CAPA DE SERVICIOS PARA LA PLATAFORMA DE PROCESAMIENTO DE DATOS EDUCATIVOS MASIVOS DE LA FACULTAD 4 DE LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

SERVICES LAYER FOR THE PLATFORM OF PROCESSING OF MASSIVE EDUCATIONAL DATA OF THE FACULTY 4 OF THE UNIVERSITY OF THE INFORMATICS SCIENCES

Orlando Grabiel Toledano López[1], Dariel Corchado López del Castillo[2], Ángel Alberto Vázquez Sánchez[3]

1 Universidad de las Ciencias Informáticas, Cuba, ogtoledano@uci.cu

2 Universidad de las Ciencias Informáticas, Cuba, dcorchado@uci.cu

3 Universidad de las Ciencias Informáticas, Cuba, aavazquez@uci.cu

RESUMEN: *La Arquitectura Orientada a Servicios permite la interconexión de sistemas aislados mediante un Bus de Servicios Empresarial, y cada organización a su vez es capaz de interconectarse y brindar servicios a otras. Esto permite extender y aportar un valor agregado a los sistemas de dominio específico permitiendo la interoperabilidad de los mismos. La Facultad 4 de la Universidad de las Ciencias Informáticas trabaja actualmente en el desarrollo de una plataforma para el procesamiento de datos educativos masivos. Con esta se pretende resolver problemas predictivos en el área de la estimación de conocimiento latente. Para este proceso, existen algoritmos que solucionan el problema, los cuales han sido implementados por el equipo de trabajo y se encuentran en fase de adaptación para la estimación de conocimiento en el contexto de datos masivos. Para ello, los mismos necesitan de bibliotecas que permitan la estimación de parámetros a partir de datos dicotómicos obtenidos por estudiantes que se someten a un grupo de ítems evaluativos, que pertenecen a un examen en línea. Para dar respuesta a la problemática se desarrolla una capa de servicios que permite publicar las principales funcionalidades de la biblioteca Psychometrics para la estimación de parámetros. Se define el proceso de selección de las herramientas y tecnologías para llevar a cabo el desarrollo de la propuesta de solución, los elementos de seguridad aplicados y el esquema general de la solución, así como aspectos técnicos relacionados con su implementación.*

Palabras Clave: Arquitectura Orientada a Servicios, Capa de servicios, Estimación de conocimiento latente, Minería de Datos Educativos Masivos.

ABSTRACT: *The Service Oriented Architecture allows the interconnection of isolated systems through a Business Services Bus, and each organization in turn is able to interconnect and provide services to others. This allows extending and adding value to the specific domain systems allowing the interoperability of them. The Faculty 4 of the University of Informatics Sciences is currently working on the development of a platform for the processing of massive educational data. This is intended to solve predictive problems in the area of latent knowledge estimation. For this process, there are algorithms that solve the problem, which have been implemented by the team and are in the adaptation phase for the estimation of knowledge in the context of massive*

data. For this, they need libraries that allow the estimation of parameters from dichotomous data obtained by students who undergo a group of evaluative items, which belong to an online test. To respond to the problem, a services layer is developed that allows publishing the main functions of the Psychometrics library for the estimation of parameters. The process of selecting the tools and technologies is defined to carry out the development of the solution proposal, the security elements applied and the general scheme of the solution, as well as technical aspects related to its implementation.

KeyWords: Service Oriented Architecture, Services layer, Latent knowledge estimation, Massive Educational Data Mining.

1. INTRODUCCIÓN

El uso extendido de las Tecnologías de la Información y las Comunicaciones por parte de las organizaciones ha incrementado la complejidad de los sistemas que satisfacen las demandas de gestión y procesamiento de la información. Cada día los sistemas son más complejos al incluir un gran número funcionalidades necesarias que garantizan la informatización de procesos en las empresas, pero sin la integración de estas funcionalidades no se logran unir los objetivos del negocio y los sistemas trabajan como entes aislados. Esto evidencia que los procesos no están optimizados y trae como consigo costos adicionales por no tener la capacidad de utilizar algo que ya existe y que la organización posee. [1] Para ello surge una Arquitectura Orientada a Servicios (SOA, siglas en inglés) que permite la interconexión y comunicación entre los procesos. Con esta se mejora la agilidad de la empresa para que responda de manera rápida a un entorno cambiante, logrando mayor escalabilidad en su infraestructura tecnológica, siempre y cuando el ambiente empresarial lo permita.

La Universidad de las Ciencias Informáticas posee una compleja y extensa infraestructura tecnológica, donde existen un considerable número de sistemas que no trabajan como entes aislados e intercambian información relacionada con los procesos medulares que se desarrollan en la universidad, tales como: Gestión universitaria, gestión documental, seguridad informática, redes y servicios telemáticos, entre otros. En muchos casos la interconexión de estos procesos realizados por los sistemas que los gestionan puede ser aún mejorable.

Como parte de la mejora de los procesos educativos docentes, principalmente en el área de los instrumentos evaluativos, en la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI) se está desarrollando una plataforma para el procesamiento de datos educativos masivos, la cual trabajará en la realización de tareas predictivas relacionadas con la estimación de conocimiento latente que poseen los estudiantes a partir de datos históricos recogidos de plataformas educativas. De estas plataformas, los datos que se recogerán son los relacionados con los exámenes en línea en los que ha participado un estudiante o grupos de estudiantes por asignaturas, y a partir de estos poder estimar el rasgo o conocimiento latente que posee cada uno. Con los resultados obtenidos del proceso de estimación la plataforma podrá gestionar de forma automática test adaptativos en línea que permitirán elevar el conocimiento que posee un estudiante en una determinada área, podrá recomendar mejoras a los instrumentos evaluativos y bancos de problemas con los que cuenten los sistemas y podrá predecir los resultados de un estudiante o grupos de estudiantes frente un determinado instrumento evaluativo.

Entre los métodos predictivos existentes, resulta de gran utilidad para el sector educacional la estimación de conocimiento latente, el cual permite determinar en qué medida un estudiante domina un conocimiento o habilidad determinada en un momento dado [2]. Para realizar una inferencia del conocimiento que posee un estudiante, entiéndase por conocimiento lo que un estudiante conoce (o domina) que puede ser significativo para la situación de aprendizaje actual (Habilidades, Hechos, Conceptos, Principios y Esquemas), existen diferentes algoritmos, tales como: Performance Factors Analysis (PFA), Bayesian Knowledge Tracing (BKT), Item Response Theory (IRT), los cuales proveen modelos donde a partir de datos dicotómicos obtenidos por diferentes test aplicados a los estudiantes permiten estimar conocimiento latente. Con ellos, de forma general, se puede medir qué conoce un estudiante en un momento específico, sus componentes de conocimiento y estimar su probabilidad de éxito frente a una pregunta o ítem en un examen.

La plataforma de procesamiento, actualmente en desarrollo, será un sistema distribuido, el cual comprenderá una capa de administración que incluirá herramientas como: HUE para la interfaz web, Zookeper y Oozie para la coordinación y planificación del flujo de trabajo. Se incluye además una capa para el procesamiento de datos, la cual comprende herramientas para la minería de datos: Mahout, Docker y Spark Mlib. Esta capa incluirá además motores de procesamiento, entre los cuales se puede aplicar MapReduce o Spark.

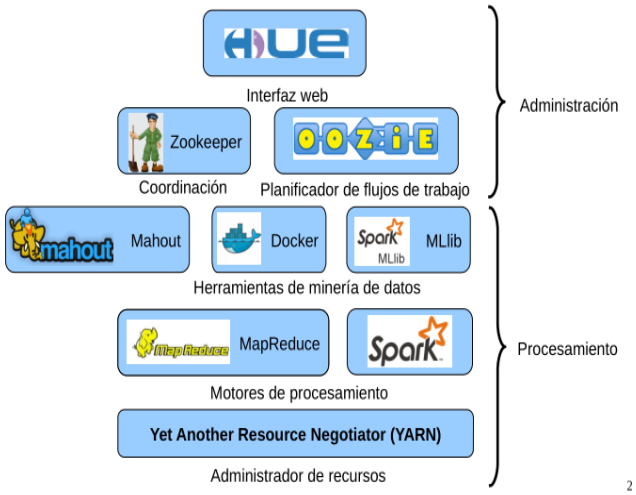

Figura 1. Plataforma para el minado de datos educativos masivos [3]

El objetivo principal de la plataforma de minado sería la estimación de conocimiento latente aplicando los algoritmos anteriormente planteados en el contexto de grandes volúmenes de datos. Cada algoritmo será montado en un clúster Hadoop y correrá sobre un contenedor Docker.

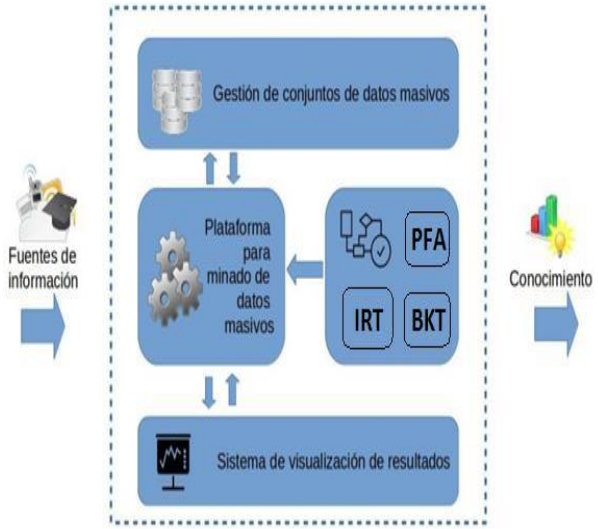

Figura 2. Esquema general de la plataforma de minado [3]

Un problema fundamental que actualmente persiste en el proceso de aplicación de los algoritmos en el contexto de datos masivos es el formato en los que aparecen los datos que se obtienen de las diferentes fuentes e-learning. De las mismas se pueden obtener información dicotómica (si responde correctamente o incorrectamente) por cada ítem evaluativo para grupos de estudiantes. A partir de estos datos de desea estimar mediante métodos estadísticos los parámetros iniciales de un ítem para que sirvan como entrada a dichos algoritmos bajo determinadas condiciones iniciales, y con esto, estimar conocimiento latente por cada estudiante. Según el modelo logístico de tres parámetros propuesto por Birnbaum en 1968, un ítem se caracteriza por tener un parámetro de dificultad, la probabilidad de ser respondido correctamente al azar y un parámetro discriminador, el cual representa la magnitud de cambio en la probabilidad de acertar el ítem, conforme varía el nivel de habilidad del individuo. [4]

Existen bibliotecas diseñadas este proceso de estimación y aplican los métodos estadísticos adecuados para ello, pero se desea que los componentes que ejecutan los algoritmos reutilicen las mismas.

Por lo planteado anteriormente se define como problema de la investigación: ¿Cómo lograr la interoperabilidad entre los componentes que ejecutan los algoritmos de estimación de conocimiento latente de forma tal que puedan reutilizar las bibliotecas de estimación de parámetros de un ítem evaluativo?

El objetivo general de esta investigación es desarrollar una capa de servicios que permita la interoperabilidad entre los componentes que ejecutan los algoritmos de estimación de conocimiento latente, para reutilizar las bibliotecas de estimación de parámetros de un ítem evaluativo.

El presente trabajo describe el proceso de desarrollo de la capa de servicios que permite reutilizar las bibliotecas de estimación de parámetros de un ítem. Se analizan las principales herramientas y tecnologías definidas para este tipo de solución y se seleccionan las más adecuadas para la propuesta de solución. Se describen los elementos de la arquitectura SOA que fueron aplicados en el desarrollo de la propuesta de solución.

2. CONTENIDO

Para el desarrollo de la propuesta de solución se proceden a analizar varias herramientas y tecnologías, de estas se analizan sus principales características y se eligen la más adecuadas para la creación de la capa de servicios, teniendo en cuenta los elementos de la arquitectura SOA y las tendencias actuales del desarrollo de servicios.

2.1 Servicios basados en REST

Para el desarrollo de la propuesta de solución se proceden a analizar varias herramientas y tecnologías, de estas se analizan sus principales características y se eligen la más adecuadas para la creación de la capa de servicios, teniendo en cuenta los elementos de la arquitectura SOA y las tendencias actuales del desarrollo de servicios.

Los servicios basados en REST (Transferencia de

Estado Representacional, por sus siglas en inglés) constituyen hoy en día una alternativa más simple a SOAP (Simple Object Access Protocol) y los servicios basados en WSDL (Lenguaje de Descripción de Servicios Web, por sus siglas en inglés). Se pueden encontrar varios casos de éxitos muchas empresas que han migrado sus servicios a estas tecnologías, tales como: Yahoo, Facebook y Google. Cabe destacar que inicialmente fue presentado por Roy Fielding en el año 2000 en una conferencia de la Universidad de California, donde presentaba principios arquitectónicos de software para usar a la web como una plataforma de procesamiento distribuido. [5]

REST define un conjunto de principios arquitectónicos por los cuales se diseñan servicios web haciendo énfasis en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes.

Características de los servicios basados en REST [5]

- **Utiliza de manera explícita métodos HTTP.** De los métodos HTTP se definen los siguientes: POST para crear un recurso en el servidor, GET para obtener un recurso, PUT para cambiar el estado de un recurso o actualizarlo y DELETE para eliminar un recurso. Constituye una característica clave de los servicios web REST es el uso explícito de los métodos HTTP, siguiendo el protocolo definido por RFC 2616, donde cada método tiene una función específica.
- **No mantiene estado.** Necesitan escalar para poder satisfacer una demanda en constante crecimiento. Se usan clusters de servidores con balanceadores de carga y alta disponibilidad, proxies, y gateways para conformar una topología de servicios, que permita transferir peticiones de un equipo a otro para disminuir el tiempo total de respuesta de una invocación al servicio web.
- **URIs con forma de directorios.** Permite crear URIs para los servicios de una forma jerárquica y organizada permitiendo mejor documentación de los servicios y que se utilicen estos de manera intuitiva.
- **Transfiere XML (Lenguaje de Marcado Extensible, siglas en inglés), JavaScript Object Notation (JSON, por su siglas en inglés), o ambos.** Se pueden usar para atender las diferentes peticiones invocadas mediante el envío de una respuesta en formato XML o JSON. Están notaciones hacen extensibles su uso y en la práctica permiten fácilmente consumir los servicios sin importar los lenguajes en las que están construidas las aplicaciones clientes.

2.2 Frameworks para la creación de servicios basados en REST

Para la selección de la herramienta a utilizar para la creación de servicios REST, se tiene en cuenta que la plataforma que se desarrollará para el procesamiento de datos masivos será un sistema distribuido que empleará tecnologías y herramientas construidas en Java; principalmente se evidencia esto en los motores de procesamiento que se emplearán, tales como: Hadoop y Spark. Por lo que lo más adecuado es emplear tecnologías con similares características y así lograr una mayor compatibilidad entre los componentes del sistema distribuido.

Para el lenguaje Java existen varios frameworks que permiten el desarrollo de servicios basados en REST, tales como: Dropwizard, Play Framework, RESTEasy, Restlet y Spring Boot. Los cuales trabajan de forma similar en el proceso de creación de servicios basados en REST y algunos de estos incluyen de forma embebida servidores de aplicaciones para Java web. De estos frameworks, se puede observar de forma general que implementan servicios REST basados en una alta modularidad, poseen buen balance de carga, son fáciles de utilizar, poseen abundante documentación y soporte. [6]

2.3 Frameworks Spring Boot para la creación de servicios REST

Spring Boot es un framework que posee las bibliotecas necesarias para la creación de aplicaciones ejecutables basadas en Spring (Framework Java basado en Modelo-Vista-Controlador). Los componentes que posee Spring Boot permiten simplificar los pasos de la selección de las dependencias necesarias para el proyecto a desarrollar y su despliegue en el servidor. No requiere de configuración usando archivos en XML y permite la creación de servicios REST en Java de manera fácil, centrando al desarrollador en los elementos críticos de desarrollo especifico de su aplicación. [7]

Características de Spring Boot [7]

- Proporciona una gama de características no funcionales que son comunes a grandes clases de proyectos (servidores integrados, seguridad, métricas, controles de estado, configuración externalizada).
- No hay generación de código y ningún requisito para la configuración XML.
- Proporciona una experiencia de inicio radicalmente más rápida y ampliamente accesible para todo el desarrollo de Spring.
- Soporta contenedores de servlets a partir

de la versión 3.0 y superior para los servidores de aplicaciones: Tomcat 8.5, Jetty 9.4 y Undertow 1.3.

Para el desarrollo de la propuesta de solución se pretende utilizar Spring Boot debido a las características de la plataforma de procesamiento que se va a desarrollar, ya que será un sistema distribuido que utilizará herramientas y tecnologías desarrolladas en lenguaje Java; esta selección previa fue analizada con anterioridad por el equipo de investigación del grupo GITAE-MDEM. Por otra parte, Spring Boot permite la creación de aplicaciones que se ejecutan en modo stand-alone al poseer de manera embebida un servidor de aplicaciones para ejecutarse de forma autónoma con solo invocar el producto desplegado usando el comando "java –jar <producto>.jar". El uso de anotaciones para la definición de los servicios y los controladores REST facilita el desarrollo rápido de los servicios aplicando un bajo acoplamiento, esto permite separar las diferentes funcionalidades y descentralizar la solución, permitiendo el desarrollo de Microservicios. Admite seguridad basada en OAuth2 y otros mecanismos para la autorización y control de seguridad de las aplicaciones web o desktop, tales como: http-basic y JSON Web Token. Por otra parte provee de forma completa los beneficios asociados al framework de desarrollo Spring MVC.

2.4 API de documentación de servicios. Spring Fox

Spring Fox es una API que permite documentar de forma rápida y sencilla servicios REST construidos con Spring Boot, se basa fundamentalmente en el uso de anotaciones para definir de forma completa cada detalle en la documentación de una determinada funcionalidad definida como un servicio. Trabaja sobre la propia configuración establecida por el framework Spring MVC y se integra fácilmente a los mecanismos de seguridad empleados. Se basa en una semántica sencilla y provee de una interfaz web extensible y personalizable para publicar la documentación de los servicios a través de Swagger. [8]

2.5 Biblioteca para la estimación de parámetros de un ítem

Para realizar el proceso de estimación de parámetros de un ítem se analizaron diferentes técnicas estadísticas y métodos numéricos aplicados en el proceso de estimación. La investigación se centra en el estudio y aplicación del método de estimación de máxima verosimilitud marginal, el cual es un método estadístico que busca ajustar un determinado modelo para estimar sus parámetros, donde la idea fundamental es tomar como estimación del parámetro estudiado el valor que haga máxima la probabilidad de obtener la muestra observada. [9]

Existe una biblioteca escrita en Java llamada Psychometrics, actualmente en su versión 1.3, que incluye funcionalidades que permiten estimar parámetros a partir de los datos dicotómicos obtenidos de un grupo de estudiantes que se han sometido a un conjunto de ítems evaluativos. Al aplicar los procedimientos que incluye la librería, se pueden obtener los parámetros de un ítem, tales como: discriminador, nivel de dificultad y probabilidad de ser contestado correctamente al azar. Estos parámetros se ajustan en consonancia con los modelos de parámetros definidos para el proceso de estimación y según cada algoritmo.

2.6 Otras tecnologías y herramientas

El desarrollo de la propuesta de solución requirió además el empleo de las siguientes tecnologías y herramientas:

- Netbeans 8.2 como entrono de desarrollo.
- Lenguaje de programación Java 8, mediante el JDK 1.8.
- Maven 3.3.9 como gestor de dependencias para aplicaciones Java.
- PostgresSQL 9.3 como gestor de bases de datos relacional.

2.7 Resultado y discusión

Para el desarrollo de la propuesta de solución se procede a crear un proyecto Maven incorporando las dependencias en el archivo de configuración pom.xml del proyecto. En este se incluyen las dependencias necesarias para el framework Spring Boot y la API de documentación Spring Fox. Se adicionan además las dependencias que utiliza la biblioteca de estimación de parámetros Psychometrics.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.2.RELEASE</version>
</parent>
```

Figura 3. Dependencia de Spring Boot con parent definido (Fuente: Elaboración propia)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
    <version>1.5.3.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security.oauth</groupId>
    <artifactId>spring-security-oauth2</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

Figura 4. Dependencia de *Spring Boot* con dependencias de seguridad basadas en OAuth2 (Fuente: Elaboración propia)

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.7.0</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.7.0</version>
    <scope>compile</scope>
</dependency>
```

Figura 5. Dependencias para la API de documentación Spring Fox con Swagger UI (Fuente: Elaboración propia)

Con esto se procede a construir el controlador de servicios REST para publicar las diferentes funcionalidades que serán consumidas por los algoritmos y el hilo principal que ejecuta la aplicación basada en Spring Boot. Se documentan los servicios mediante las anotaciones que provee Spring Fox y se personaliza la interfaz de usuario de la API mediante las anotaciones que provee Swagger UI.

A continuación de definen los servicios que provee la propuesta de solución:

a) **Listar ítems.** Permite listar los ítems almacenados en la base de datos en formato JSON, cuyos parámetros ya han sido estimados.
b) **Actualizar ítem.** Actualiza un ítem existente dado su ID y nuevos parámetros.
c) **Crear ítem.** Crea un nuevo ítem usando el modelo de tres parámetros. El modelo de tres parámetros propuesto por Birnbaum define para un ítem los siguientes parámetros: discriminador, dificultad y probabilidad de ser contestado correctamente al azar. Por cada ítem se debe suministrar un valor de respuesta dicotómico, 0 si la respuesta es correcta y 1 en caso contrario
d) **Estimar distribución latente.** Estima la distribución latente para el conjunto de ítems almacenados en la base de datos.
e) **Obtener densidad de un punto.** Calcula la densidad de un punto en la distribución latente obtenida.
f) **Eliminar ítem.** Elimina un ítem de la base de datos dado su id.
g) **Cargar dataset en formato CSV con valores dicotómicos.** Carga un dataset en formato CSV, a partir de este se estiman los parámetros basándose en el modelo de tres parámetros de Birnbaum (discriminador, dificultad y probabilidad de ser contestado correctamente al azar) y guarda el resultado de la estimación en la base de datos.
h) **Obtener un ítem dado su ID.** Obtiene los datos un ítem dado su ID y lo devuelve en formato JSON.
i) **Obtener la probabilidad de responder correctamente un ítem.** Obtiene la probabilidad de responder correctamente un ítem dado su ID y la habilidad inicial del estudiante.

2.7.1 Implementación de un servicio REST utilizando Spring Boot y la API de documentación Spring Fox

Para la implementación de servicios REST utilizando las tecnologías antes mencionadas se debe crear uno o varias clases controladoras que tendrán entre sus miembros a los métodos o funcionalidades definidas como servicios. Las clases encargadas deben utilizar anotación *@RestController*, la cual indica que la misma es un controlador REST, se puede incluir la URL principal del grupo de servicios mediante la anotación *@RequestMapping* y especificar la descripción general del grupo de servicios del controlador mediante la anotación *@Api*, esta última pertenece a *Spring Fox.*

2.7.2 Elementos de seguridad aplicados a la capa de servicio

Para el establecimiento de la seguridad de la capa de servicios fueron aplicados mecanismos que provee el protocolo OAuth2, disponible para Spring Boot. Para ello se definen filtros de seguridad, proveedores de usuarios, el tipo de algoritmo de cifrado y las rutas de control de acceso según el rol seleccionado. Luego se crea una clase que actuará como configurador de OAuth la cual heredará de *AuthorizationServerConfigurerAdapter*, con esta se definen los tiempos de expirado de las sesiones y el algoritmo de cifrado de los para la carga útil de los tokens de seguridad. [10]

Se debe tener en cuenta que OAuth2 es un framework o protocolo de autorización que no necesita mantener sesiones entre el cliente y el servidor. Su funcionamiento se basa en el envío de tokens de seguridad para los clientes que incluyen información cifrada del usuario que accede y sus permisos. Incluye cuatro roles principales, tales como Propietario de recursos, Servidor de recursos, Cliente y Servidor de autorización. [10]

2.7.3 Esquema general de la propuesta de solución

A continuación, se presenta el esquema general donde aplica la propuesta de solución:

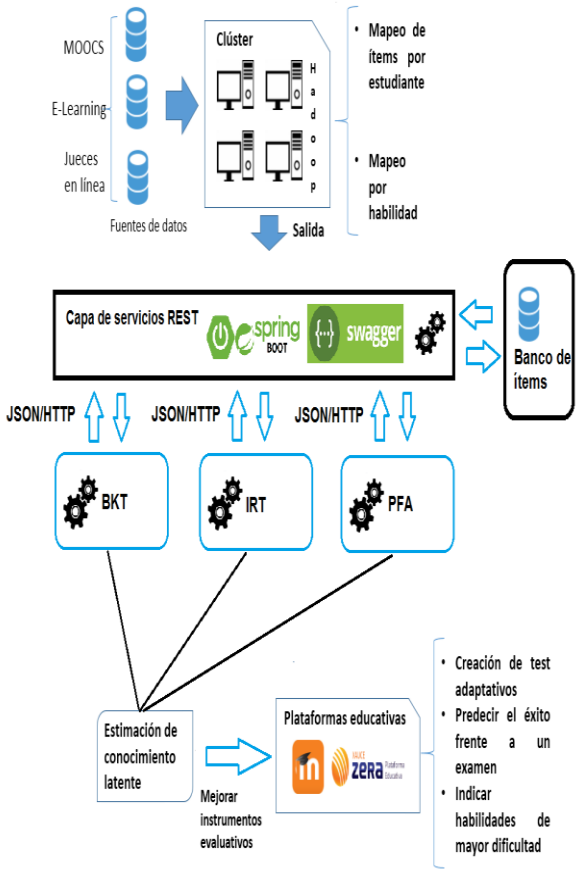

Figura 6. Esquema general de la propuesta de solución (Fuente: Elaboración propia)

Se puede observar uno de los componentes de la plataforma de procesamiento de datos masivos educativos, el cual es el clúster de Hadoop donde se reciben los datos obtenidos de los test evaluativos de diferentes plataformas, estos se procesan de forma distribuida y se almacenan en formato CSV dentro de varios HDFS (Sistemas de Archivos Distribuidos de Hadoop, por sus siglas en inglés). Mediante la capa de servicios REST los componentes que implementan los algoritmos de estimación pueden consumir los servicios de la librería de estimación de parámetros. Los resultados se almacenan en una base de datos que contiene los ítems con sus parámetros estimados. Cada ítem procesado puede ser consumido por los algoritmos y cada uno genera como salida la estimación de conocimiento latente a partir de diferentes enfoques, que son característicos de cada algoritmo. Con esto se pueden mejorar los instrumentos evaluativos mediante la predicción del éxito frente a un examen y la creación de test adaptativos en línea y se pueden indicar las habilidades de mayor dificultad para el estudiante.

3. CONCLUSIONES

Al término de este trabajo se arriban a las siguientes conclusiones:

- El estudio de las diferentes herramientas y tecnologías para la implementación de servicios basados en REST permitió elegir y aplicar Spring Boot en conjunto con Spring Fox para el desarrollo de la propuesta de solución según las características y requerimientos de los algoritmos que trabajarán sobre la plataforma de procesamiento de datos educativos masivos.
- El uso del protocolo OAuth2 permitirá incrementar los niveles de seguridad para el acceso a los servicios futuros que brindará la plataforma de minado, siendo este un mecanismo muy efectivo para garantizar el correcto control de acceso.
- La implementación de una capa de servicios que provee de funcionalidades para la estimación de parámetros de un ítem permitirá reutilizar los elementos necesarios que requieren los algoritmos en sus entradas para la posterior estimación de conocimiento latente.

4. REFERENCIAS BIBLIOGRÁFICAS

1. Vera, Macarena. 2014. ¿Qué se entiende por SOA, y cuáles son sus beneficios? Intelligence to Business. [En línea] 2014. [Citado el: 9 de octubre de 2017.] http://www.i2btech.com/blog-i2b/tech-deployment/que-se-entiende-por-soa-y-cuales-son-sus-beneficios/.

2. J. A. Larusson, B. White. 2014. Learning Analytics. From research to practice. United States: s.n., 2014. ISBN 978-1-4614-3304-0.

3. GITAE-MDEM. 2016. Plataforma de minado de datos educativos masivos. La Habana: s.n., 2016.

4. Pérez Gil, José Antonio. 2004. Modelos de Medición: Desarrollos actuales, supuestos, ventajas e inconvenientes. Universidad de Sevilla: s.n., 2004.

5. Seta, Leonardo De. 2008. Introducción a los servicios web RESTful. Dos Ideas. [En línea] 18 de 11 de 2008. https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful.

6. Gajotres. 2017. Top 8 Java RESTful Micro Frameworks – Pros/Cons. Gajotres. [En línea] 31 de 10 de 2017. https://www.gajotres.net/best-available-java-restful-micro-frameworks/3/.

7. Phillip Webb, et. al. 2017. Spring Boot Reference Guide. 2017.

8. SpringFox. 2017. Springfox Reference Documentation. SpringFox. [En línea] 2017. [Citado el: 1 de 11 de 2017.] https://springfox.github.io/springfox/docs/current/.

9. Universidad de Barcelona. 2017. Estimación de máxima verosimilitud. Universidad de Barcelona. [En línea] 2017. [Citado el: 1 de 11 de 2017.] http://www.ub.edu/stat/GrupsInnovacio/Statmedia/demo/Temas/Capitulo7/B0C7m1t10.htm.

10. Martin, et al. 2017. Spring Boot REST API (4) - Security with OAuth2. Gisterius Blog. [En línea] 1 de 3 de 2017. https://gigsterous.github.io/engineering/2017/03/01/spring-boot-4.html.

5. SÍNTESIS CURRICULARES DE LOS AUTORES

Acerca de Orlando Grabiel Toledano López: De nacionalidad cubana y nacido el 18 de octubre de 1991, en el municipio Holguín de la provincia Holguín. Cursó estudios superiores en la Universidad de las Ciencias Informáticas en el período 2010-2015, donde alcanzó el título de Ingeniero en Ciencias Informáticas, realizó su adiestramiento como profesor del Departamento de Programación de la Facultad 4, impartiendo clases en dicha disciplina. Obtuvo la categoría docente principal de Profesor Instructor en el año 2016 y actualmente se desempeña como investigador en el grupo de investigación GITAE-Minería de Datos Educativos Masivos.