 Desplegar Agente clasificador 18
6.5 Ejecución del servicio Web 18
6.6 Prueba de servicios web 19
6.7 Cliente REST 19
6.7.1 Uso 20
6.7.2 Servicio No 1: Obtener términos 20
6.7.3 Servicio No 2: Obtener recursos 21
6.8 Pruebas desde la interfaz 22
6.9 Mensajería entre agentes 22

7 Despliegue **23**
7.1 Infraestructura: 23
7.2 Creación instancia base de datos 23
7.3 Creación instancia y conexión 24
7.4 Instalación de software y plataformas necesarias 27

8 Resultados **31**

1 Descripción de la necesidad

1.1 Antecedentes

Dentro de las iniciativas Open Data, y Open Linked Data se encuentran diferentes problemáticas de acceso y procesamiento de la información contenida en repositorios abiertos. Entre ellos podemos encontrar problemas de disponibilidad, heterogeneidad de formatos, calidad de la información presentada, consulta de información innecesaria como metadatos de un elemento, insuficientes mecanismos de consulta y presentación de la información. Para dar solución a estos problemas hay herramientas diseñadas para diferentes repositorios como CKAN o Datahub que evalúan los datasets. También existen modelos semánticos para la evaluación en diferentes plataformas como en [2], donde se define un modelo para evaluar recursos de aprendizaje sobre plataformas LCMS (Learning

Content Management System) a través de un modelo semántico y el uso de razonadores. Estas propuestas presentan algunas falencias, como la incapacidad de procesar datos con formatos diferentes para los que fueron diseñadas, la presentación de la información a veces no es clara, no fácil para usuarios que no tengan conocimientos en computación, entre otras. Por estos motivos se hace necesario seguir investigando y desarrollando técnicas que permitan el acceso a esta información a cualquier usuario y garantice un alto nivel de calidad de esta información. En el desarrollo de la web semántica ya se han adelantado mecanismos de definición de niveles de confianza como web of trust (WOT), que trabaja sobre infraestructura de clave privada (PKI) y es un referente para definir estos niveles sobre contenidos abiertos.

1.2 Necesidad

Se necesita un software que facilite la búsqueda e identificación de recursos abiertos bajo principios de niveles de confianza sobre Datos Abiertos Enlazados. El sistema debe contemplar desde la indexación y clasificación, hasta la búsqueda y consulta de recursos por parte del usuario.

2 Alcance

Por la complejidad del proyecto y teniendo en cuenta que está enmarcado dentro de un proyecto de investigación más grande, el trabajo presentado para la asignatura Ingeniería de software I desarrolla una implementación parcial del sistema propuesto que evidencia el uso de los conocimientos adquiridos durante el curso. Esta implementación parcial contiene la publicación de dos servicios con arquitectura Restful utilizando los métodos GET y POST del protocolo Http que permiten realizar búsquedas de recursos de datos abiertos enlazados en una base de datos de índices. Los servicios web están conectados al sistema multi agentes a través de una interfaz provista por la plataforma JADE. Los agentes se encargan de recoger la información de la búsqueda consultar la base de datos y devolver los resultados al usuario.

3 Análisis de requerimientos

3.1 Requerimientos Funcionales

- El sistema debe permitir realizar búsquedas a un usuario por término de búsqueda
- El sistema debe rastrear automática y periódicamente la nube de datos enlazados para indexar los recursos disponibles
- El sistema debe clasificar los recursos según niveles de confianza teniendo en cuenta métricas de calidad de confianza

- El sistema debe registrar los datos de procedencia de los recursos rastreados y permitir definir niveles de confianza para las entidades de procedencia.
- El sistema debe permitir a los usuarios calificar los recursos encontrados para evaluar las métricas subjetivas de las dimensiones de confianza.
- El sistema debe permitir consultar el estado de rastreo e indexación de recursos.
- El sistema debe permitir añadir fuentes de rastreo de recursos

3.2 Requerimientos no funcionales

- La solución debe ser implementada a través de un Sistema Multiagentes
- El sistema debe responder las consultas de los usuarios en menos de 5 segundos.
- El sistema debe presentar los resultados de búsqueda de forma amigable, el usuario debe llegar al recurso solicitado en máximo de 2 clics.
- El sistema de consulta debe estar disponible a través de Servicios Web con arquitectura Restful.

4 Metodología

Para el desarrollo del sistema se utilizó la metodología desarrollada por [1], descrita en la gráfica, desarrollada específicamente para JADE (Java Agent DEvelopment Framework).:

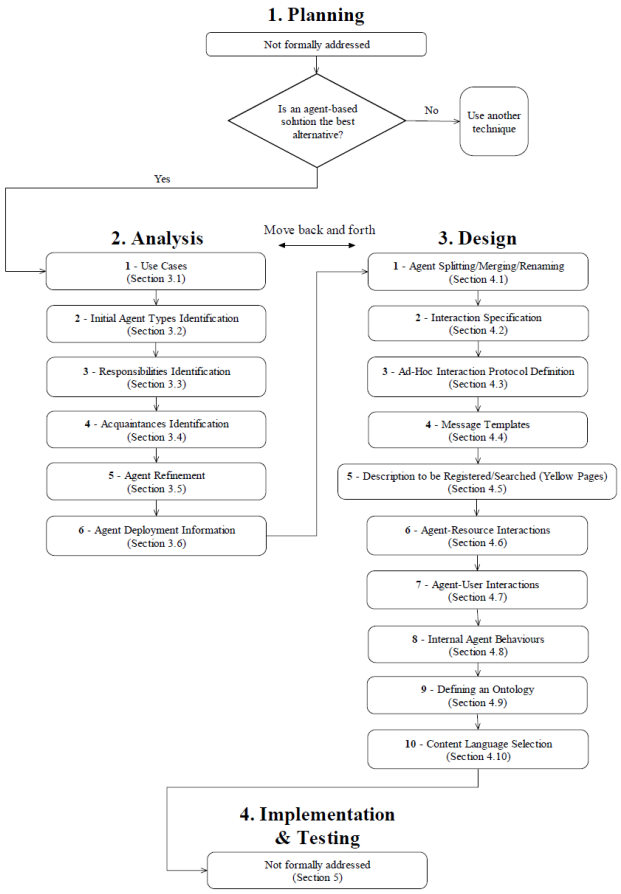

Figure 1: Metodología JADE [1]

5 Aplicación de la metodología

5.1 Planeación

Dentro de la metodología no existe una definición formal de la etapa de planeación, por lo tanto se realizó el análisis de requerimientos listados anteriormente. Resolviendo la pregunta indicada por la metodología y teniendo en cuenta que la nube de Datos abiertos enlazados está en constante crecimiento se seleccionó un modelo basado en agentes inteligentes, dadas sus características de autonomía e interacción. Los agentes tienen la responsabilidad de mantener la evaluación de confiabilidad de recursos a través el tiempo y mantener disponible los resultados para la consulta por parte del usuario.

Adicionalmente se creó un modelo de arquitectura de alto nivel para describir las funcionalidades básicas del sistema del sistema. Por las características de

los sistemas multiagentes, la semántica de lo los lenguajes ADL no es suficiente para describir la arquitectura de estos sistemas. Por esto se utilizó la propuesta presentada en [3] para elaborar el modelo de arquitectura del sistema. Esta propuesta utiliza redes de Petri y calculo π para modelar la arquitectura dinámica del sistema. Elementos:

1. Tuplas O,MPR
   - O: Conjunto de objetos de redes de Petri (OPN)
   - MPR: Conjunto de mensajespasando relaciones a través de los objetos
2. Objetos Oi
   - Tuplas P,IP,OP,T,F,IIA,OIA,E,C
     - P: Posición de objetos físicos
     - P: Posición de entrada
     - OP: Posición de salida
     - : Transición
     - F: Relaciones entre entrada-Salida
     - IA: Arco de interface de entrada
     - OIA:: Arco de interface de salida
     - : Función de expresión del arco
     - C: Color

Agentes de cómputo: Son los responsables de la comunicación con los usuarios y el ambiente, Se basan en el modelo de intención, deseo, intención.

Agentes de comunicación: Facilitan la comunicación entre los agentes, definen las reglas de la interacción de la información.

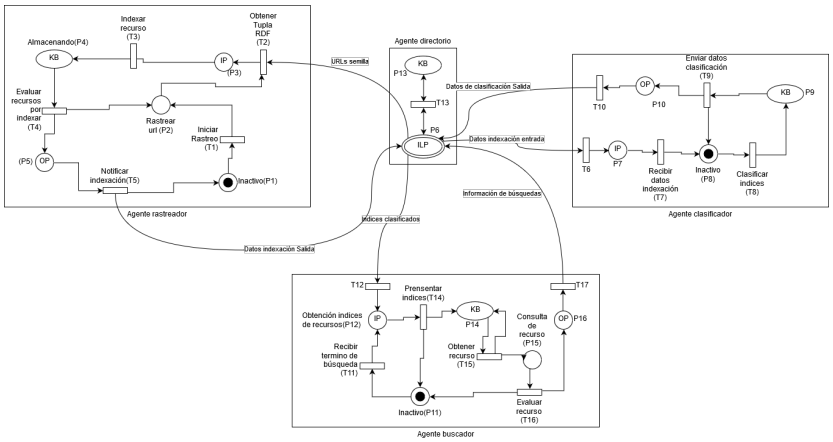

Figure 2: Arquitectura del sistema

5.2 Análisis

5.2.1 Casos de uso

Se identificaron 3 actores, plasmados en el diagrama de casos de uso:

- Un usuario que realiza búsquedas en el sistema multiagente
- Un rastreador de recursos que rastrea la nube de datos abiertos enlazados y almacena los índices de los recursos encontrados
- Un clasificador de evalúa el nivel de confianza y lo registra

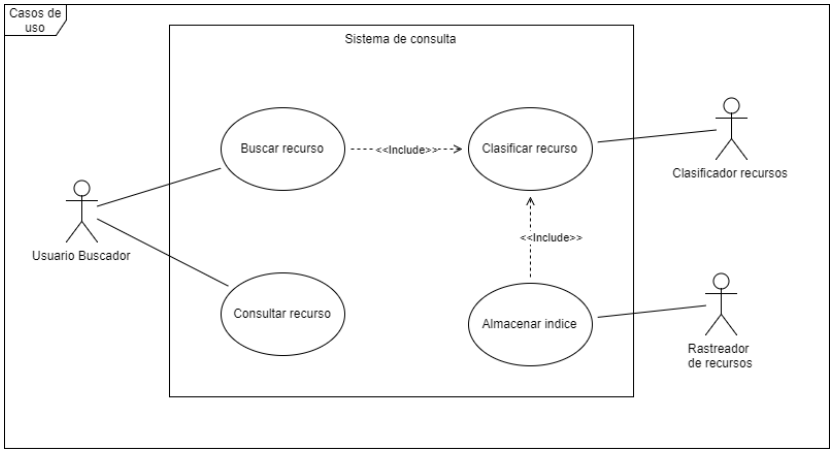

Figure 3: Diagrama de casos de uso

5.2.2 Identificación de agentes

A diferencia de la notación de UML, en la identificación de agentes se discriminan las personas y los sistemas de cómputo. En el diagrama, las personas se representan con el actor como en los casos de uso, y los otros sistemas se representan con cuadrados.

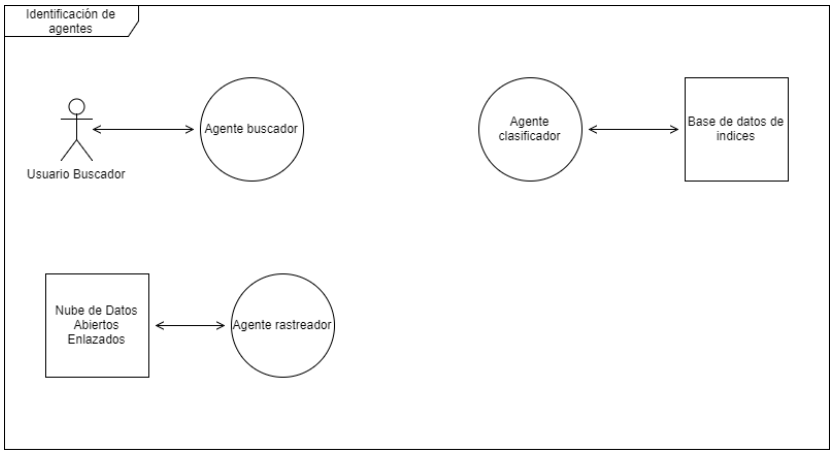

Figure 4: Identificación de agentes

5.2.3 Identificación de responsabilidades

En este paso se elaboró una tabla de responsabilidades derivada de los casos de uso identificados en el paso 2.1.

Tipo de Agente	Responsabilidades
Agente rastreador	Rastrear la nube de datos abiertos enlazados y obtener recursos
Agente clasificador	Clasificar recursos indexados Recuperar recursos solicitados del repositorio central Almacenar índices de recursos en un repositorio central
Agente buscador	Obtiene la consulta del usuario Devuelve resultados de búsqueda Devuelve recurso al usuario

Table 1: Identificación de responsabilidades

5.2.4 Identificación de conocidos

En este pase se identifica la interacción necesaria entre los agentes, se actualiza el diagrama de identificación de agentes con las relaciones y aparecen nuevas responsabilidades.

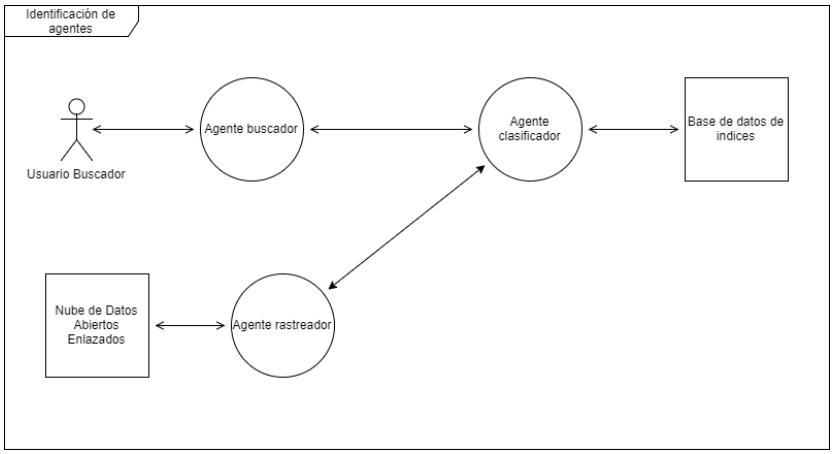

Figure 5: Interacción entre agentes

Tipo de Agente	Responsabilidades
Agente rastreador	Rastrear la nube de datos abiertos enlazados y obtener recursos Informar la obtención de un recurso al agente clasificador pertinente
Agente clasificador	Clasificar recursos indexados Recuperar recursos solicitados del repositorio central Presentar recursos a agentes buscadores Actualizar información de consulta del recurso Modificar clasificación según uso del recurso Almacenar índices de recursos en un repositorio central
Agente buscador	Obtiene la consulta del usuario Devuelve resultados de búsqueda Devuelve recurso al usuario Solicitar recurso a un agente clasificador Informar al agente clasificador acerca de la efectividad de la búsqueda

Table 2: Responsabilidades agentes con interacción

5.2.5 Refinamiento de agentes

En este paso se aplican algunas consideraciones a los agentes identificados, relativas a los aspectos descritos en seguida:

- Soporte: En el paso anterior ya se ha tenido en alguna información necesaria para que los agentes realicen su función. El agente rastreador requiere un listado de urls semilla para realizar el rastreo, el agente clasificador requiere la información provista por el rastreador y las reglas para realizar la clasificación de los recursos, además de información de calificación provista por el usuario.

- Descubrimiento: Para que conocer que agentes se encuentran activos y que servicios prestan se añadió un agente de directorio para implementar un mecanismo de páginas amarillas para descubrir nuevos servicios.
- Monitoreo: Para realizar tareas de monitoreo se definió un agente monitor de rastreo que se encarga de recopilar los errores en los agentes y de verificar el estado global del rastreo.

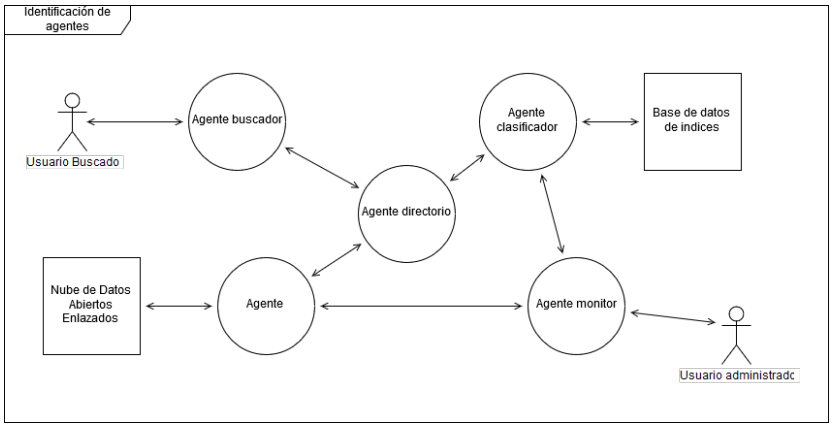

Figure 6: Identificación agentes después de refinamiento

Tipo de Agente	Responsabilidades
Agente rastreador	Rastrear la nube de datos abiertos enlazados a partir de un listado de url semilla y obtener recursos Informar la obtención de un recurso al agente clasificador pertinente
Agente clasificador	Clasificar recursos indexados Recuperar recursos solicitados del repositorio central Presentar recursos a agentes buscadores Actualizar información de consulta del recurso Modificar clasificación según uso del recurso Almacenar índices de recursos en un repositorio central
Agente buscador	Obtiene la consulta del usuario Devuelve resultados de búsqueda Devuelve recurso al usuario Solicitar recurso a un agente clasificador Informar al agente clasificador acerca de la efectividad de la búsqueda
Agente directorio	Registrar los servicios de agentes conectados al sistema
Agente monitor	Monitorear el estado de indexación, clasificación y aparición de errores

Table 3: Responsabilidades agentes refinamiento

5.2.6 Información de despliegue de los agentes

El despliegue de los agentes se realizará en 3 dispositivos, un servidor central que indexa clasifica y expone un Servicio restful con la información indexada. Un cliente http para que interactúe con el usuario. Agente rastreador externo para indexar repositorios por fuera de la nube de datos abiertos enlazados.

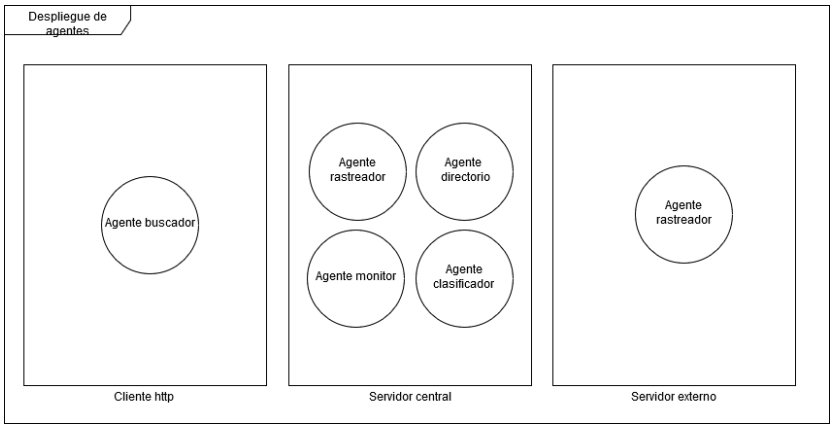

Figure 7: Identificación agentes después de refinamiento

5.3 Diseño

5.3.1 División/Unión/Cambio nombre agentes

En este paso no se modificarán los agentes obtenidos en la fase de análisis ya que los agentes que se diseñarán conservan el equilibrio entre la cantidad de agentes y la complejidad de cada agente.

5.3.2 Especificación de la interacción

Especificar las interacciones en una tabla con:

- Nombre de la interacción
- Responsabilidad de la tabla de responsabilidades de la fase de análisis
- Protocolo de interacción
- El rol del agente en la interacción (I- Iniciador, R- Respondiente)

5.3.3 Definición de protocolos de interacción Ad hoc

5.3.4 Definición de protocolos de interacción Ad hoc

Como todas las interacciones utilizaran el protocolo FIPA Request no es necesario definir protocolos AD HOC.

Interacción	Resp	Protocolo Int	Role	Con	Cuando
Buscar termino	1	FIPA REQUEST	I	Agente buscador	El usuario envía un término de búsqueda
Obtener resultados búsqueda	2	FIPA INFORM	R	Agente clasificador	El agente recibe un término para búsqueda
Devolver recursos a agente buscador	3	FIPA INFORM	I	Agente clasificador	El agente clasificador recupera la información

Table 4: Tabla de interacción

5.3.5 Plantillas de mensajes

Interacción	Plantilla
Buscar termino	Búsqueda
Obtener resultados búsqueda	ResultadoBusqueda
Devolver recursos a agente buscador	ResultadoBusqueda

Table 5: Tabla de plantillas de mensajes

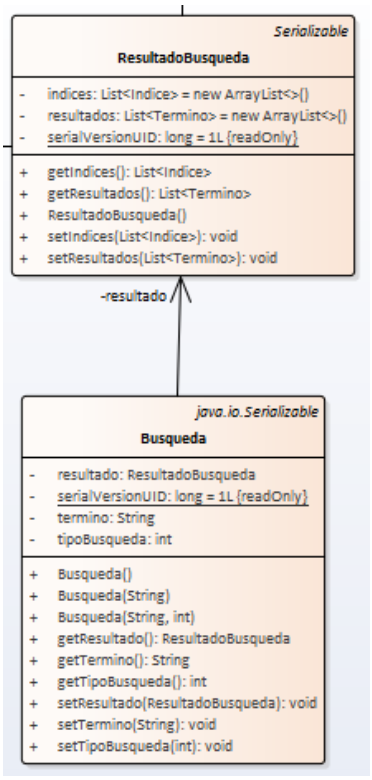

Figure 8: Plantillas de mensajes

5.3.6 Descripción a ser registrada/consultada en páginas amarillas.

Los Agentes diseñados hasta el momento no se registran ni realizan consultas a las páginas amarillas

5.3.7 Interacción Agentes – Recursos

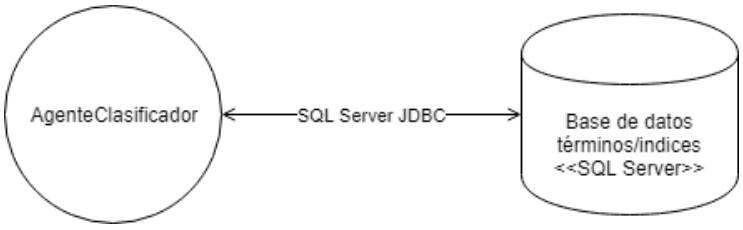

Figure 9: Interacción AgenteClasificador con Base de datos

5.3.8 Interacción Agentes – Usuarios

La interacción con el usuario se realiza a través de un servicio REST que se conecta con el AgenteBuscagor que extiende de la clase GatewayAgent que ermite establecer la comunicación entre JADE y recursos externos.

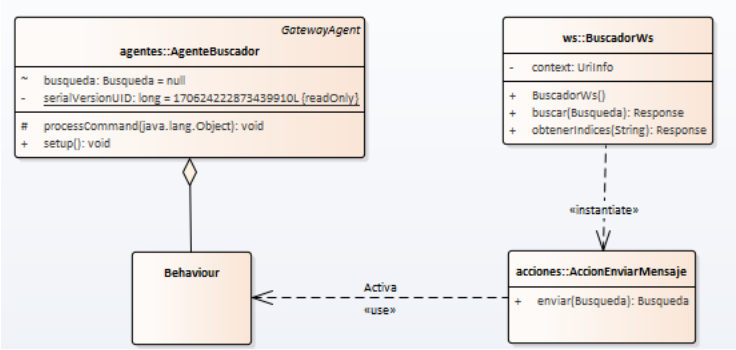

Figure 10: Interacción AgenteBuscador con Clase de Implementación Servicio web

5.3.9 Comportamientos internos agentes

Los comportamientos son las características de los agentes que les permiten cambiar la información del medio.

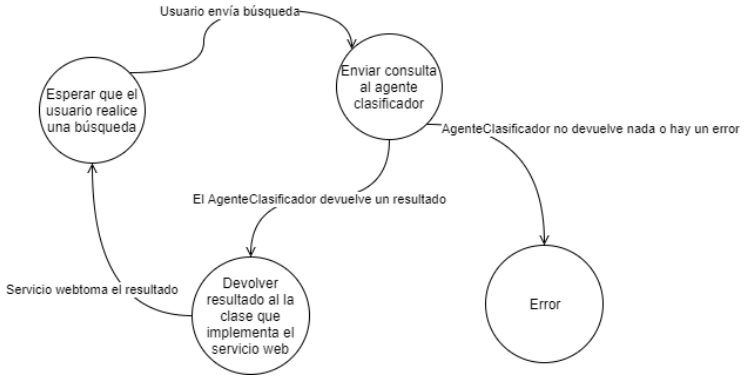

Figure 11: Comportamiento AgenteBuscador

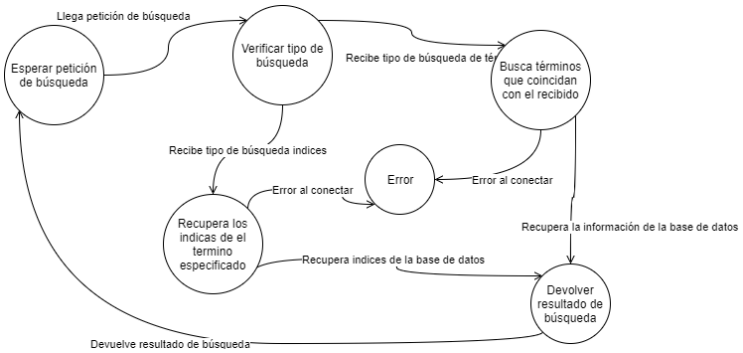

Figure 12: Comportamiento AgenteClasificador

5.3.10 Definiendo una ontología

Para el diseño de las funcionalidades actuales no ha sido necesario definir una Ontología.

5.4 Diseño integración con servicio web

Se diseño un servicio web que utiliza Las librerías de JBOSS Wildfy, específicamente REST Easy que interactua a través de la clase GatewayAgent con JADE para desplegar un AgenteBuscador que envíe las búsquedas de los usuarios a la plataforma JADE.

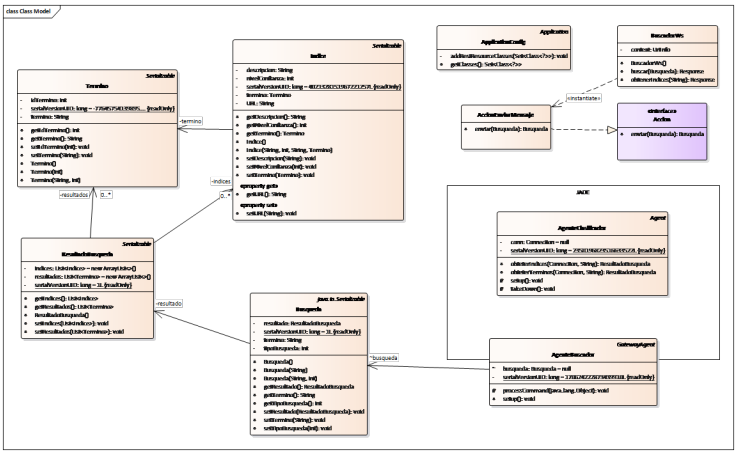

Figure 13: Diagrama de clases de toda la aplicación

5.4.1 Diseño de base de datos

Se diseño una base de datos para el almacenamiento de términos e índices.

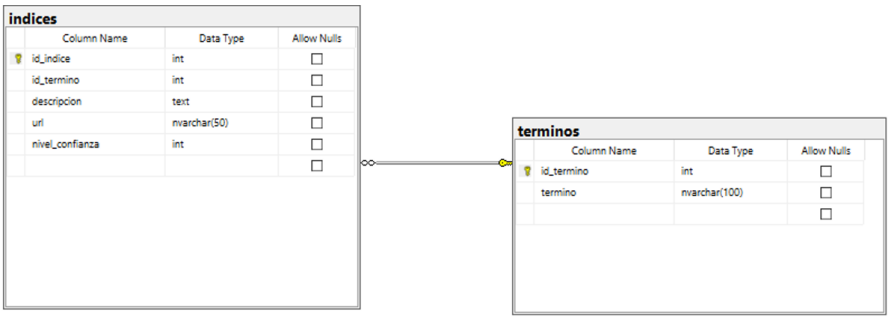

Figure 14: Diagrama de la base de datos

6 Implementación

Para la implementación se utilizó el lenguaje de programación Java Enterprise edition. Se desarrollo una aplicación web que se ejecuta sobre un servidor de aplicaciones JBOSS Wildfly para permitir a los usaurios hacer las consultas. La consulta de los índices se realiza sobre una base de datos SQL server con el diseño presentado en la sección anterior. Los Agentes se desplegaron sobre la plataforma JADE.

Para verificar el funcionamiento de la plataforma se deben realizar las actividades listadas a continuación.

6.1 Restaurar base de datos

En el directorio adjunto Despliegue/DB se encuentran un backup para restaurar la base de datos a un SqlServer local. Si hay problemas restaurando el backup, se debe replicar la base de datos y ejecutar el archivo "datos" que contiene las inserciones con los datos de pruebas.

En el servidor donde se ejecute la plataforma solo debe ejecutarse una instancia de SQL Server ya que si se ejecutan dos como SQLSERVER Development Edition y SQLSERVER XPRESS EDITION.

6.2 Compilar librerías agentes

La clase AgenteClasificador es la que se encarga de la interacción con el repositorio de índices. Antes de compilar la librería de agentes es necesario establecer la cadena de conexión a la base de datos que se está utilizando. Una vez modificado el proyecto desde Eclipse dar clic derecho en el proyecto y seleccionar la opción exportar JAR. Luego seleccionar los paquetes agentes e is1, seleccionar la carpeta de librerías de jade y guardar.

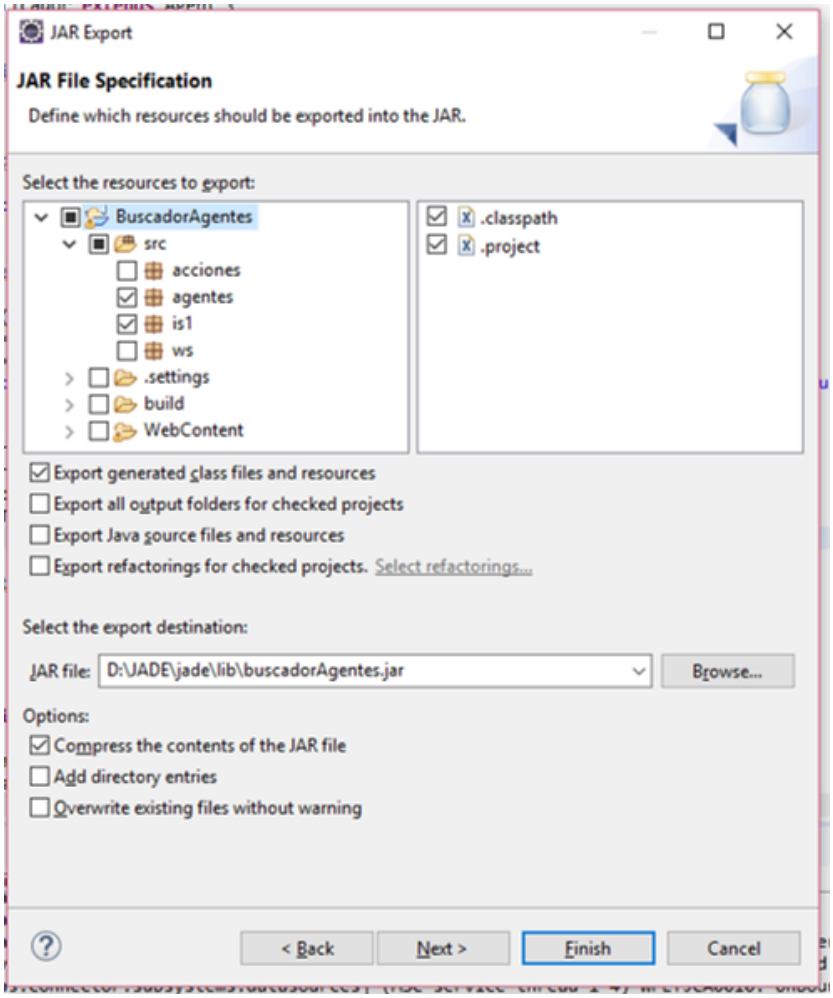

Figure 15: Ventana exportación de librerías en eclipse

Copiar librería con el driver de SQL server al mismo directorio.

6.3 Iniciar contenedor de agentes

Iniciar una consola y sobre el directorio Jade ejecutar el comando de inicio del contenedor de agentes para la última versión de java:

```
java -cp "lib/jade.jar;lib/buscadorAgentes.jar;lib/mssql-jdbc-6.2.2.jre8.jar;lib/commons-coc
```

Si existe algún conflicto entre el driver JDBC y la versión de java se debe utilizar una versión anterior del driver:

```
java -cp "lib/jade.jar;lib/buscadorAgentes.jar;lib/sqljdbc4-2.0.jar;lib/commons-codec/commor
```

Como se puede observar en el comando se incluyen las librerías necesarias en el Classpath de java para que los agentes puedan realizar sus tareas.

6.4 Desplegar Agente clasificador

Desde la interfaz gráfica de administración de JADE lanzar AgenteClasificador desde el contenedor principal

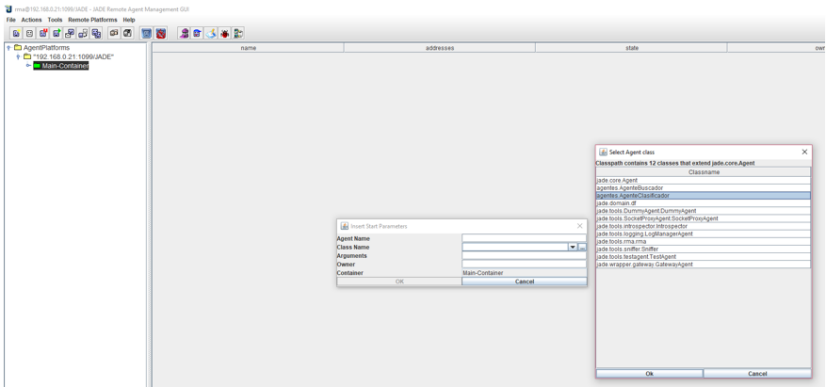

Figure 16: Inicialización AgenteClasificador

En el menú herramientas iniciar el agente sniffer y añadir el AgenteClasificador para ver la interacción entre los agentes.

Figure 17: Inicialización sniffers

6.5 Ejecución del servicio Web

Desde el proyecto de eclipse, ejecutar la aplicación en un servidor de aplicaciones. Como se mencionó anteriormente en este caso se utilizó el servidor JBOSS WILDFLY 9.

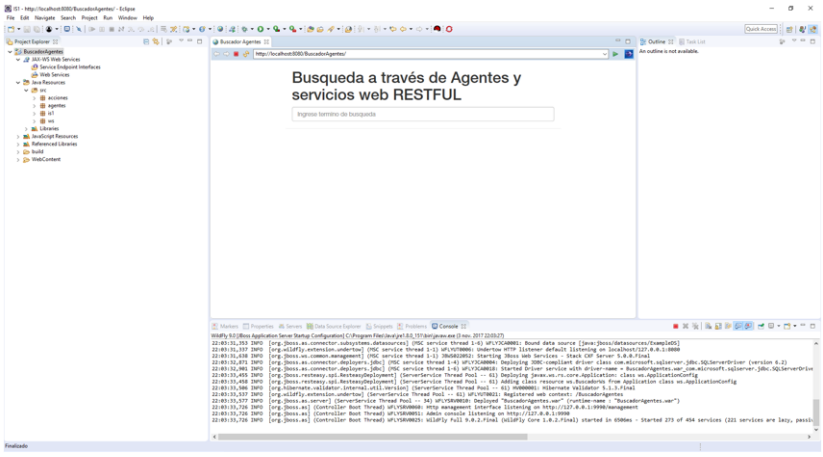

Figure 18: Ejecución del proyecto en eclipse

6.6 Prueba de servicios web

Para realizar pruebas del servicio web del proyecto se utilizó el complemento de Google Chrome Advance Rest Client, un cliente http que permite realizar peticiones http haciendo uso de todos los métodos del protocolo. Posteriormente se implementó una vista en Html con funciones Javascript para consumir el api.

6.7 Cliente REST

Para descargarlo solo es necesario buscar en google Advance Rest Client, y abrir el complemento.

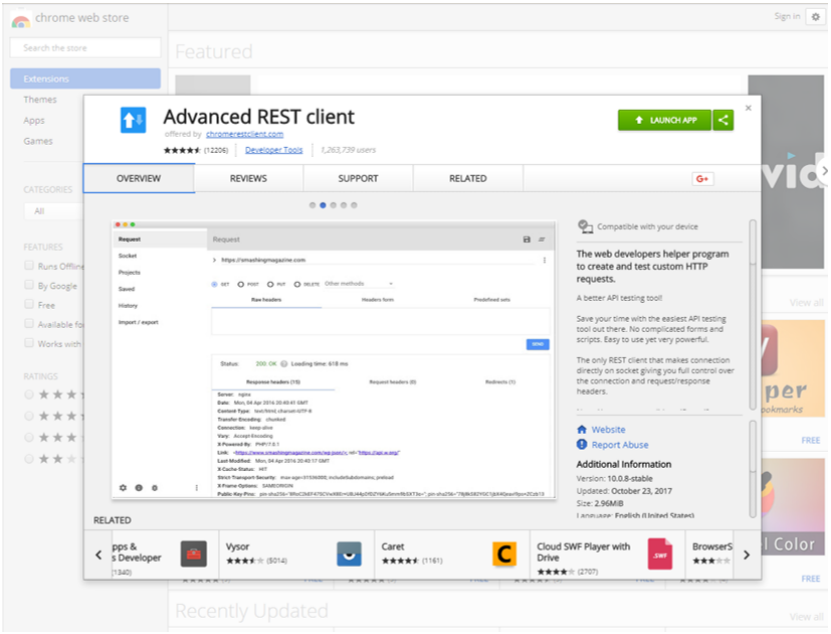

Figure 19: Cliente servicios web Rest Easy para Chrome

6.7.1 Uso

Este complemento nos permite realizar peticiones con cualquier método, añadir las cabeceras necesarias e ingresar el cuerpo del mensaje en diferentes formatos (Texto plano, xml, json). Además de esto nos permite almacenar las consultas a los servicios web y generar un archivo con las peticiones almacenadas. Para utilizarlo solo debemos especificar el método, la URL las cabeceras y el cuerpo del mensaje. Para el caso del proyecto se generaron dos servicios, uno POST y uno GET

6.7.2 Servicio No 1: Obtener términos

Método: POST
URL: http://localhost:8080/BuscadorAgentes/servicios/buscar
Con la cabecera: Content/Type: application/json
En el cuerpo un objeto búsqueda en formato JSON:

```
{
    "termino":"termino de busqueda"
}
```

Resultado: Posibles términos que coinciden con la búsqueda

```
{
    "termino": "Abaratar el despido",
```

```
    "idTermino": 3
},
```

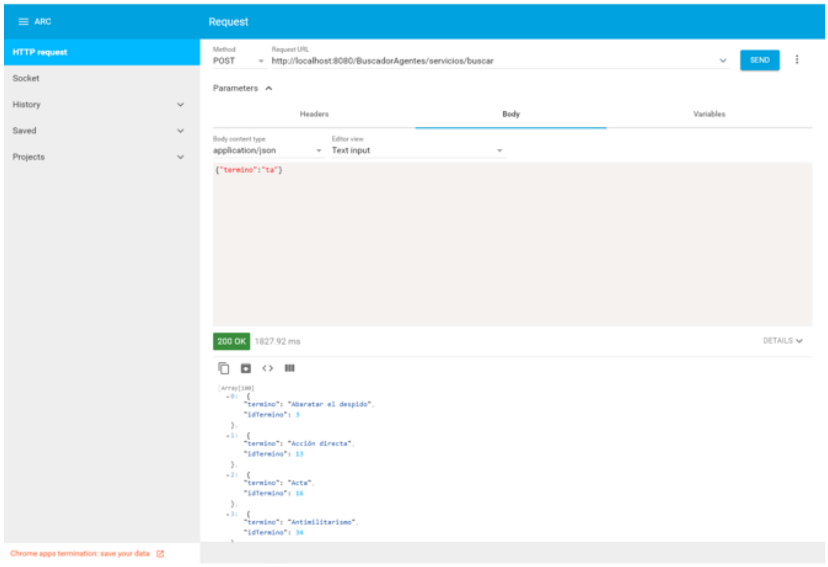

Figure 20: Consumo de servicio términos desde ReastEasy Chrome

6.7.3 Servicio No 2: Obtener recursos

Método: GET
URL: http://localhost:8080/BuscadorAgentes/servicios/buscar/id_recurso
Con la cabecera: Content/Type: application/json
Resultado: Recursos indexados que coinciden con el termino

```
{
    "descripcion": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec
    volutpat felis enim, nec dictum mi dictum quis. Donec feugiat auctor nisi sit
    amet eleifend. Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
    "nivelConfianza": 637,
    "termino": {
        "termino": null,
        "idTermino": 36
    },
    "url": "https://www.google.com.co"
},
```

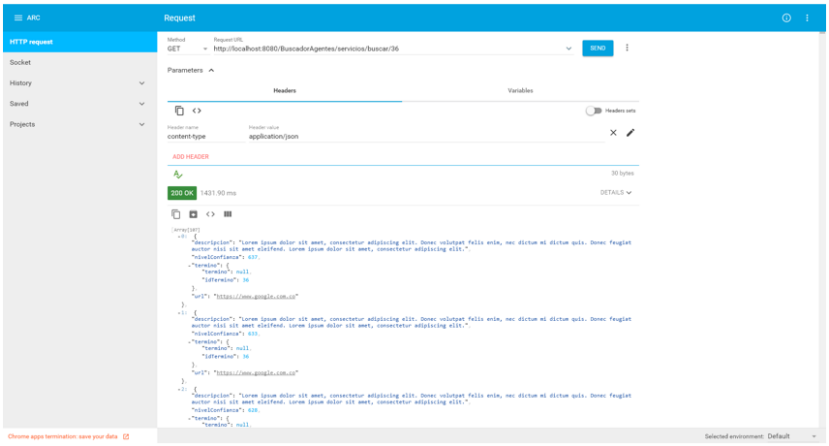

Figure 21: Consumo de servicio indices desde ReastEasy Chrome

6.8 Pruebas desde la interfaz

La interfaz consiste en un Archivo HTML con un campo de texto para ingresar el término de búsqueda. Al escribir el termino aparece un listado se sugerencias y al seleccionar la sugerencia se muestran los recursos clasificados por nivel de confianza

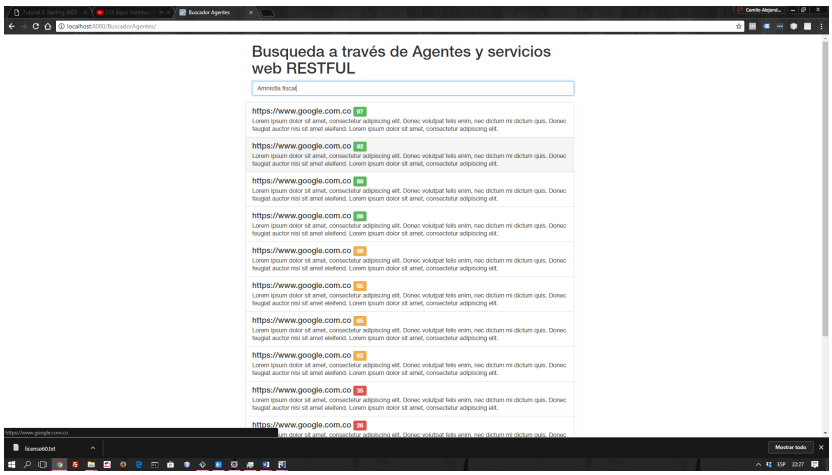

Figure 22: Interfaz gráfica aplicación

6.9 Mensajería entre agentes

Desde el Agente Sniffer se puede observar el paso de mensajes entre los agentes.

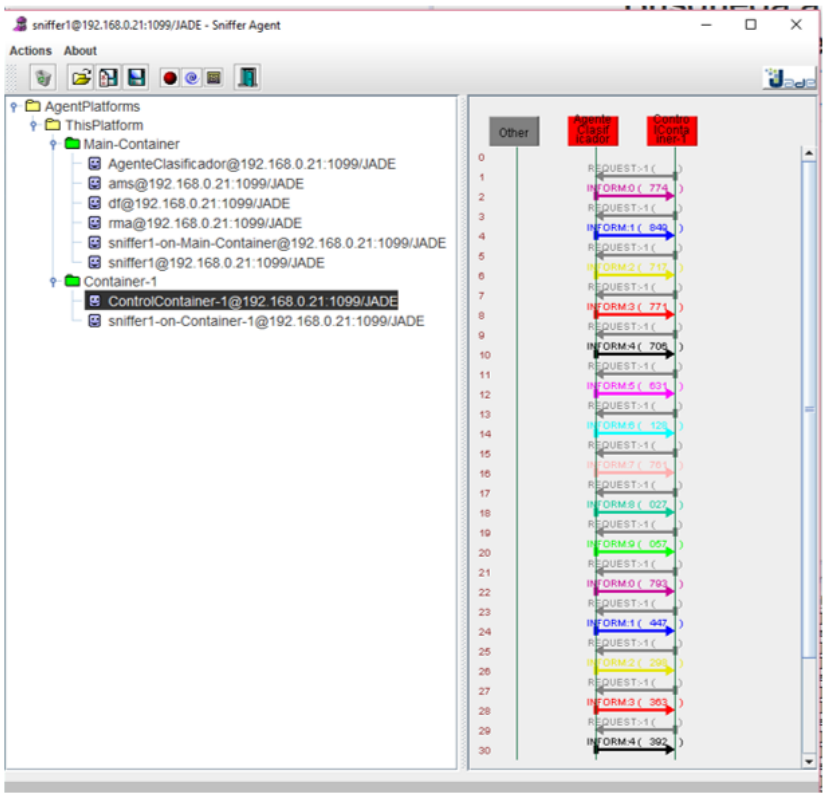

Figure 23: Interfaz gráfica aplicación

7 Despliegue

El despliegue se realizó en un servidor central utilizando el servicio en la nube de Amazon EC2 utilizando una cuenta para educación Ver anexo 1 (Creación de cuenta Amazon);

7.1 Infraestructura:

Para el servidor de aplicaciones se utilizó un sistema operativo Redhat Enterprise Edition. La plataforma donde se ejecutan los servicios web es la plataforma libre Wildfly y la plataforma jade para desplegar los agentes.

7.2 Creación instancia base de datos

Para la base de datos se utilizó el servicio de Amazon RDS (Relational Database Service) y el proceso de creación puede ser verificado en el Anexo 2.

7.3 Creación instancia y conexión

A continuación se definen los pasos para la creación de la instancia.

1. En la consola de administración de Amazon se debe utilizar el servicio EC2

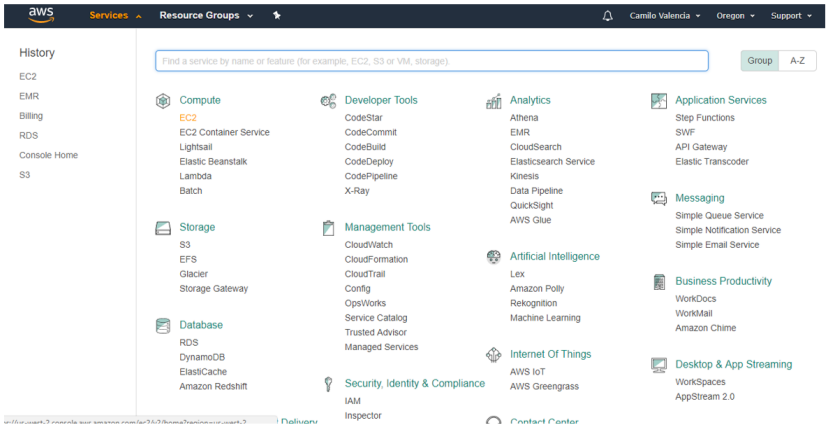

Figure 24: Captura de pantalla Consola de administración de Amazon

2. Iniciar instancia desde el panel de administración EC2

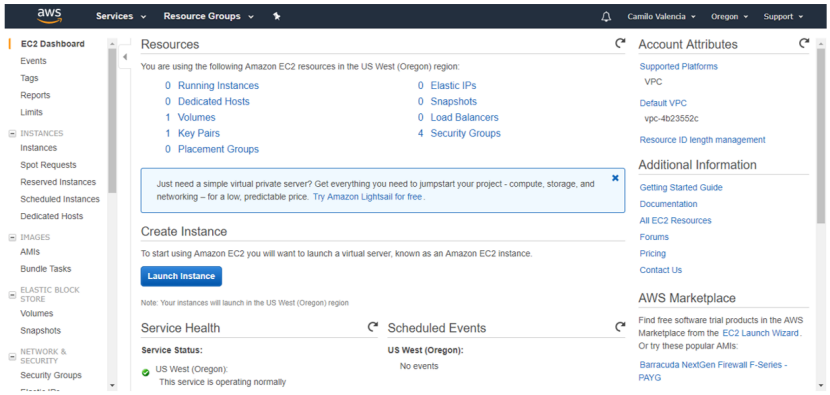

Figure 25: Panel administración instancias

3. Seleccionar Sistema Operativo Se debe seleccionar un sistema operativo, en la lista de sistemas disponibles los que tienen la marca FREE TIER ELEGIBLE son los que se pueden utilizar de forma gratuita durante 1 año.

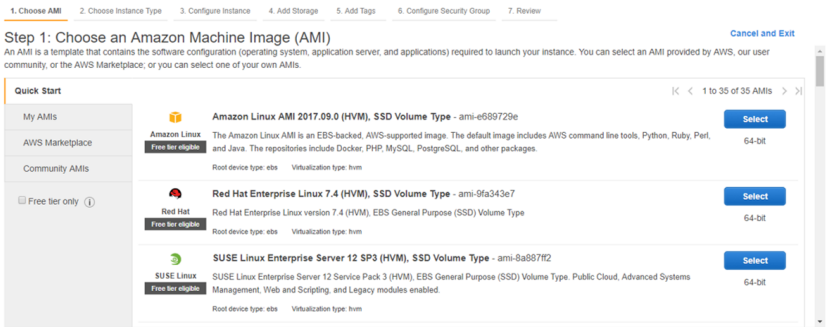

Figure 26: Paso 1 Selección del sistema operativo

4. Seleccionar tipo de instancia Se debe seleccionar el tipo de instancia según la memoria, numero de procesadores, calidad de la red entre otras. Como en el paso anterior. Nuevamente para usar la capa gratuita de Amazon elegimos FREE TIER ELEGIBLE

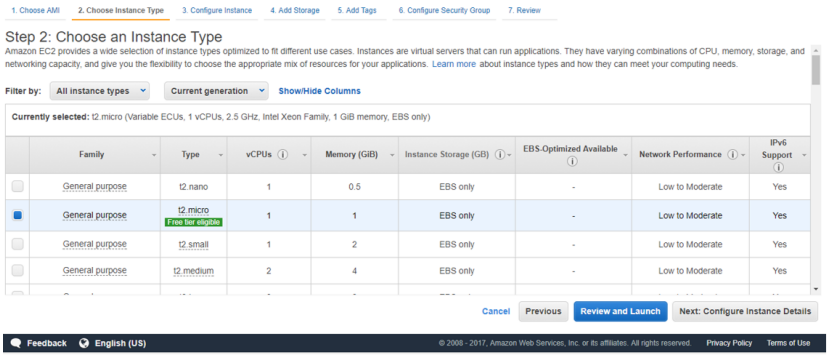

Figure 27: Seleción tipo de instancia

5. Configuración de seguridad Editamos el grupo de seguridad para permitir todo el tráfico de red. En un escenario de producción se debería permitir solo el tráfico necesario según principio del menor privilegio.

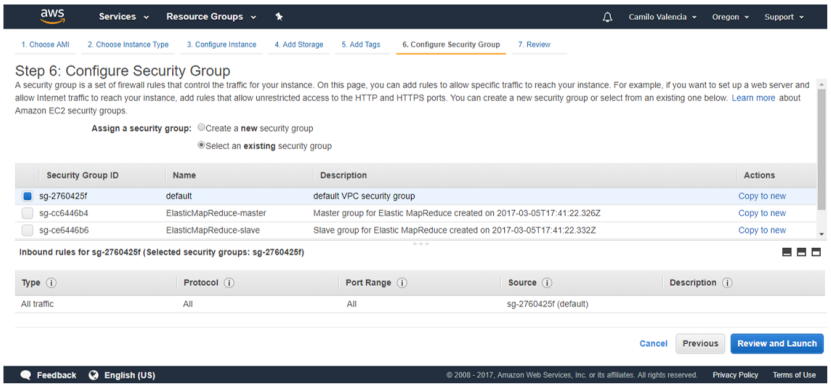

Figure 28: Configuración de seguridad

6. Iniciar instancia Se lanza la instancia: en este paso se deben generar las llaves de autenticación y almacenarlas en un lugar seguro para realizar la posterior conexión a través de ssh.

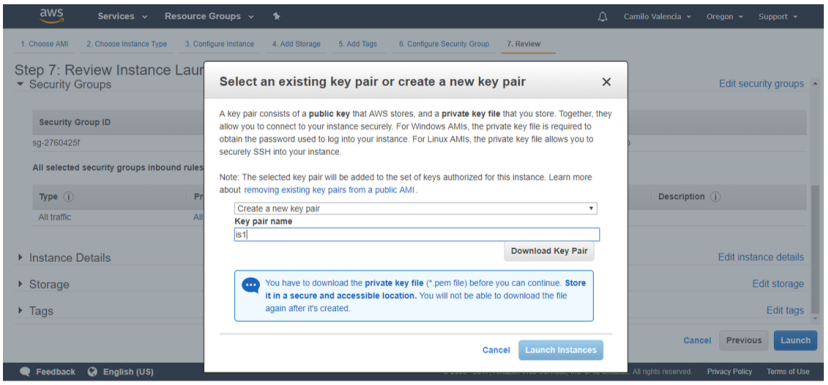

Figure 29: Creación de llaves para conexión SSH

Esperar unos minutos a que la instancia sea lanzada.

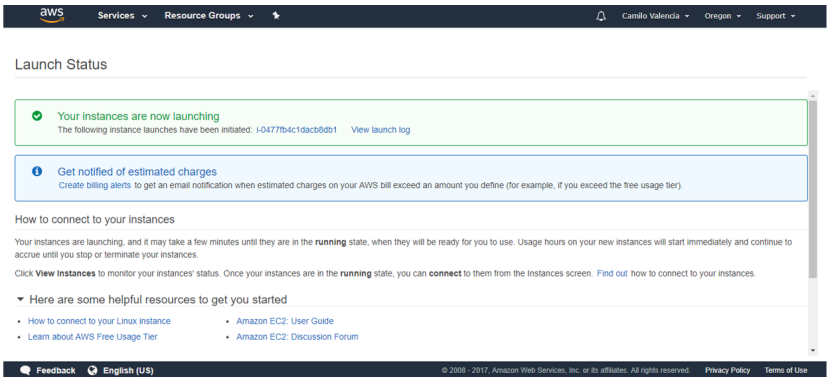

Figure 30: Ejecución instancia

7.4 Instalación de software y plataformas necesarias

Para la instalación del software es necesario conectarse a través de ssh y ejecutar las comandos descritos a continuación:

1. Obtenemos la cadena de conexión desde el panel de administración de instancias, seleccionando las instancias hacer clic en acciones y conectar.

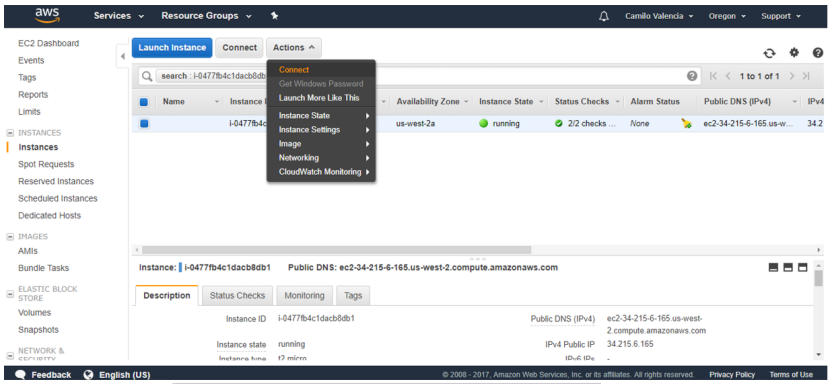

Figure 31: Opción conectar del panel de administración

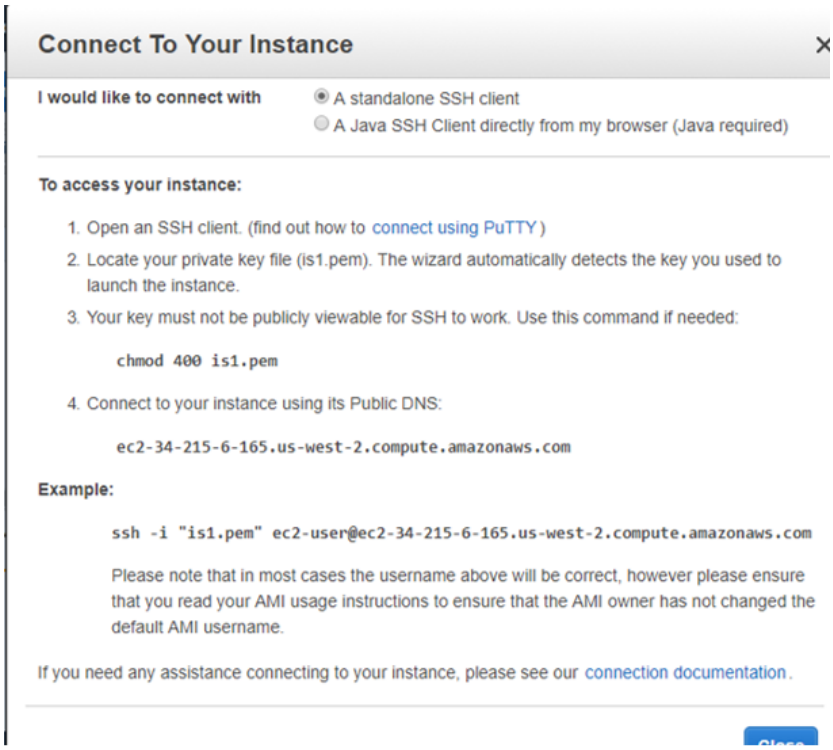

Figure 32: Cadena de conexión SSH

2. Conexión SSH Conectarse por consola a través de Putty o algún cliente ssh. En estè caso se utilizó la consola Git Bash que tiene el comando para realizar la conexión.

```
ssh -i "is1.pem"
ec2-user@ec2-34-215-6-165.us-west-2.compute.amazonaws.com
   \begin{figure}[H]
  \centering
    \includegraphics[width=0.9 \textwidth]{am10}
  \caption{Conexión SSH}
  \label{semanticweb:workflowedge}
\end{figure}
```

3. Instalación Software

   - Actualizar el software y los repositorios pre instalados

     `Sudo yum {y update`

- Instalación de java

  ```
  Sudo yum {y install java
  ```

- Instalación JBOSS

  ```
  Sudo yum {y install wget
  // Obtener zip
  wget http://download.jboss.org/wildfly/
  11.0.0.Final/wildfly-11.0.0.Final.zip
  // Instalar unzip para descomprimir
  Sudo yum {y install unzip
  // Descomprimir
  unzip wildfly-11.0.0.Final.zip
  // Ir al directorio de binarios
  cd wildfly-11.0.0.Final/bin/
  ```

- Añadir usuario Administrador

  ```
  ./add-user.sh
  ```

- Inicializar el servidor de aplicaciones

  ```
  sudo ./standalone.sh -b 0.0.0.0 -bmanagment 0.0.0.0
  ```

4. Verificar conexión: Obtener el DNS público de la consola de administración de la instancia:

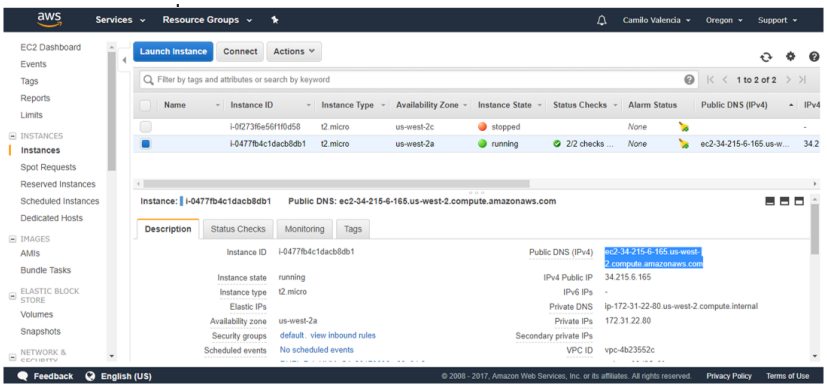

Figure 33: DNS público de instancia

Ir a la enlace publico puerto por defecto 8080 y se debe visualizar la pantalla de inicio de Wildfly

http://ec2-34-215-6-165.us-west-2.compute.amazonaws.com:8080/

Figure 34: Página inicial de JBOSS

5. Verificar acceso a la consola de administración Ingresar a http://ec2-34-215-6-165.us-west-2.compute.amazonaws.com:9990 Utilizar el usuario y contraseña definidos anteriormente.

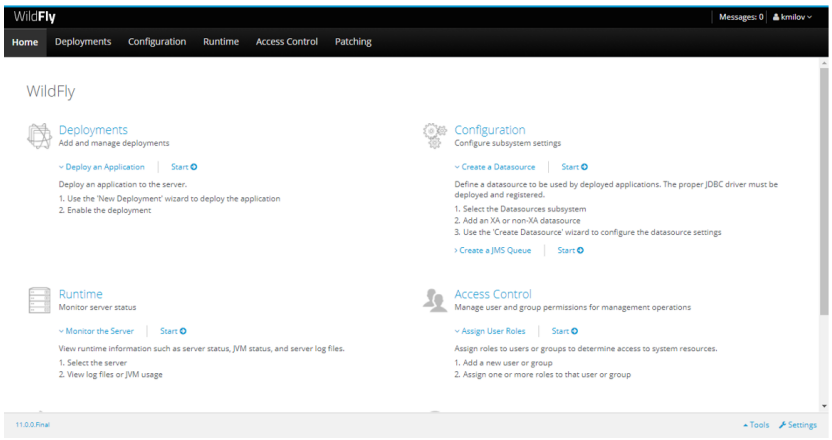

Figure 35: Consola administración JBOSS

6. Desplegar aplicación

   - Subir librerías de agentes a la instancia Se cargan las librerías a través de SCP(Secure copy)

     ```
     scp -i "is1.pem" ../Dropbox/Docs\ Maestria/Clases/IS1/Despliege/*
     ec2-user@ec2-34-215-6-165.us-west-2.compute.amazonaws.com:jade/lib
     ```

Figure 36: Resultado carga de archivos

- Iniciar plataforma y ejecutar los agentes Conectarse por ssh a la instancia y ejecutar dentro del directorio jade:

  ```
  java -cp "lib/jade.jar:lib/buscadorAgentes.jar:
  lib/mssql-jdbc-6.2.2.jre8.jar:lib/commons-codec/commons-codec-1.3.jar"
  jade.Boot -agents AgenteClasificador:agentes.AgenteClasificador
  ```

- Desplegar aplicación web Desde el proyecto de Eclipse, exportar un compilado .war de la aplicación. Ir a la consola de administración de JBOSS y en el menú Deployments, subir la aplicación compilada .war. Una vez desplegada la aplicación los servicios web quedarán disponibles en la url:
  http://ec2-34-215-6-165.us-west-2.compute.amazonaws.com:8080/BuscadorAgentes

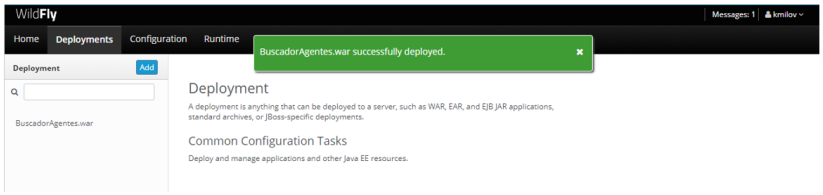

Figure 37: Despliegue archivo .war

8 Resultados

- Aplicación publicada en AWS (Amazon Web Services): http://ec2-34-215-6-165.us-west-2.compute.amazonaws.com:8080/BuscadorAgentes
- Documentación Java http://ec2-34-215-6-165.us-west-2.compute.amazonaws.com:8080/BuscadorAgentes/
- Repositorio Git con el contenido completo del proyecto: https://github.com/CamiloValencia/BuscadorAg

Referencias

[1] Magid Nikraz, Giovanni Caire, and Parisa A Bahri. A methodology for the development of multi-agent systems using the jade platform. *International Journal of Computer Systems Science & Engineering*, 21(2):99–116, 2006.

[2] Camilo Alejandro Valencia Martínez. *Definición de un Modelo Ontológico, que Facilite la Medición de Niveles de Usabilidad de Herramientas en Plataformas LCMS Mediante SNA (Social Network Analysis)*. 2016.

[3] Zhenhua Yu, Yuanli Cai, Ruifeng Wang, and Jiuqiang Han. π-net adl: An architecture description language for multi-agent systems. *Advances in Intelligent Computing*, pages 218–227, 2005.