UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Capa de servicios web para el Sistema de Gestión para el Ingreso a la Educación Superior

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Roberto Verdecia Sánchez

Tutores:

Ing. Jorge Luis Piña González.

Ing. Arletys González Madera

“Año 60 de la Revolución”

Ciudad de la Habana, 28 de junio de 2018

Declaración de autoría

Declaro ser único autor del presente trabajo “Capa de servicios web para el Sistema de Gestión para el Ingreso a la Educación Superior”, y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas, a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de ___ del año ___.

Roberto Verdecia Sánchez

_____

Firma del autor

Ing. Jorge Luis Piña González

_____

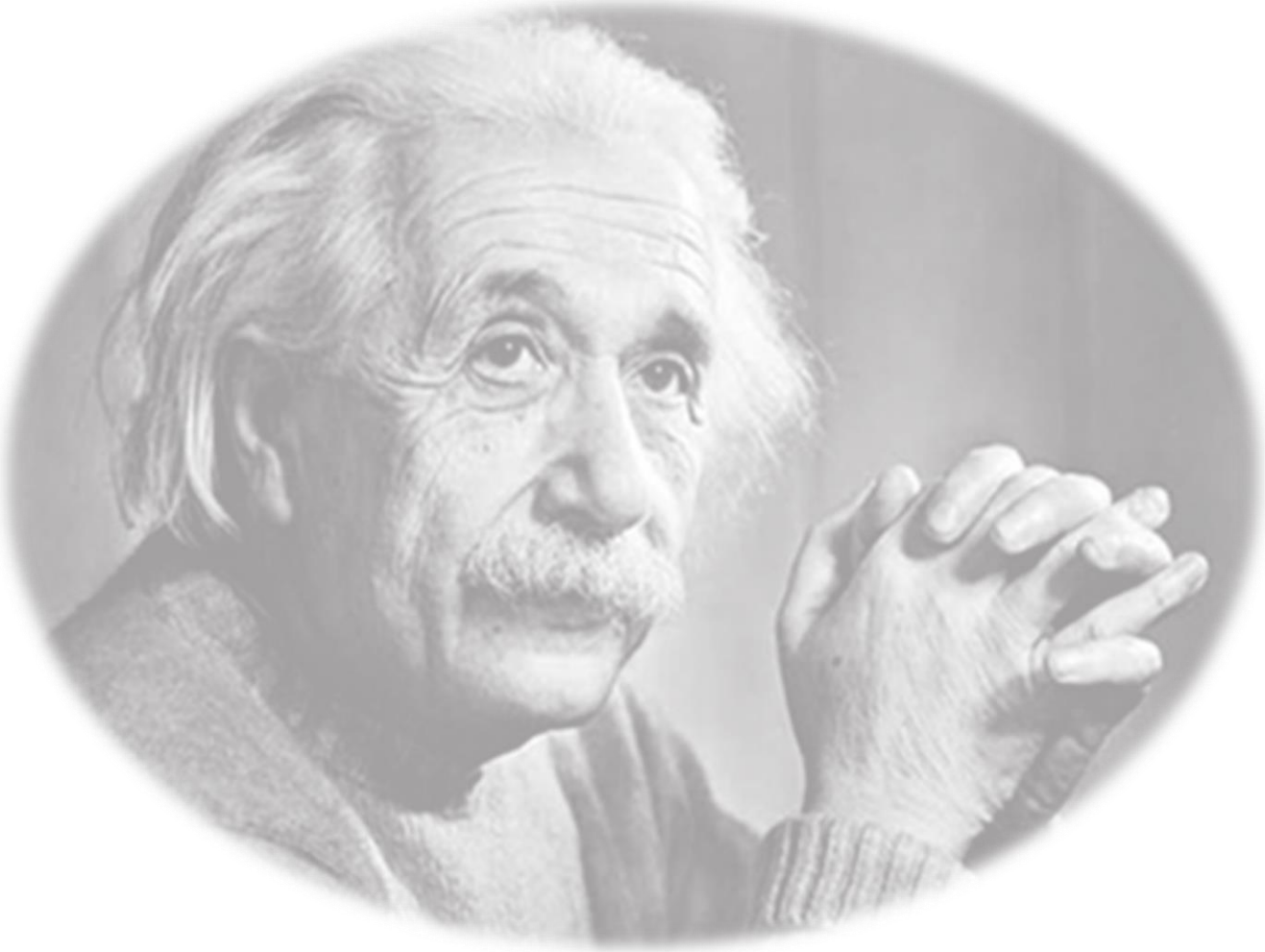
Firma del tutor

Ing. Arletys González Madera

_____

Firma del tutor

Pensamiento



"Todos somos ignorantes, lo que ocurre es que no todos ignoramos las mismas cosas".

Albert Einstein

Agradecimientos

Sin lugar a duda el primero de mis agradecimientos es para Dios, pues se que mi madre y mi abuela que son las dos mujeres másimportante es mi vida están de acuerdo con mi decisión de darle el primer lugar a Él.

Agradecer a mi mamá pues aunque no me ayudó a programar me ayudó a no desmayar durante el proceso de la tesis. Eso vale más.

Agradecer a mi abuela por su amor incondicional a su nieto. Es lo mejor en calidad de abuela.

Agradezco a mi padre por su ayuda en el proceso.

Agradezco a Lili por su compromiso conmigo, pues ella me demostró que realmente me considera su hermano, pues sin tener compromiso, me ayudó y alentó en el proceso de tesis.

Agradezco de manera especial a Jose, Alejandro, Robin, Daniel, Josué, Mariam, Sobrino, Osmayli y Mariela por su ayuda y consejos.

Agradezco a mis tutores por su sacrificio, empeño y preocupación por mi en el momento crítico del proceso de tesis. A Arletys pues que me hubiese hecho sin ella para revisar lo más que pude el docuento, gracias por no almorzar el día 21, y también al profe Piña por sus consejos sobre la aplicación.

Agradezco a todos los amigos que logré aca en la Universidad tanto cristianos como no cristianos, por haberme enseñado algo en la vida o darme la oportunidad de aconsejarles. Aprendí que no lo sabemos todo, que ser humilde es más provechoso que la abundancia de conocimiento.

Agradezco a mis amigos de Niquero por su ayuda en todos estos 5 años de carrera. Especialmente a Ennier, Gerar, Alialys, Yami y mi pastor Yimi.

Agradezco a mis compañeros de aula por soportar mis charlas cristianas en los turnos, y respetar mis postura sobre las cosas. Se que les hice bien, ya lo verán. De manera especial a los 4 fantásticos: Leo, Randy, Nurin y Yancel, pues este año todos vivimos juntos el proceso de tesis.

Agradezco de forma especial a Haydeé pues le prometí que estaría en mis agradecimientos pues fue de gran ayuda el dia 10 de junio cuando me cantó una bella canción que me decía que todo iba a salir bien.

Agradezco a Magalis por su amistad, también le prometí que estaría en mis agradecimientos.

Dedicatoria

Dedico el presente trabajo a Dios pues ciertamente pensé que no lo lograría. Pasé muchos días de angustias donde solo Él fue mi aliento.

Dedico mis esfuerzos a mi madre y mi abuela por su sacrificio, que sería de mi sin estas dos bellezas. Quizas no son "Mis universo" pero si las reinas de mi mundo.Además a mi padre.

Dedico mi tesis a mis amigos pues quiero que sientan orgullo de lo que he logrado son importar una nota. Dicen que vale más el resultado que el esfuerzo, y es relativo en el punto que a veces en la vida no se optienen cosas que se quieren, pero si se optienen experiencias sobre como conducirse en el futuro. Sin importar el resultado "Me esforcé", eso me deja tranquilo. Me gradué teniendo pasión por otras materias, música, pintura y teatro.

Resumen

El Centro de Tecnologías para la Formación, enmarcado dentro de la Universidad de las Ciencias Informáticas posee, dentro de su misión, el desarrollo de software destinados a la educación. Con tal premisa, en el año 2015 se desarrolló el Sistema de Gestión para el Ingreso a la Educación Superior, que gestiona los datos recopilados durante el periodo de ingreso de los estudiantes a la Educación Superior. La no existencia de una vía de intercambio de datos entre aplicaciones externas al negocio y el sistema gestor, a través de la red, imposibilita la obtención de información referente a estudiantes, asignación de carreras, plazas de ingreso, y otros datos, provocando que el proceso sea realizado manualmente a través de la exportación e importación de ficheros XML. El propósito de la presente investigación consiste en la implementación de una capa de servicios web que permita establecer la comunicación entre SIGIES y terceros. Esta comunicación posibilita que los datos sean accesibles a clientes del sistema, el cual está implementado bajo el protocolo de autorización Oauth2, garantizando la seguridad de la capa de servicios. El estudio de sistemas que afontaron este tipo problemas permitió determinar las tendencias y mejores prácticas para el desarrollo de una capa de servicios, además de una correcta selección de herramientas y tecnologías. Para guiar el proceso de desarrollo se hace uso de la metodología AUP-UCI. La realización de pruebas al software desarrollado, servieron para validar la solución propuesta.

Palabras claves: arquitectura, capa de servicio, servicios web.

Índice

Índice

Índice

Introducción

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC), constituyen uno de los factores más decisivos en el desarrollo actual de los países, pues son numerosos los proyectos que sustentan su potencial en la utilización de dichas tecnologías. Este auge ha influenciado en gran medida la rama de la informática a raíz de la propia necesidad del hombre de informatizar los procesos de la vida cotidiana. Su comienzo en dicha rama fue con aplicaciones monousuarios implementados en grandes ordenadores. Posteriormente estas aplicaciones alcanzaron la capacidad de atender a diferentes usuarios dando lugar al concepto multiusuario. Con el correr de los años nació el modelo de desarrollo cliente-servidor que dividía la aplicación en dos: Una parte que interaccionaba con el usuario y otra parte destinada al procesamiento de la información. En este acercamiento se consiguió que cada una de las partes que constituían la aplicación pudiera residir en computadoras distintas. Actualmente el desarrollo de la computación ha llegado al concepto de aplicaciones distribuidas, en las cuales los procesos se realizan en diferentes unidades de porcesamiento. (Gil Aros, 2009)

Los servicios web son un paso adelante en la computación ya que de esta forma un ordenador no esconsiderado un núcleo de cómputo, sino un repositorio de servicios de un gran número de aplicaciones distribuidas en diferentes lugares geográficos. La alta disponibilidad y accesibilidad de la información en el mundo actual, ha abierto el espectro a un conjunto de aplicaciones orientadas al manejo de la enorme cantidad de datos que son generados a diario (Gil Aros, 2009). La puesta en práctica de estos avances en el sector educacional ha propiciado que las universidades posean un mejor manejo de la información y gestión de datos referentes a sus estudiantes.

Cuba no ha estado exenta de este fenómeno, muchos han sido los esfuerzos del gobierno encaminados a dotar al país de la infraestructura necesaria para la implementación de los procesos relacionados con la gestión y automatización de la información. Con este fin se ha creado un plan de informatización que incluye a cada uno de los sectores de la población (Guevara, 2016). Las universidades cubanas constituyen un eje importante en la sociedad ya que son instituciones innovadoras de excelencia científica, académica y productiva que forma profesionales altamente calificados, con valores patrios y un profundo sentido humanista reflejado en la formación integral del profesional (MES, 2015).

Dentro de las instituciones del Ministerio de Educación Superior (MES) se encuentra la Universidad de las Ciencias Informáticas (UCI), organización vanguardia en la rama de la informática. Su misión fundamental es producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación, y servir de soporte a la industria cubana de la informática (UCI, 2012).

El Centro de Tecnología para la Formación (FORTES), tiene dentro de sus encomiendas el desarrollo de software destinados a la educación. En 2015, este centro dio inicio al proyecto Sistema de Gestión para el Ingreso a la Educación Superior (SIGIES), con el fin de realizar las tareas de recopilación y manejo de los subprocesos de Organización, Exámenes, Asignación y Otorgamiento comprendidos en este proceso**.** Existe la necesidad de que algunos datos almacenados en SIGIES sean consumidos por

sistemas externos, como es el Sistema de Gestión de la Nueva Universidad (SIGENU), el cual cuenta con varios módulos desplegados en las instituciones universitarias del país. SIGENU se encarga de gestionar la información de los estudiantes desde que matriculan hasta que se gradúan o causan baja de una institución.

Otra plataforma con la cual el sistema debe intercambiar sus datos es Inteligencia de Negocio para Estudios Sociales (INPES), el cual está orientado al análisis y la toma de decisiones. Actualmente está en desarrollo una aplicación androide para SIGIES, la cual necesitará consumir datos del mismo para responder a las funcionalidades que se le implementarán. La ausencia de un mecanismo que garantice la interoperabilidad entre SIGIES y otras aplicaciones informáticas trae como consecuencia que el alcance del uso de los datos del sistema esté limitado a un área específica.

Actualmente todas las funcionalidades del SIGIES son completamente inaccesibles para cualquier sistema automatizado, tal es el caso del SIGENU, el INPES y la aplicación androide mencionada anteriormente. Debido a esto el sistema queda aislado del auge que toman las tecnologías en el uso de las aplicaciones móviles; se restringe el impacto social del sitio a nivel nacional y se pierde la oportunidad de hacer que más usuarios conozcan y tengan acceso a algunas funcionalidades del sistema.

Siendo esta la principal problemática se plantea como **problema de investigación:** ¿Cómo garantizar la interacción y el intercambio de información entre el SIGIES con aplicaciones informáticas de terceros?

A partir de la situación problémica descrita se puede inferir como **objeto de estudio:** la interoperabilidad entre sistemas informáticos, enmarcado en el **campo de acción:** proceso de desarrollo de servicios web en SIGIES.

Para dar solución a la interrogante planteada en el problema de investigación se define como **objetivo general:** Desarrollar una capa de servicios web para garantizar la interoperabilidad de aplicaciones informáticas de terceros con el SIGIES.

Para guiar la investigación se definen las siguientes **preguntas científicas.**

- ¿Cuáles son los fundamentos teóricos que se necesitan para lograr que la capa de servicios Web sobre SIGIES cumpla los objetivos propuestos?
- ¿Cómo guiar el diseño de la propuesta de solución a partir de la descripción de la propuesta de solución?
- ¿De qué manera se implementará la capa de servicios que permita extender los procesos de SIGIES a terceros?
- ¿Qué tipos de pruebas de software se pueden utilizar en la validación de la propuesta de solución?

A partir del objetivo general definido se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico-conceptual referente a las tecnologías empleadas para el desarrollo de una solución que garantice el intercambio de información entre sistemas informáticos.
- Realizar el análisis y diseño de la capa de servicios web de SIGIES.

- Implementar las funcionalidades de la propuesta de solución.
- Validar la propuesta de solución mediante pruebas de software para verificar el correcto funcionamiento del sistema.

Con el fin de resolver el problema de la investigación y dar cumplimiento a los objetivos planteados se trazaron las siguientes **tareas de investigación:**

1. Realización de un estudio del estado del arte a nivel internacional, nacional y local, para conocer los referentes teóricos del objeto de estudio.
2. Realización del estudio de la metodología de software a emplear en el desarrollo de la solución.
3. Selección de las tecnologías y herramientas de desarrollo adecuadas para la implementación de la capa de servicios web.
4. Elaboración del modelo de dominio o modelo conceptual del sistema.
5. Identificación de los requisitos funcionales y no funcionales de la capa de servicios web para SIGIES.
6. Realización del modelado de análisis del sistema que se desea desarrollar.
7. Caracterización y selección de los patrones de diseño necesarios para la correcta implementación de la capa de servicios web.
8. Realización del modelado del diseño de todas las funcionalidades identificadas.
9. Implementación de la capa de servicios web de SIGIES.
10. Validación de la propuesta de solución para validar el correcto funcionamiento de la capa de servicios web desarrollada.

Para lograr los objetivos se definen como **posibles resultados:**

- Capa de servicios web que permita el acceso de sistemas externos a la información almacenada en el Sistema de Gestión para el Ingreso a la Educación Superior.
- Documentación asociada el proceso de investigación y los artefactos ingenieriles generados, de manera que pueda continuarse con el mantenimiento de este, así como el desarrollo de futuras versiones.

Para el desarrollo de la investigación se emplearon los siguientes métodos teóricos y empíricos:

Métodos teóricos:

Analítico-sintético: se hace uso de este método para la identificación de conceptos dentro del campo de los servicios web, analizando documentos, para la extracción de los elementos más importantes sobre el tema en cuestión.

Histórico-Lógico: utilizado para analizar la evolución histórica de soluciones similares y las tendencias más recientes sobre la utilización de servicios web en sistemas informáticos, para concluir qué aspectos son necesarios en el desarrollo de la solución que se propone.

Modelación: se emplea en la creación de diagramas para representar el proceso de desarrollo mediante el lenguaje de modelado UML, para reflejar la estructura, relaciones internas y características de la solución.

Métodos empíricos:

Análisis documental: este método es usado en el análisis de documentos y conceptos que apoyan la investigación certificando los más adecuados para la misma.

Entrevista: se utiliza para recoger la información tanto primaria como secundaria que tributa al desarrollo de la capa de servicios para SIGIES. Permite una conversación planificada para obtener información. Se realiza al ingeniero: Jorge Luis Piña, jefe de proyecto de SIGIES persiguiendo los siguientes objetivos:

1. Especificar el problema de investigación y alcance de la capa de servicios web para SIGIES.
2. Definir el objetivo general de la Capa de Servicios web para SIGIES.
3. Identificar los procesos de SIGIES que serán llevados a la capa de servicios web.
4. Determinar los requisitos funcionales.

El presente trabajo se estructura en tres capítulos, en los que se encuentra el contenido distribuido de la siguiente manera:

Capítulo 1 Fundamentación teórica: en este capítulo se exponen los elementos teóricos que sustentan la investigación. Se realiza un estudio de las soluciones existentes a problemas similares. Además, se hace un análisis para seleccionar en base al estudio realizado, la metodología, las herramientas y las tecnologías adecuadas para la implementación de la capa de servicios web para el SIGIES.

Capítulo 2 Descripción de la solución propuesta: en este capítulo se determinan los requisitos funcionales y no funcionales que debe cumplir el sistema para un correcto funcionamiento. Se construyen los artefactos correspondientes al análisis y diseño, definiendo la estructura de la capa de servicios web para el SIGIES.

Capítulo 3 Implementación y prueba: en este capítulo se describen los elementos fundamentales desarrollados en la etapa de implementación y los resultados de las diferentes pruebas de software, con el objetivo de garantizar la calidad del mismo.

Capítulo 1. Fundamentación teórica

En el presente capítulo se abordarán los conceptos fundamentales relacionados con los servicios web, se enfatizará en las características de la metodología utilizada, así como el estado del arte. Además, se incluirán las tecnologías y herramientas que serán empleadas en el desarrollo de la capa de servicios web.

1.1 Conceptos asociados al dominio del problema

Para la correcta comprensión del problema y su solución es necesario establecer y analizar los principales conceptos y definiciones asociadas al dominio del problema.

Arquitectura Orientada a Servicios

SOA (*Service Oriented Architectura* por sus siglas en inglés), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos de software del usuario. Este tipo de arquitectura permite la creación o modificación de los procesos de negocio desde la perspectiva de Tecnología de Información de forma ágil, a través de la composición de nuevos procesos utilizando las funcionalidades de negocio que están contenidas en la infraestructura de aplicaciones actules o futuras (Gutiérrez, y otros, 2009).

SOA define las siguientes capas de software (Gutiérrez, y otros, 2009):

- Aplicaciones básicas: sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- Exposición de funcionalidades: las funcionalidades de la capa aplicativa son expuestas en forma de servicios.
- Integración de servicios: facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.
- Composición de procesos: define que el proceso en términos del negocio y sus necesidades varía en función del negocio.
- Entrega: los servicios son desplegados a los usuarios finales.

Los beneficios que se pueden tener al adoptar *SOA* son (Gutiérrez, y otros, 2009):

- Mejora en los tiempos de realización de cambios en procesos.
- Facilidad para evolucionar a modelos de negocios basados en tercerización.
- Facilidad para abordar modelos de negocios basados en la colaboración con otros entes (socios, proveedores).
- Poder para reemplazar elementos de la capa aplicativa *SOA* sin disrupción en el proceso de negocio.
- Facilidad para la integración de tecnologías disímiles.

Servicio web

La *World Wide Web Consortium*[1] lo define como "un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red". (Consortium, 2004).

Un servicio web es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. (Gil Aros, 2009).

IBM[2] define servicio web como el término que designa una tecnología que permite que las aplicaciones se comuniquen en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML[3] estandarizada. Usa protocolos basados en el lenguaje XML con el objetivo de describir una operación para ejecutar o datos para intercambiar con otro servicio web. Un grupo de servicios web que interactúa de esa forma define la aplicación de un servicio web específico en una arquitectura orientada a servicios (SOA) (IBM, 2014).

Un servicio web es una vía para compartir información entre aplicaciones a través de la red sin importar la plataforma en que estén implementadas.

Interoperabilidad

La interoperabilidad es una característica esencial para arquitecturas de información enlazadas para trabajar en entornos heterogéneos y a lo largo del tiempo. Sin embargo, emplear y entender el concepto es todavía muy diverso: la interoperabilidad está concebida en una relación con objetos o en una perspectiva funcional, en términos de multilingüismo o de significados técnicos y protocolos. Además, está concebida en diferentes niveles de abstracción: desde la capa de flujo de bits hasta la de tipo semántica (Gradmann, 2009).

El Vocabulario de Información y Tecnología ISO/IEC 2382 define interoperabilidad como "*la capacidad de comunicar, ejecutar programas, o transferir datos entre varias unidades funcionales de forma que el usuario no tenga la necesidad de conocer la característica única de estas unidades*" (ISO, 2000).

Orientación a servicios

Según Thomas Erl, los principios que rigen la orientación a servicios son:

- Los servicios deben ser reusables: todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación.
- Los servicios deben tener bajo acoplamiento: los servicios tienen que ser independientes los unos de los otros.

[1] Principal organización de normalización del *World Wide Web* (WWW).
[2] Máquinas de negocios internacionales o International Business Machines por sus siglas en inglés.
[3] Lenguaje de Marcado Extensible o Extensible Marking Language por sus siglas en inglés.

- Los servicios deben permitir la composición: todo servicio debe ser construido de tal manera que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel.
- Los servicios deben de ser autónomos: todo servicio debe tener su propio entorno de ejecución.
- Los servicios deben poder ser descubiertos: todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades (Erl, 2014).

Formatos para el intercambio de datos entre aplicaciones

XML: Es un formato de texto flexible derivado de SGML[4]. Originalmente diseñado para afrontar los retos de la gran edición electrónica, está desempeñando un papel cada vez más importante en el intercambio de una amplia variedad de datos en la *web* y otros lugares (W3C, 2008).

Ha sido regularmente criticado por su nivel de detalle y complejidad. El mapeo del modelo de árbol básico de XML hacia los sistemas de tipos de lenguajes de programación o bases de datos puede ser difícil, especialmente cuando se utiliza para el intercambio de datos altamente estructurados entre aplicaciones, lo que no era su objetivo primario de diseño. XML es difícil de parsear y de leer, su modelo de datos es incompatible con la mayoría de los modelos de datos de los lenguajes de programación (Lin, y otros, 2012).

JSON[5]: Es un formato ligero para la serialización e intercambio de datos entre sistemas y tecnologías. Describe la información con una sintaxis dedicada que se usa para identificar y gestionar datos. Está basado en un subconjunto del lenguaje de programación *JavaScript* que permite representar estructuras de datos (arreglos) y objetos (arreglos asociativos) en forma de texto. Una de las ventajas que tiene el uso de JSON es que puede ser leído por varios lenguajes de programación como es el caso de *C, C++, Java, JavaScript, Perl, PHP* y *Python*. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías (International, 2013).

A continuación se muestra una tabla comparativa que resalta las principales ventajas y desventajas entre JSON y XML.

Tabla 1 Comparación entre JSON y XML

	XML	**JSON**

[4] Lenguaje de marcado generalizado estándar o Standard Generalized Markup Language pos sus siglas en inglés.
[5] Notación de objetos JavaScript o JavaScript Object Notation pos sus siglas en inglés.

Ventajas	Es muy simple (más que SGML). Es totalmente interoperable. Fácil de leer para el humano. Posee un lenguaje de marcado muy extensible. Maneja estándares. Soporta cualquier tipo de dato clásico y otros (multimedia, ejecutables). Validacion incluida (esquema XSD[6]).	Es más simple aún (que XML). Al igual que XML es interoperable (basado en texto). Fácil de leer para el humano y mucho más fácil para la máquina. Preferida por los programadores. Posee convención simple lo que lo hace más liviano. Complejidad de analizador bajo (función javascript eval()).
Desventajas	Poco elegida por programadores. Utiliza documentos de esquemas lo que lo hace más pesado. Complejidad de analizador alto.	No tiene lenguaje de marcado por lo tanto no es extensible. No maneja estándares. No soporta envío de grandes cargas, solo datos comunes. Validación a través de agentes externos.

A continuación se muestra una gráfica que ilustra el uso de XML y JSON en API (Trends, 2018).

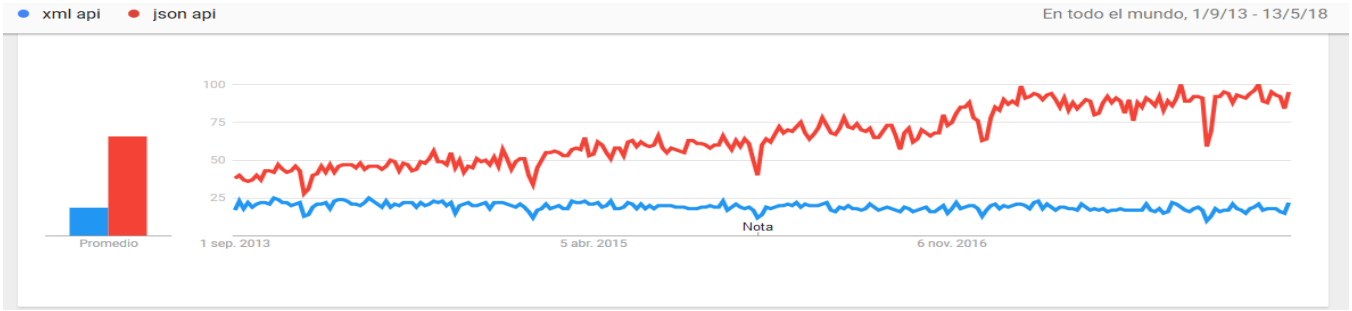


Figura 1 Gráfica ilustrativa del uso de XML y JOSN

Teniendo en cuenta que XML posee una estructura de datos más compleja que JSON, lo que dificulta el proceso de parsear el código, y que JSON es más ligero, se decide utilizar JSON como formato de intercambio de datos.

[6] XML Schema Definition.

Protocolos de comunicación para servicios web

SOAP[7]: Es un protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes de computadoras. Permite utilizar lenguajes de alto nivel para llamar e implementar el servicio web. Es básicamente un paradigma de una sola vía pero con la ayuda de las aplicaciones se puede llegar a crear patrones más complejos. Está constituido por: un marco que describe el contenido del mensaje e instrucciones de proceso, un conjunto de reglas para representar los tipos de datos definidos, convenciones para representar llamadas a procedimientos remotos y respuestas (Weerawarana, y otros, 2005).

Este protocolo proporciona el marco por el cual la información específica de la aplicación puede ser transmitida de una manera extensible. Ofrece una descripción completa de las acciones necesarias tomadas por un nodo SOAP al recibir un mensaje SOAP. Es un protocolo basado en XML que consta de:

- Un marco para describir el contenido del mensaje y las instrucciones de proceso.
- Un conjunto opcional de reglas de codificación para representar tipos de datos definidos.
- Una convención para representar llamadas a procedimientos remotos y respuestas (PUTTE, y otros, 2004).

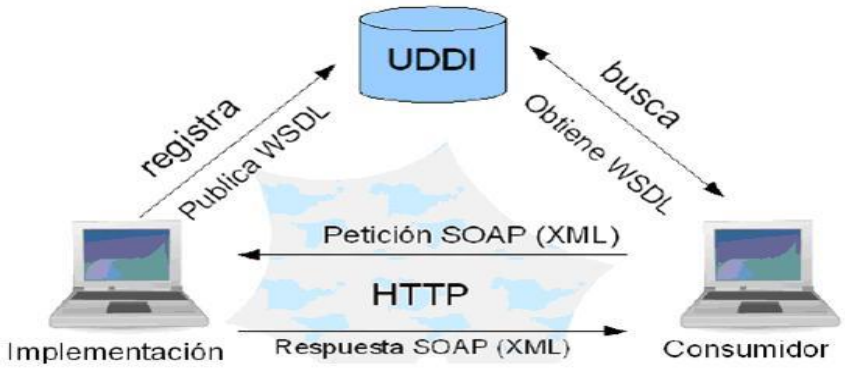


Figura 2 Funcionamiento de SOAP (Weerawarana, y otros, 2005)

Los servicios basados en **REST**[8], constituyen hoy en día una alternativa más simple a SOAP y los servicios basados en WSDL[9]. Se pueden encontrar varios casos de éxitos teniendo en cuenta que muchas empresas que han migrado sus servicios a estas tecnologías, tales como: *Yahoo*[10], *Facebook*[11] y *Google*[12]. Inicialmente, este protocolo 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 (Seta, 2008). REST es un estilo de arquitectura para el diseño de aplicaciones en red. La idea es que, en lugar de utilizar los mecanismos

[7] Simple Object Access Protocol: Protocolo de Acceso de Objetos Simples.
[8] Representational State Transfe: Transferencia de Estado Representacional.
[9] Web Services Description Language: Lenguaje de Descripción de Servicios Web.
[10] Yahoo es una empresa global de medios con sede en Estados Unidos que posee un portal de Internet, un directorio web y una serie de servicios tales como el popular correo electrónico Yahoo.
[11] Facebook: compañía estadounidense que ofrece servicios de redes sociales y medios sociales en línea con sede en Menlo Park, California.
[12] Google: compañía principal subsidiaria de la multinacional estadounidense Alphabet Inc., cuya especialización son los productos y servicios relacionados con Internet, software, dispositivos electrónicos y otras Tecnologías.

complejos, tales como RPC[13] o SOAP para la conexión entre máquinas, se utiliza HTTP para hacer llamadas entre las máquinas (Catalani, 2012).

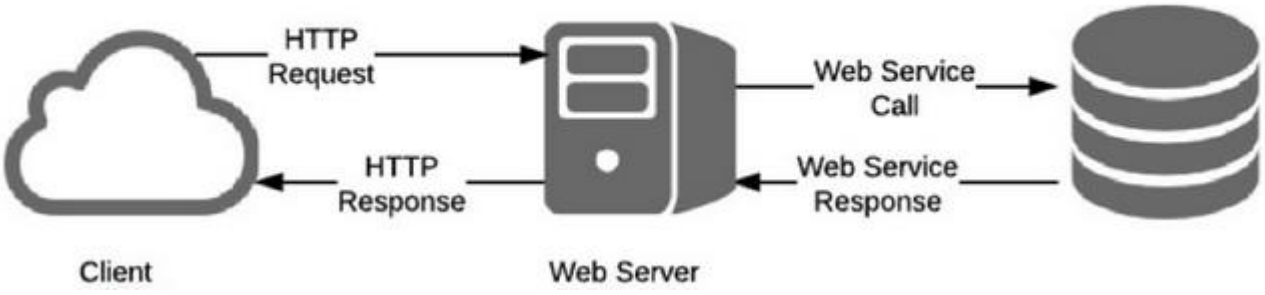


Figura 3 Funcionamiento de REST

A continuación, se muestra una tabla que relaciona algunos aspectos referentes a los protocolos de comunicación para el desarrollo de servicios, con el fin de seleccionar uno acorde a la capa de servicios de SIGIES.

Tabla 2 Comparación entre REST y SOAP

	REST	**SOAP**
Características principales	Las operaciones se definen en los mensajes.	Las operaciones son definidas como puertos WSDL.
	Una dirección única para cada instancia del proceso.	Dirección única para todas las operaciones.
	Cada objeto soporta las operaciones estándares definidas.	Múltiples instancias del proceso comparten la misma operación.
	Componentes débilmente acoplados.	Componentes fuertemente acoplados.
Tecnología	Pocas operaciones con muchos recursos.	Muchas operaciones con pocos recursos.
	Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia.	Se centra en el diseño de aplicaciones distribuidas.
Descripción del servicio	Confía en documentos orientados al usuario que define las direcciones de petición y las respuestas.	WSDL.
	Interactuar con el servicio supone horas de	Se pueden construir automáticamente

[13] Remote Procedure Call : Llamada a Procedimiento Remoto.

	[14]testeado y depuración de URLs.	stubs (clientes) por medio del WSDL.
	WADL[15] propuesto en noviembre de 2006.	WSDL 2.0.
Seguridad	HTTPS[16].	WS-Security.
Metodología de diseño	Identificar recursos a ser expuestos como servicios.	Listar las operaciones del servicio en el documento WSDL.
	Definir URLs para direccionar los servicios.	Definir un modelo de datos para el contenido de los mensajes.
	Distinguir los recursos de solo lectura (GET) de los modificables (POST, PUT, DELETE).	Elegir un protocolo de transporte apropiado y definir las correspondientes políticas de seguridad.
	Implementar e implantar el servidor Web.	Implementar e implantar el contenedor del Servicio Web.

Después de haber analizado las características, tecnología y seguridad se decide usar REST, debido a la escalabilidad de interacción con los componentes y la generalidad de interfaces a través de HTTP. Para dicha selección se tuvo en cuenta además, que REST puede hacer uso de los dos lenguajes de intercambio de datos más conocidos entre desarrolladores: JSON y XML, no siendo asi con SOAP. Además, REST ofrece la ventaja de separar cliente/servidor y permite desarrollar en cualquier tipo de tecnología o lenguaje que se corresponda con la filosofía o necesidades del proyecto, garantizando de esta manera, minimizar el tiempo de desarrollo.

1.2 Estudio del estado del arte

El número de sistemas informáticos que publican sus funcionalidades a través de servicios web crece de manera vertiginosa en la actualidad. Este tipo de avance surge como solución a la antigua forma de intercambio de información entre sistemas o aplicaciones. La comunicación a través de servicios web, sustituyó el proceso de exportación e importación al que eran sometidos los datos. La evolución de dichos servicios teniendo en cuenta la rapidez, estructura del código y ligereza en el proceso de parseo, continuó avanzando según lo hacían las tecnologías empleadas. Inicialmente SOAP constituía el protocolo de comunicación usado, el cual definía XML como lenguaje de intercambio de datos, y aunque no quedó obsoleto, REST por su flexibilidad, simpleza y ligereza en el envio y recepción de datos, tomó su lugar, constituyendo en la actualidad la tecnología de preferencia en el marco del desarrollo de

[14] Uniform Resuorce Locator : Localizador de Recursos Uniforme
[15] Web Application Description Language: Lenguaje de Descripción de Aplicaciones Web.
[16] Hyper Text Transfer Protocol Secure: Protocolo Seguro de Transferenca de Hipertexto.

servicios web. A continuación se realiza un estudio sobre sistemas que implementan una capa de servicio como solución a la necesidad de intercambiar información con terceros.

Facebook

Desde su fundación en 2004, la misión de *Facebook* es ofrecer a las personas la posibilidad de crear comunidad y acercar el mundo. Dos años después de su fundación, el 1ro de agosto de 2006, *Facebook* presenta la primera versión de su *API*[17], con el objetivo de facilitar la comunicación de la plataforma con otras aplicaciones interesadas en interactuar con la red social (Facebook, 2018).

La API de Facebook es la forma principal de ingresar y acceder a los datos contenidos en la plataforma. Está basada en *HTTP*, permitiendo que aplicaciones puedan usar de forma programática la consulta a datos, publicación de historias, subida de fotos y otras operaciones. La información representada en la plataforma social se compone por: nodos, perímetros y campos. Normalmente, los nodos se usan para obtener datos sobre un objeto específico, los perímetros, para obtener colecciones de objetos a partir de un objeto único, y los campos, para la obtención de datos sobre un objeto único o todos los objetos en una colección (Facebook, 2018).

Otra de las características que distinguen esta *API* es que las solicitudes requieren un token de acceso. Los tokens de acceso siguen el protocolo OAuth 2.0, que permite que las entidades como los usuarios o las páginas les brinden autorización. Por lo general, esto se realiza a través de la web. Una vez que reciben la autorización, las aplicaciones pueden usar los tokens para acceder a información específica sin necesidad de proporcionar su contraseña. El uso de tokens permiten además identificar la aplicación, el usuario que la utiliza y el tipo de datos a los que estos pueden acceder (Facebook, 2018).

COJ

Como resultado de la integración de la UCI al movimiento ACM-ICPC (ACM-Asociación de los Sistemas Informáticos, ICPC-Competición Internacional Universitaria de Programación), surge el Juez en Línea Caribeño (COJ) con el fin de probar, mejorar y compartir habilidades en la resolución de problemas, la programación de computadoras y el trabajo en equipo, además de obtener una fuerte capacitación para participar en competiciones de programación a nivel internacional. En el año 2016 se implementó una capa de servicios web que permitió establecer la comunicación de aplicaciones informáticas con el COJ. Esta comunicación posibilitó que cualquier sistema autorizado pudiese utilizar los servicios web brindados por el juez en línea, contribuyendo a una integración más consolidada con los demás jueces en línea del mundo. Dentro de los servicios disponibles se encuentran: los de obtención de problemas, obtención de envíos, gestión del perfil de usuario, y los de gestión de correos (González Rodríguez, y otros, 2016).

SIGE

[17] Application Programming Interface.

El Centro de Tecnologías de Gestión de Datos (DATEC) enmarcado en la UCI, desarrolló el Sistema Integrado de Gestión Estadística (SIGE), con el objetivo de llevar a cabo el proceso de automatización de los procesos de gestión estadística en la ONEI[18] y en algunos de los organismos asociados a la misma. Para estos organismos utilizar algún proceso de SIGE tenían que contar con todo el sistema aunque no lo necesitaran en su totalidad, además, la solicitud y envío de la información era un proceso que podía demorar varios días. Para erradicar esa situación se desarrolló una capa de servicios web que permitiría a los organismos asociados a la ONEI utilizar algunos de sus procesos sin requerir la totalidad del sistema, agilizándose también el proceso de solicitud y envío de la información (García Companioni, y otros, 2016).

Tabla 3 Comparación entre sistemas homólogos

Sistema	Protocolo de comunicación	Formato de la respuesta	Métodos HTTP soportados	Código abierto
Facebook	REST	JSON	GET, POST, PUT, DELETE	Sí
COJ	REST	JSON	GET, POST, PUT, DELETE	Sí
SIGE	REST	JSON	GET, POST, PUT, DELETE	Sí

Una vez realizado el estudio del estado del arte, se concluye que el desarrollo de una capa de servicios web facilitará el acceso por parte de sistemas externos a la información almacenada en SIGIES. Esta capa de servicios implementará el protocolo REST y manejará los datos en el formato JSON. A pesar de que los sistemas homólogos estudiados soportan los cuatro métodos fundamentales de HTTP, la capa de servicios a desarrollar solo soportará el método GET ya que la información del sistema SIGIES no puede ser modifcada por aplicaciones externas.

1.3 Metodología de desarrollo

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo (Gómez, y otros, 2010). Las metodologias se clasfican en dos grandes grupos: tradicionales y ágiles.

[18] Oficina Nacional de Estadística e Información.

Metodologías tradicionales

Las metodologías de desarrollo tradicionales o también llamadas metodologías pesadas, tales como Cascada, Modelo-V o RUP[19], son metodologías que se basan en una serie de iteraciones secuenciales, llamadas fases. Estas fases dependen de un conjunto de procesos y documentación predeterminada que son elaboradas durante el desarrollo del proyecto y sirven de guía para futuros desarrollos (Calderón, y otros, 2015).

Como ejemplo de este tipo de metodología podemos citar el **modelo cascada**, el más característico y que ejemplifica mejor la metodología de desarrollo tradicional, además de haber sido probado con el tiempo y ser de fácil comprensión. Este es un modelo estático y se caracteriza por gestionar los desarrollos de sistemas de una manera lineal y secuencial, completando una actividad antes que la otra, de allí proviene su nombre. Dicho modelo consta de cuatro fases las cuales una vez desarrolladas por completo se procede a continuar con la siguiente pero ya no existe la posibilidad de regresar a la fase anterior, lo que requiere indispensablemente que el proyecto tenga un enfoque estructurado y orientado a los procesos. Esta característica hace que tenga una debilidad bien notoria, ya que si por alguna razón durante la fase de diseño surge algún problema que requiera regresar a la parte de análisis, o si el cliente solicita agregar o cambiar algunos requerimientos es imposible de realizar con el enfoque que tiene esta metodología. (Calderón, y otros, 2015)

Metodologías ágiles

Los constantes cambios en los ambientes de desarrollo de negocios y la evolución de la tecnología, dieron lugar a la necesidad de metodologías que asumieran tales retos de manera rápida durante las fases de desarrollo. Debido a esto, surgieron un nuevo tipo de metodologías que estaban basadas en el conocimiento de que los requerimientos eran cambiantes y dinámicos, las metodologías de desarrollo ágil. Las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Según Sommerville, estas metodologías son métodos de desarrollo iterativo que se centran en la especificación, diseño e implementación del sistema de forma incremental e implican directamente a los usuarios en el proceso de desarrollo (Sommerville, 2005).

Al no existir una metodología de software universal, toda metodología seleccionada para la realización de un proyecto, debe ser adaptada a las características del mismo, exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP[20] perteneciente a la categoría ágil, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. (UCI, 2016)

Esta variación de metodología, AUP-UCI, consta de tres fases para definir el ciclo de vida de los proyectos manteniendo la fase de Inicio, aunque se realizan modificaciones a la misma, englobando lo que era Elaboración, Construcción y Transición en Ejecución y agregando la fase de Cierre. Al igual que

[19] Rational Unified Process : Proceso Racional Unificado.
[20] Agil Unified Process : Proceso Unificado Ágil.

AUP, la versión UCI expone 7 disciplinas, pero concebidas de una manera más elemental: modelado de negocio,requisitos, análisis y diseño, implementación, pruebas internas, pruebas de liberación, pruebas de aceptación y despliegue.

La metodología AUP-UCI propone cuatro escenarios para el desarrollo de sus proyectos permitiendo así que los mismos puedan ajustar su modelo de negocio al más conveniente. En este caso se estará aplicando el escenario No.4 Historias de usuario, pues se adapta a las características del proyecto, ya que una vez evaluado el negocio a informatizar se obtienen procesos complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad (UCI, 2016).

1.4 Herramientas y tecnologías

Las herramientas informáticas son programas, aplicaciones o instrucciones que permiten la realización de un diverso número de funcionalidades con disímiles propósitos. Las tecnologías, en cambio, son saberes técnicos, ordenados científicamente que permiten perfilar y crear bienes y servicios. A continuación, se muestran las herramientas y tecnologías utilizadas para el desarrollo de la capa de servicios web para el SIGIES.

1.4.1 Lenguaje de programación

JAVA es un lenguaje de programación de alto nivel orientado a objetos de propósito general y basado en clases. Su sintaxis se asemeja a la de C y C++, pero elimina sus características menos usadas y más confusas proporcionando de esta forma simplicidad y facilidad de trabajo. Incluye la gestión de almacenamiento automático y es compilado a un conjunto de instrucciones de código de *bytes* y a formato binario. Cuenta además con potentes entornos de desarrollo como el *IDE*[21] *NetBeans* y una abundante documentación tanto en formato duro como digital. Proporciona portabilidad al software implementado ya que el mismo podrá ser ejecutado sobre cualquier plataforma utilizando el *JDK*[22] (Gosling, y otros, 2015).

1.4.2 Lenguaje de modelado

UML es un lenguaje gráfico para construir, documentar, visualizar y especificar un sistema de software. Viabiliza la realización de diagramas estáticos, dinámicos, de entorno y organizativos. Ofrece un estándar para describir de la manera más sencilla y entendible para cualquier desarrollador, los aspectos conceptuales tales como procesos de negocio y funciones del sistema (Booch, y otros, 2000).

1.4.3 Herramienta de modelado

El diseño de software normalmente se efectuará con un poco de ayuda de las herramientas de ingeniería de software asistida por ordenador, o herramientas *CASE*[23]. Es básicamente el uso de apoyo basado en la computadora por los desarrolladores para implementar y mantener el *software*, especialmente en la

[21] Entorno de Desarrollo Integrado
[22] Java Development Kit : Kit de Desarrollo de Java.
[23] Computer Aided Software Engineering : Ingeniería de Software Asistida por Computadora.

escala más grande, o para los proyectos más complejos. Las herramientas *CASE* permiten a los ingenieros de *software* dar un paso atrás de las complejidades reales de código al mirar el cuadro más grande y el diseño de sus proyectos más grandes. Desde el desarrollo, el diseño del sistema, la codificación a través de la prueba y mantenimiento, estas herramientas informáticas pueden utilizarse durante todo el ciclo de vida de *software* para asegurar que el producto final es de alta calidad, con defectos mínimos, y en la mayor parte de tiempo eficiente y de manera rentable posible (SOFTWAREENGINEERINSIDER, 2014).

Es seleccionada como herramienta CASE para el modelado de la capa de servicios **Visual Paradigm 8.0**, el cual cuenta con las siguientes características:

- Apoya a los arquitectos de sistemas, desarrolladores y diseñadores *UML*.
- Acelera el proceso de análisis y diseño de aplicaciones empresariales.
- Facilita la visualización *UML* en la última notación de *UML* 2.1 (SOFT112, 2014)

1.4.4 Entorno de Desarrollo Integrado

Durante el desarrollo del sistema se utiliza como herramienta para la implementación del código, **NetBeans**, pues es un entorno de desarrollo integrado libre, gratuito, de código abierto, sirve para un número importante de lenguajes de programación (Netbeans, 2016). Esta herramienta posee un potente depurador integrado, brinda soporte para *Symfony*, un *framework* Modelo-Vista-Controlador escrito en *PHP*[24] y además ofrece la posibilidad de crear aplicaciones web con dicho lenguaje.

1.4.5 ORM

Como *ORM*[25], es utilizado **Hibernate**, conocido por su excelente estabilidad y calidad, probado por la aceptación y el uso de desarrolladores de Java. Este ORM es una implementación de la especificación Java Persistence API (JPA). Se puede usar fácilmente en cualquier entorno compatible con JPA, incluidas aplicaciones *Java SE*[26], servidores de aplicaciones *Java EE*[27] y contenedores *Enterprise OSGi*[28]. Hibernate permite desarrollar clases persistentes siguiendo expresiones idiomáticas orientadas a objetos naturales que incluyen herencia, polimorfismo, asociación, composición y el marco de colecciones de Java. No requiere interfaces o clases base para clases persistentes y permite que cualquier clase o estructura de datos sea persistente. Admite la inicialización diferida, numerosas estrategias de búsqueda y bloqueo optimista con versiones automáticas y sellado de tiempo. Hibernate no requiere tablas ni campos de base de datos especiales y genera gran parte del SQL[29] al momento de la inicialización del sistema en lugar de hacerlo en tiempo de ejecución (Hibernate, 2017).

[24] Hypertext Preprocessor : Procesador de Hipertexto.
[25] Object-Relational Mapping : Mapeo Objeto-Relacional.
[26] Java Platform Standard Edition : Plataforma Java Edición Estandar.
[27] Java Platform Enterprise Edition : Plataforma Java Edición de Empresa.
[28] Open Services Gateway initiative : Iniciativa de Puertos de Servicios Abiertos.
[29] Structured Query Language : Lenguaje de Consulta Estructurado.

1.4.6 Sistema Gestor de Base de Datos

Como sistema gestor de bases de datos se utiliza **PostgreSQL** en su versión 9.4.1, a causa de que está distribuido bajo licencia BSD[30] (la cual permite el uso del código fuente en software no libre), demuestra un buen funcionamiento con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto, el sistema continuará funcionando (PostgreSQL, 2016).

Para administrar las bases de datos se utiliza **PgAdmin III** en su versión 1.18.1, pues es una aplicación de diseño y administración de bases de datos funcional con PostgreSQL. Está escrito en C ++ usando las *wxWidgets*[31] y es un marco multiplataforma (pgAdmin, 2013).

1.4.7 Sistema de Control de Versiones

Como herramienta para mantener el control de versiones se utiliza el **Git** en su versión 2.6.0, pues es un sistema de control de versiones, distribuido, libre y de código abierto, diseñado para manejar desde pequeños a muy grandes proyectos con rapidez y eficiencia. Supera a otras herramientas, con funciones tales como ramificación local (la cual consiste en que el usuario puede tener múltiples ramas locales independientes entre sí en las cuales alojar su proyecto) y múltiples flujos de trabajo (GitHub, 2016).

1.4.8 Marco de trabajo

La suite **WSO2**[32] ofrece un amplio conjunto de herramientas que ayudan a implementar una plataforma *SOA*. Tiene la característica de usar *JAVA* como lenguaje de programación lo que la hace una de las favoritas en sus funciones a raíz de la rapidez que brinda dicho lenguaje. Dentro de sus productos podemos encontrar: *Aplication Server, BAM, Data Service Server, Developer Studio, API Manager* e *Identity Server*.

Los productos de *WSO2* que sean seleccionados para cubrir las necesidades, ofrecen por defecto una funcionalidad muy general y flexible, sin embargo, es probable que necesiten ser personalizados a necesidades específicas. En una extensa comparativa de las alternativas *open source* para plataformas *SOA*, la suite *WSO2* sale favorecida frente a otras soluciones *open source* existentes. Para ilustrar dicha comparativa entre plataformas *SOA open source* se muestra el siguiente gráfico.

[30] Berkeley Software Distribution : Destribución de Software Berkeley.
[31] wxWidgets: son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++.
[32] Compañía que desarrolla aplicaciones de software abierto enfocadas en proveer una arquitectura orientada a servicios.

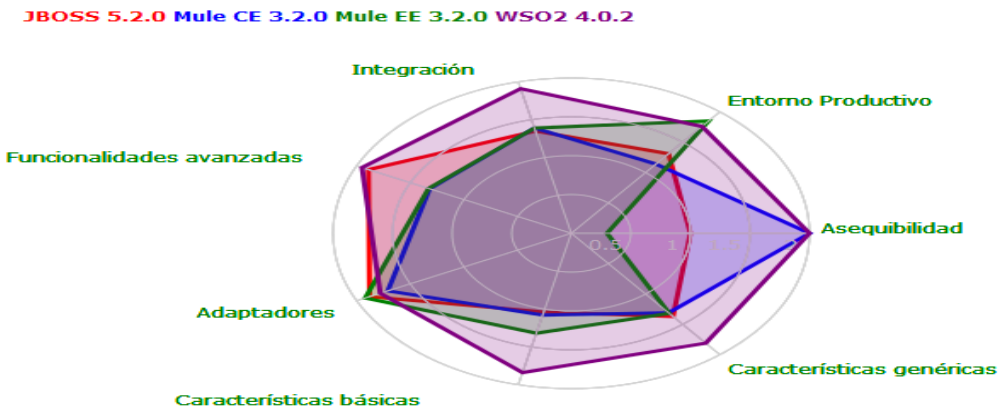


Figura 4 Comparativa entre alternativas *open source* para plataformas *SOA*

Esta suite tiene como característica la implementación de servicios web a través de *SOAP* y *REST*. Aunque en sus inicios sus productos estaban basados en *SOAP*, al cobrar auge, REST como protocolo de comunicación entre sistemas informáticos, productos como *Data Services Server* y *API Manager* implementan sus soluciones a través de *REST*.

Spring Framework

Para el lenguaje Java existen varios *framework* que permiten el desarrollo de servicios basados en REST, tales como: *Dropwizard, Play Framework, RESTEasy, Restlet y Spring Boot*. Dichos *framework*, 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 *framework*, 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 (Gajotres, 2017). Cualquier sistema operativo con una maquina virtual de Java puede ejecutar aplicaciones desarrolladas con este *framework* . Al poseer la característica de ser modular se hace flexible y configurable para cualquier tipo de aplicación y unida a su presencia en la categoría de software código libre, se convierte en uno de los entornos de trabajo favoritos de los desarrolladores.

Spring Boot

Como se puede apreciar, *Spring* tiene una amplia gama de módulos que necesitan gran número de configuraciones. Estas configuraciones pueden tomar mucho tiempo, pueden ser desconocidas para principiantes y suelen ser repetitivas. A raíz de esta situación y como solución a *Spring* surge *Spring Boot*, que aplica el concepto de *Convention over Configuration(CoC)* (Faraoni, 2015).

CoC es un paradigma de porgramación que minimiza las desiciones que tiene que tomar los desarrolladores, simplificando tareas. No obstante, la flexibilidad no se pierde, ya que a pesar de otorgar valores por defecto, siempre se puede configurar de forma extendida. De esta forma se evita la repetición de tareas básicas a la hora de construir un proyecto (Faraoni, 2015).

Algunas de las características de *Spring Boot* son citadas a continuación (Phillip Webb, 2017):

- Permite externalizar su configuración para trabajar con la misma aplicación código en diferentes entornos. Puede usar archivos de propiedades, archivos *YAML*[33], variables de entorno y argumentos de línea de comandos para externalizar la configuración.
- Es ideal para el desarrollo de aplicaciones web. Puede crear fácilmente un servidor *HTTP* autónomo que usa *Tomcat* integrado, *Jetty*, *Undertow* o *Netty*.
- Los repositorios JPA de *Spring Data* son interfaces que se pueden definir para acceder a los datos. Se crean consultas JPA automáticamente desde los nombres del método. Por ejemplo, una interfaz *CityRepository* podría declarar un método *findAllByState (String state)* para encontrar todas las ciudades en un estado determinado.

1.5 Conclusiones del capítulo

Para el desarrollo de la capa de servicios web de SIGIES se realizó un estudio de los conceptos principales asociados a servicios web, que permitieron hacer la selección de la metodología, herramientas y tecnologías, asociadas a dicha implementación. Como guía del proceso de desarrollo de la capa de servicios fue seleccionada AUP-UCI por ser una metodología ágil que ofrece un escenario conveniente para representar niveles de detalles y relaciones entre procesos. El lenguaje de modelado previsto es *UML* v2.1 y como herramienta *CASE Visual Paradigm* en su versión 8.0. Por otra parte, *JAVA8* es el lenguaje de programación escogido mientras que *NetBeans v8.1* es el *IDE* de desarrollo a emplear. Se utiliza PostgreSQL v9.4.1 como Sistema Gestor de Bases de Datos y como administrador de bases de datos *PgAdmin III*. Como framework de desarrollo fue seleccionado *Spring Boot* v2.0.1 por las ventajas de desarrollo que ofrece y su actualización en el desarrollo de aplicaciones Java web.

[33] Formato de serialización de datos

Capítulo 2: Descripción de la solución propuesta

En el presente capítulo se exponen características de la propuesta de solución. Dichas especificaciones son mostradas mediante el modelo de dominio; el cual permite a través de las relaciones entre los principales conceptos comprender el negocio. Se hace además un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr el correcto funcionamiento de la capa de servicio que se desea implementar. Se describe el análisis y diseño de las clases y se construyen los artefactos correspondientes según la metodología seleccionada. Además, se realizan los diagramas de secuencia, se definen los patrones de diseño y patrón arquitectónico como parte del diseño del módulo.

2.1 Modelo de dominio

Un modelo de dominio o conceptual explica los conceptos significativos en un dominio del problema; es el artefacto más importante a crear durante el análisis orientado a objetos. Además una cualidad esencial que debe ofrecer un modelo de dominio es que represente cosas del mundo real, no componentes del software (Craig, 1999).

2.1.1 Diagrama de modelo de dominio

A continuación se muestra el modelo de dominio que describe la relación existente entre los conceptos del dominio identificados, para un mayor entendimiento del problema.

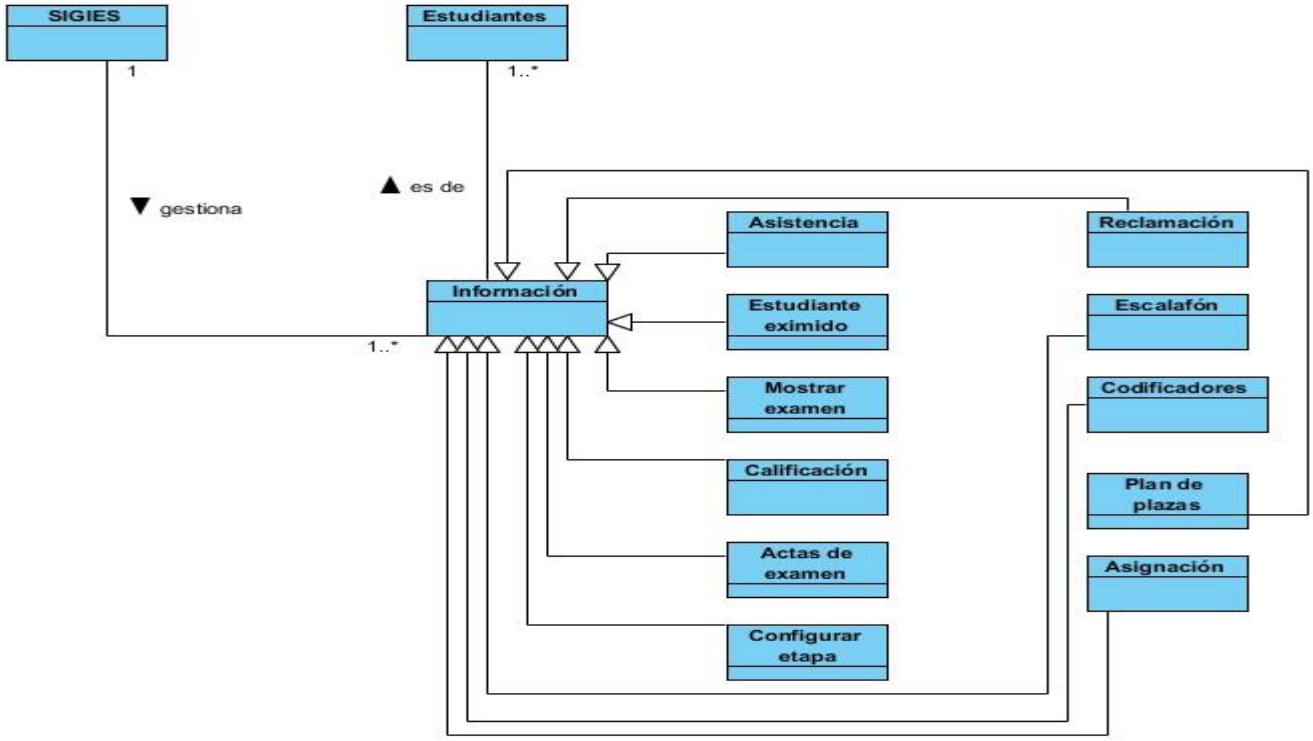


Figura 5 Diagrama de Modelo de dominio

2.1.2 Definición de los conceptos del modelo de dominio

SIGIES: sistema que gestiona el proceso de ingreso de los estudiantes a la Educación Superior.

Asistencia: representa la participación de un estudiante en su correspondiente examen de ingreso a la educación superior.

Estudiante eximido: representa el estudiante que no deba realizar algún examen de ingreso.

Mostrar examen: representa el instrumento evaluativo donde el estudiante evidenciará su nivel de conocimiento en una asignatura dada. Además, es el proceso donde se realiza una revisión de un examen en una convocatoria dada donde un estudiante no está conforme con su nota de reclamación.

Calificación: representa la nota obtenida por un estudiante en los exámenes de ingreso a la educación superior realizados en una convocatoria dada.

Actas de examen: representa un documento oficial que se genera como evidencia de la culminación de un proceso dado.

Configurar etapa: representa la configuración de las distintas etapas del sistema.

Reclamación: representa el proceso donde se realiza una revisión de un examen en una convocatoria dada donde un estudiante no está conforme con su nota de calificación.

Estudiante: representa la persona que aspira al ingreso de la educación superior.

Escalafón: representa el valor númerico asociado a un estudiante relacionado con las notas de examen y el promedio académico, o solo con las notas de examen.

Codificadores: término que es utilizado para representar varios elementos en el sistema, como son: carrera, área de la ciencia, organismo formador, etc.

Plan de plazas: representa el documento que aborda las carreas que se ofertan cada año, así como la cantidad de plazas de cada una generado por la Dirección de Ingreso y Ubicación Laboral para cada provincia.

Asignación: gestiona las convocatorias que se realizan durante el proceso de ingreso incluyendo los procesamientos que se realizan en cada una de ellas, y además incluye el reajuste al plan de plazas y la asignación de plazas.

2.2 Descripción de la propuesta de solución

Para dar solución a la problemática presentada por SIGIES, se propone desarrollar una capa de servicios web que garantice la interoperabilidad entre aplicaciones informáticas de terceros y el sistema. La capa abordará tres procesos principales: seguridad, autorización y servicios. La autorización y la seguridad, procesos llevados a cabo en el servidor de autorización, tienen una estrecha relación, pues validan el acceso correcto a las funcionalidades del sistema. La autorización consistirá en la obtención de permisos para acceder a un tipo de información, los cuales son evaluados por la seguridad implementada en cada servicio del servidor de recursos. Se delimitarán las respuestas a las solicitudes mediante el encapsulamiento de las mismas en tres niveles de acceso: máximo, medio, y bajo. Los servicios que expondrá la capa permitirán la obtención de datos personales perteneciente a un estudiante, la

publicación del periodo de ingreso, la convocatoria del examen que se realiza, el otorgamiento de carrera, y el estado del procesamiento una vez realizado el examen de ingreso. Para garantizar la gestión de clientes, permisos y autorizaciones, se desarrollará una interfaz para la administración de credenciales de clientes, tipo de información a la que podrá acceder y el tipo de autorización que presentará al servidor.

2.3 Especificación de requisitos del software

Al realizar la especificación de requisitos podremos definir las exigencias que ha de cumplir el software para satisfacer las necesidades del cliente. Estos requisitos se dividen en dos clases: funcionales y no funcionales.

2.3.1 Requisitos funcionales

Los requisitos funcionales (en lo adelante RF) son las declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a las entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los RF de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Sommerville, 2007). A continuación se muestran los requisitos funcionales que debe cumplir la capa de servicios web a implementar.

RF_1: Autenticar usuario.

RF_2: Autorizar cliente.

RF_3: Listar clientes.

RF_4: Registrar cliente.

RF_5: Eliminar cliente.

RF_6: Editar cliente.

RF_7: Filtrar cliente.

RF_8: Incluir escenario a un cliente.

RF_9: Incluir autorización a un cliente.

RF_10: Mostrar periodo de ingreso.

RF_11: Mostrar estado de procesamiento de exámenes.

RF_12: Mostrar convocatoria de examen.

RF_13: Mostrar datos personales del estudiante.

RF_14: Mostrar si un estudiante está exonerado de algún examen.

RF_15: Mostrar resultado de la asignación de carrera.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener que lo hacen atractivo, usable, rápido o confiable. Estos requisitos pueden marcar la diferencia entre un producto aceptado y otro con poca aceptación, no definen el éxito general del producto, pero si influyen en la evaluación del cliente (Sommerville, 2005).

Hardware del servidor:

RNF1. Para un buen funcionamiento el servidor de aplicación web donde se ejecute el sistema debe contar con las siguientes características mínimas: Procesador Intel Core i3, CPU 3.30 GHz, Memoria física de 4Gb y 100Gb de disco duro.

Software del Servidor:

RNF2. GNU/Linux como sistema operativo.

RNF3. Máquina Virtual de Java.

Seguridad:

RNF4. Los métodos que requieren de autentificación en la capa de servicios deben estar disponibles solo para aquellos usuarios correctamente autenticados y que posean los permisos requeridos por el servicio.

RNF5. Por las características del sistema, se ha definido que se devuelve un token al autenticar un cliente y que el tiempo límite de demora que tiene para solicitar un servicio antes de que expire su token, dependerá de la cantidad de información que maneje el cliente según sus permisos, pasado ese tiempo tendría que volver a obtener uno nuevo.

Portabilidad:

RNF6. Independientemente de la plataforma de desarrollo en la que fue implementada la capa de servicios web, las aplicaciones que la consuman pueden hacerlo sin tener en cuenta el lenguaje de programación o la plataforma en que fue desarrollada.

Rendimiento:

RNF7. La capa de servicios debe ser capaz de responder una petición en el menor tiempo posible dependiendo del tipo de petición y los datos que se manejan en esta.

Soporte:

RNF8. Se requiere una documentación apropiada que describa todas las funcionalidades del sistema desarrollado, así como una guía para su uso.

Flexibilidad:

RNF9. El sistema permitirá que se defina mediante parámetros las diferentes formas en las que se realizarán las peticiones.

Disponibilidad:

RNF10. La aplicación debe estar disponible en todo momento.

2.4 Modelación a través de Historia de Usuarios

Una historia de usuario (HU), es una represntación de un requisito en lenguaje común usada en metodologías de desarrollo ágiles. Estas sustituyen a los documentos de especificación funcional, y a los casos de uso. Las HU deben poder ser programadas en un tiempo de una a tres semanas, de lo contrario ha de ser dividida. En esencia, son las ideas del cliente organizadas y agrupadas de acuerdo a su funcionalidad. A su vez, también se tiene en cuenta un orden tal que permita al mismo priorizar sus necesidades y al equipo de trabajo definir las que resultan críticas o claves en el momento de desarrollo de la solución. En la claridad de su descripción radica el éxito del proyecto (JOSKOWICZ, 2008). A continuación se muestra la HU correspondiente al requisito funcional: Mostrar periodo de ingreso.

Tabla 4 HU_Mostrar periodo de ingreso

Número: 10	**Nombre del requisito:** Mostrar periodo de ingreso.
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Mostrar periodo de ingreso para realizar acciones dentro del correspondiente.

2- Acciones para lograr el objetivo (precondiciones y datos):

La aplicación cliente debe estar correctamente autenticada y ha de solicitar dicho servicio a través de la URL definida para el mismo.

Debe existir el periodo de ingreso en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Una vez realizada la validación de las credenciales del cliente, los permisos de acceso a datos y el token para la solicitud del servicio, el sistema muestra en formato JSON una estructura de información compuesta por:

Nombre del periodo de ingreso.

Identificador del periodo ingreso.

En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL:

Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la petición al servidor de recurso.

Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.

4- Flujo de la acción a realizar:

El cliente envía sus credenciales al servidor de autenticación especificando en la URL el tipo de autorización que presenta.

El sistema retorna en caso satisfactorio el tipo correspondiente al token de acceso y la cadena correspondiente al mismo, además del tiempo en que expira el token obtenido. En caso contrario especifica si el error producido proviene de los datos del cliente, el tipo de autorización o permisos de acceso a datos.

El cliente utiliza el token especificando el tipo del mismo para acceder a la información que necesita del servidor de recurso. Cada petición al servidor estará encabezada por dicho token de acceso.

El sistema muestra en formato JSON los datos del año escolar.

Una vez vencido el tiempo de validez, el sistema muestra un mensaje de error al cliente.

El cliente debe realizar el proceso desde el inicio si desea seguir consultando información.

Tabla 5 HU_Obtener datos personales del estudiante

Número: 13	**Nombre del requisito:** Obtener datos personales del estudiante.
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Mostrar datos personales de un estudiante conociendo su carnet de identidad.

2- Acciones para lograr el objetivo (precondiciones y datos):

La aplicación cliente debe estar correctamente autenticada y ha de solicitar dicho servicio a través de la URL definida para el mismo.

Debe existir la información del estudiante en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Una vez realizada la validación de las credenciales del cliente, los permisos de acceso a datos y el token

para la solicitud del servicio, el sistema muestra en formato JSON una estructura de información compuesta por:

Nombre
Apellidos
CI
Dirección particular
Provincia de residencia
Municipio de residencia
Color piel
Sexo
Índice académico
Vía de ingreso
Preuniversitario
Procedencia
Identificador de la plaza otorgada
SMA

En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL:

Se devuelve el error HTTP 401 UNAUTHORIZED si el token de usuario es incorrecto.

Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.

4- Flujo de la acción a realizar:

El cliente envía sus credenciales al servidor de autenticación especificando en la URL el tipo de autorización que presenta.
El sistema retorna en caso satisfactorio el tipo correspondiente al token de acceso y la cadena correspondiente al mismo, además del tiempo en que expira el token obtenido. En caso contrario especifica si el de error producido proviene de los datos del cliente, el tipo de autorización o permisos de acceso a datos.
El cliente utiliza el token especificando el tipo del mismo para acceder a la información que necesita del servidor de recurso. Cada petición al servidor estará encabezada por dicho token de acceso.
El sistema muestra en formato JSON los datos correspondientes al estudiante cuyo carnet de identidad se corresponde con el pasado por parámetro.
Una vez vencido el tiempo de validez, el sistema muestra un mensaje de error al cliente.
El cliente debe realizar el proceso desde el inicio si desea seguir consultando información.

2.5 Modelo de diseño

El modelo de diseño crea un modelo de software enfocado en la representación de los datos, las funciones y el comportamiento requerido. Permite al ingeniero de software modelar el sistema o producto que se va a construir, prosibilitando evaluar su calidad y efectuar mejoras antes de generar el código. Con este se obtiene una representación arquitectónica, de interfaz y despliegue del sistema (Pressman, 2010).

2.5.1 Arquitectura del sistema

La arquitectura de software de un sistema informático es la estructura o estructuras del sistema, que comprenden elementos de software, las propiedades externamente visibles de esos elementos y la relación entre ellos (Bass, 1998). La arquitectura no es el software operacional sino la representación que capacita al ingeniero del software para: (1) analizar la efectividad del diseño para la consecución de los requisitos fijados, (2) considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil, y (3) reducir los riesgos asociados a la construcción del software (Pressman, 2010).

La capa de servicios web de SIGIES se implementará bajo la arquitectura de software ***n-capas***, específicamente cuatro capas: presentación, controladora, repositorio y modelo. La siguiente figura muestra la arquitectura de la capa de servicios.

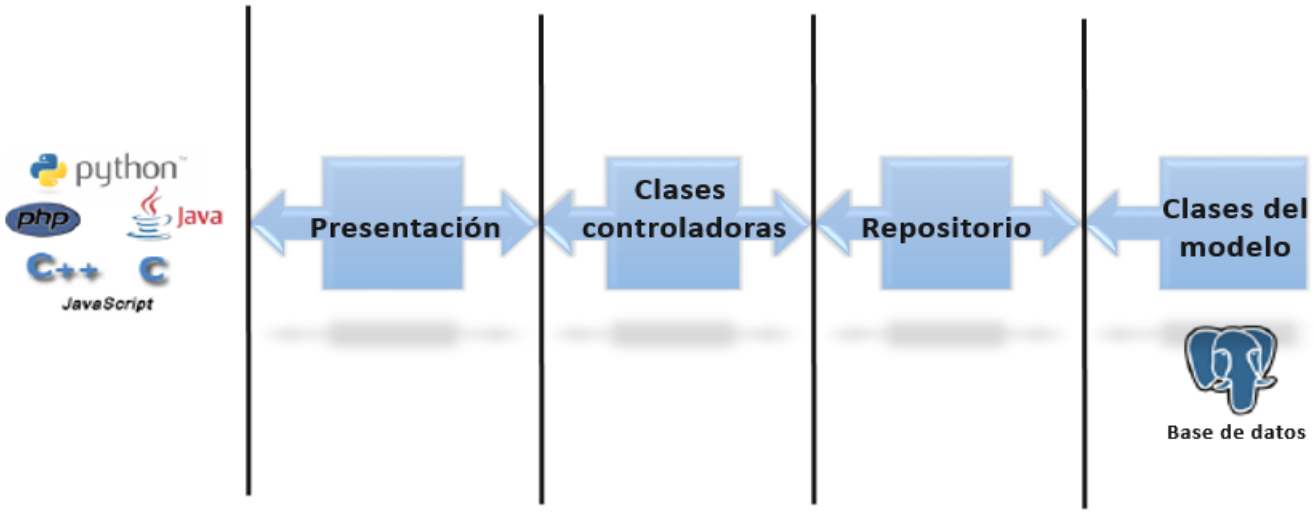


Figura 6 Arquitectura de la capa de servicios de SIGIES

Este tipo de arquitectura estructurada en capas está fundamentada en un funcionamiento jerárquico de los roles y responsabilidades otorgadas a cada una, con el fin de proporcionar una distribución efectiva del problema a resolver. Puede realizar el desarrollo en varios niveles y de ser necesario realizar algún cambio, solo se efectua en el nivel requerido, constiuyendo esta característica su principal ventaja.

Cada uno de los módulos del sistema estará implementado bajo esta arquitectura, difiriendo en la forma de presentación de los datos solicitados. El módulo de servicios en su capa de presentación retornará

una respuesta en formato JSON, teniendo en cuenta que los datos del servidor de recursos serán intercambiados a nivel de aplicación. Por otra parte, en la administración se obtendrán como respuesta páginas HTML.

2.5.2 Diagramas de clases del diseño

Los diagramas de clases del diseño representan un conjunto de interfaces, colaboraciones y sus relaciones. Gráficamente son una colección de nodos y arcos. Muestran una serie de clases, elementos y contenidos, representados a través de las relaciones entre ellos, conformando de esta manera la estructura del sistema. Un diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Estas clases se especifican utilizando la sintaxis del lenguaje de programación elegido (Pressman, 2010). A continuación se muestra el diagrama de clases asociado a los servicios web del sistema pertenecientes a un estudiante.

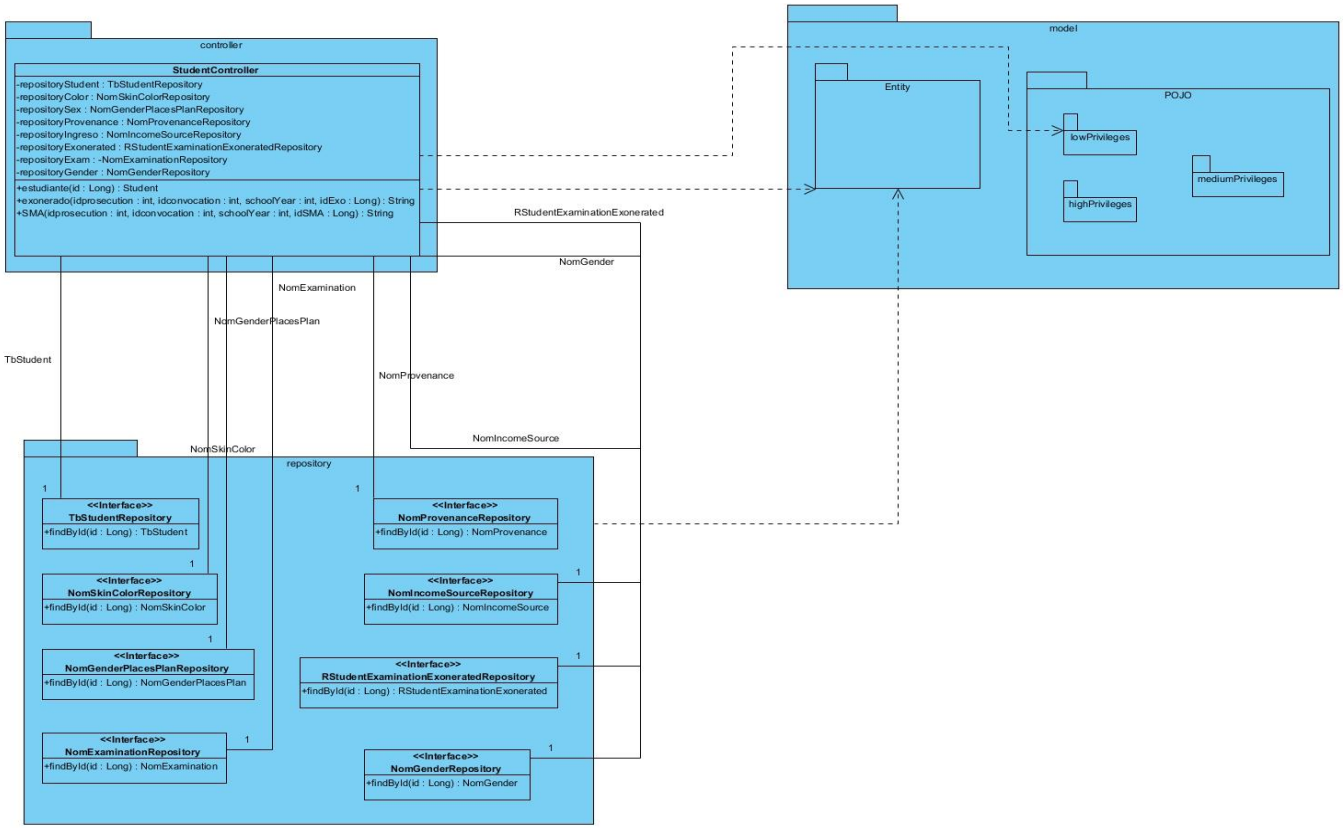


Figura 7 Diagrama de clases asociado a los servicios de un estudiante

2.5.3 Patrones de diseño

Un patrón de diseño es una descripción de un problema y su solución, el cual recibe un nombre y puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software. Al contrario, estos intentan codificar el conocimiento, las expresiones y los principios ya existentes (Craig,

1999).

Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés General Responsibility Asignment Software Patterns), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (Craig, 1999). A continuación se describen los patrones GRASP utilizados en el desarrollo de la capa de servicios web.

Experto: consiste en realizar una correcta asignación de responsabilidades en las clases utilizadas de tal forma que la tarea de implementar un método o crear un objeto, debe recaer sobre aquella que conoce toda la información necesaria para llevar a cabo dicha acción. De este modo se obtiene un diseño con mayor cohesión, la información se mantiene encapsulada y se disminuye el acoplamiento (Carmona, 2012). Se evidencia dicho patrón en la clase PlazaController, por ser experta en información para cumplir con la responsabilidad de crear un objeto de tipo PlazaNombre o ejecutar un método dentro de dicha clase.

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/service/plaza/")
public class PlazaController   {
@GetMapping("plazas")
        public List<PlazaNombre> obtCarrer(){
            List<NomCareer> carreras = repositoryCarrer.findAll();
}
```

Figura 8 Fragmento de código que ilustra el uso del patrón de diseño Experto

Controlador: es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Guía la asignación de responsabilidades relacionadas con la creación de objetos, lo cual permite instanciar y crear las clases que le son necesarias para cumplir sus funcionalidades. La arquitectura MVC brinda una capa específicamente para los controladores, que son el núcleo de este, y especifica la presencia de este patrón (Carmona, 2012). Este patrón se pone de manifiesto en las clases cuyo nombre terminan en Controller, pues la arquitectura de Spring (MVC) brinda una capa específicamente para los controladores, en la que se evidencia la presencia de este patrón. El siguiente fragmento de código pone de manifiesto su uso:

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/service/student/")
    public class StudentController {
```

Figura 9 Fragmento de código que ilustra el uso del patrón de diseño Controlador

Bajo acoplamiento: El acoplamiento es el grado de relación que tienen los elementos de un sistema, dígase clases, módulos u otro elemento. El bajo acoplamiento es la idea de tener estos elementos o clases lo menos ligadas entre sí, de modo que de ocurrir una modificación en una de ellas se tenga la mínima repercusión posible en el resto de las clases. Significa que debe haber pocas dependencias entre las clases para ser más entendidas cuando estén aisladas, porque no tendrían tantas dependencias de otras clases (Carmona, 2012). Este patrón se aplica en todas las clases desarrolladas.
Creador: Se encarga de asignar a la clase B la responsabilidad de crear una instancia de clase A, si se cumple una de las siguientes condiciones (Craig, 1999)

- B contiene A.
- B agrega A.
- B tiene los datos de inicialización de A.
- B registra A.
- B utiliza A muy de cerca.

Se utiliza en las clases controladoras, pues a estas se le da la responsabilidad de crear instancias de las entidades que sean necesarias para llevar a cabo una acción. Por ejemplo, para la acción de listar el plan de plazas por provincia, la clase PlazaController crea instancias de RPlacesPlanCareer.

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/service/plaza/")
public class PlazaController   {
   @GetMapping("plazas")
public List<PlazaNombre> obtCarrer(){
            List<NomCareer> carreras = repositoryCarrer.findAll();
            List<RPlacesPlanCareer> PlacesPlanCareer = repositoryPlacesPlan.findAll();
}
```

Figura 10 Fragmento de código que ilustra el uso del patrón de diseño Creador

Alta cohesión: Se encarga de asignar las responsabilidades de modo que se mantenga una alta cohesión (Craig, 1999). Este patrón se utilizó en todas las clases desarrolladas, especialmente en la asignación de las responsabilidades de los distintos controladores. Esto permitió que cada controlador

realizara las acciones más afines al negocio que representa, evitando, por ejemplo, que PlazaController se encargue de mostrar la información referente a un estudiante en lugar de StudentController.

Patrones GoF

Los patrones de diseño Pandilla de los Cuatro (GoF, por sus siglas en inglés Gang of Four), describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. Se dividen en tres tipos fundamentales, agrupados según el tipo de soluciones que aplican: creacionales, estructurales y de comportamiento. A continuación se describe el patrón GoF, utilizado en el desarrollo de la capa de servicios.

Contrato Uniforme: Es un patrón usado exclusivamente en servicios web, ya que plantea el uso de los cuatro verbos principales del protocolo HTTP en tareas específicas. Se pone de manifiesto en la capa de servicios debido al uso de estos verbos de forma estándar para las siguientes tareas:

- POST: Crear recursos.
- GET: Servicios que devuelven el estado de los recursos.
- PUT: Para actualizar recursos dado la URL del mismo.
- DELETE: Eliminar un recurso y después la URL del mismo no es válida.

La capa de servicios de SIGIES solo utiliza el método GET, pues no permite a los usuarios realizar acciones de inserción, actualización y eliminación sobre los datos que almacena.

2.6 Diseño de base de datos

El diseño de base de datos consiste en definir la estructura de los datos en un sistema de información. La estructura del modelo relacional se define como un conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias, claves foráneas. (Costal, s.f)

El modelo entidad-relación es uno de los enfoques de modelado de datos que más se utiliza actualmente por su simplicidad y legibilidad, la cual proporciona una notación diagramática muy comprensiva. Es una herramienta que ayuda al diseñador a reflejar en un modelo conceptual los requisitos del mundo real, y permite comunicarse con el usuario final sobre el modelo conceptual obtenido para verificar si satisface sus requisitos (Costal, s.f).

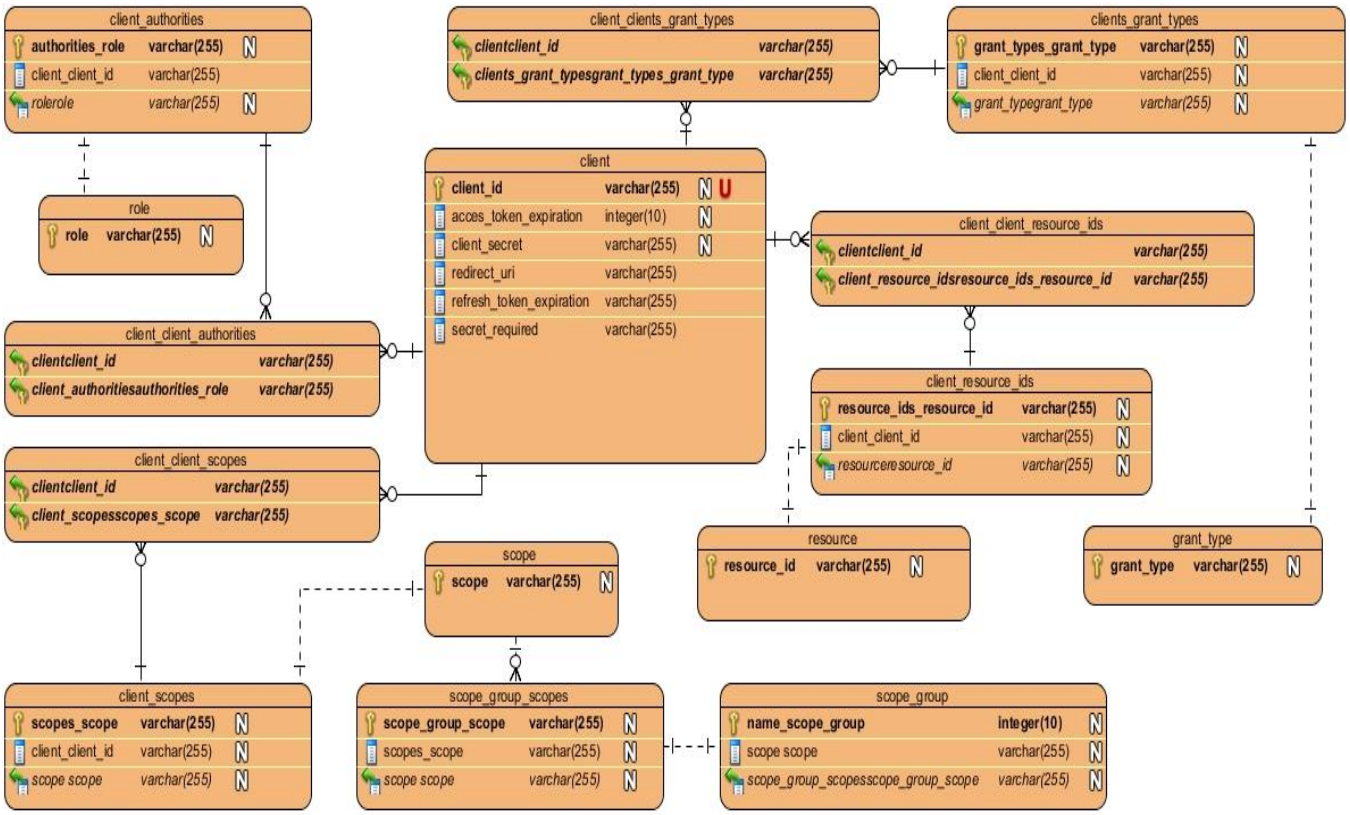


Figura 11 Diseño de base de datos

2.7 Conclusiones del capítulo

En el desarrollo del presente capítulo se realizó una representación visual de clases conceptuales del entorno real de los objetos del proyecto a través del modelo de dominio. Fueron identificados 15 requisitos funcionales y 10 no funcionales que la capa de servicios debe cumplir para su correcto funcionamiento. Teniendo en cuenta el escenario número cuatro de AUP-UCI, se realizaron las historias de usuario (HU), por requisitos, resultando ser un método eficaz para la descripción y especificación de los mismos. Se aplicaron patrones de diseño *GRASP* y *GoF*, dirigidos a diseñar correctamente la capa de servicios web. Además, se definió como arquitectura del sistema n-capas, sirviendo como puto de partida para la ejecución de la implementación del sistema.

Capítulo 3. Iplementación y prueba

En el presente capítulo se analizan los aspectos relacionados con la implementación de funcionalidades de la capa de servicios. Además, se muestra el diagrama de despliegue y el diagrama de componente, realizándose finalmente las pruebas al software que permitan verificar el correcto desarrollo de los requisitos establecidos.

3.1 Modelo de implementación

El proceso de implementación parte como resultado del análisis y diseño de la propuesta de solución planteada, se tiene como objetivo llevar la arquitectura y el sistema como un todo. Describe como se organizan los componentes de acuerdo con los mecanismos de estructuración y modelación disponibles en el entorno de implementación. Además de los lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (Jacobson, 2000).

3.1.1 Diagrama de componentes

Los diagramas de componentes representan la estructura física del código. Asignan la vista lógica de las clases del proyecto a los archivos que contienen el código fuente en el que se implementa la lógica. Cuando se genera código representan la ubicación de los archivos de código fuente para sus clases. Al realizar ingeniería inversa en un proyecto ya existente pueden ayudar a establecer relaciones entre cada diagrama de clases y los archivos de código fuente.

El uso más importante de este diagrama es mostrar la estructura de alto nivel del modelo de implementación, especificando los subsistemas de implementación, sus dependencias a la hora de importar el código y para organizar los subsistemas de implementación en capas y en paquetes (ALTOVA, 2016). A continuación se presenta el diagrama de componentes correspondiente a la capa

de servicios.

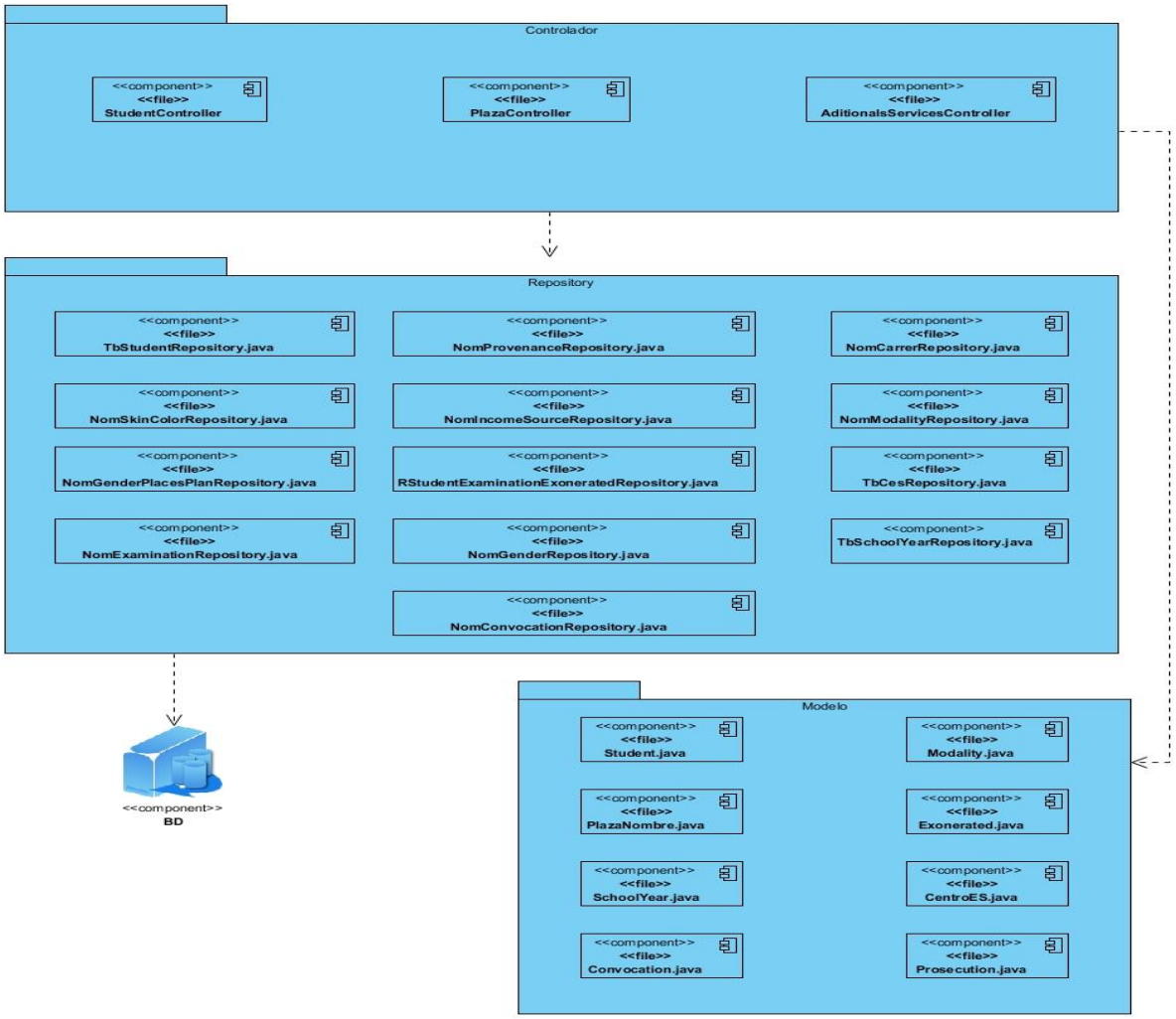


Figura 12 Diagrama de componentes del sistema

3.1.2 Diagrama de despliegue

El diagrama de despliegue es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue de los artefactos del software en los destinos de distribución (Melchor, 2012).

El diagrama de despliegue correspondiente a la capa de servicios web presenta como flujo la conexión de una aplicación cliente al servidor de autenticación utilizando HTTP como protocolo de comunicación. Este a su vez se conecta a la base de datos de clientes que contiene mapeada el modelo de datos. Después de realizar la autorización se accede al servidor de recursos el cual a través de TCP-IP extrae los datos almacenados en SIGIES. A continuación se muestra el diagrama de despliegue correspondiente a la capa de servicios web.

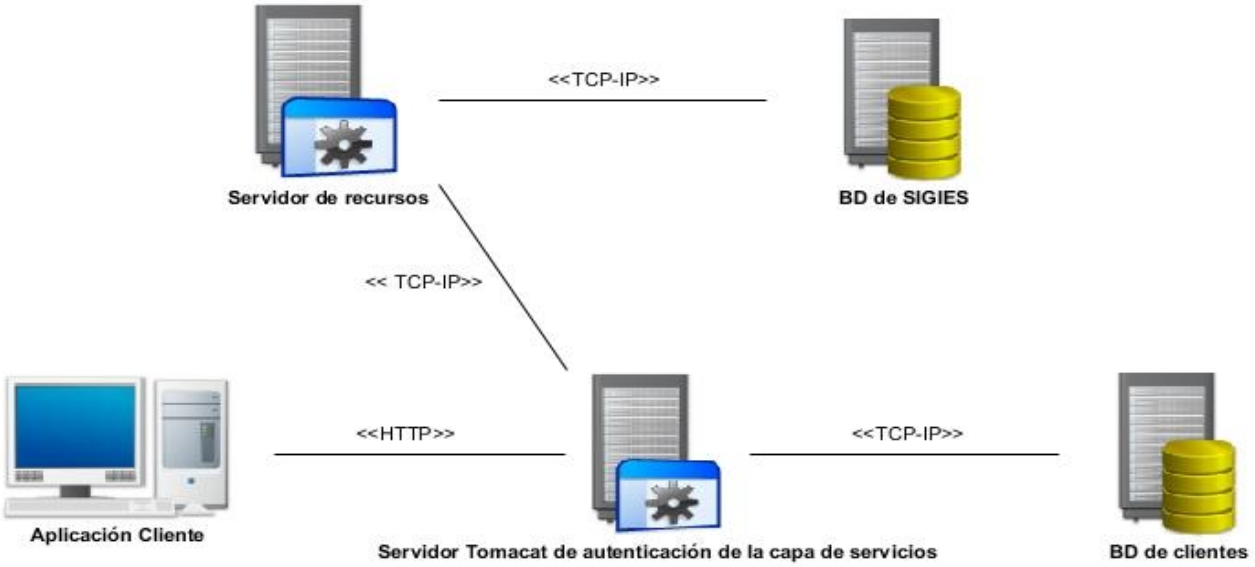


Figura 13 Diagrama de despliegue del sistema

3.2 Módulo oauth2-server

El módulo dedicado a la seguridad se encargará de proteger la información y asegurar que cada cliente acceda a la correspondiente según sus permisos. La capa de servicios *web* de SIGIES implementa Oauth2 como protocolo de autorización, teniendo en cuenta que es conveniente aplicarlo en APIs que poseen más de un cliente. Este protocolo surge a partir del nacimiento de la *web* social, con el objetivo de permitir que los usuarios autoricen a terceros a acceder a su información sin que estos tengan que conocer las credenciales del usuario.

Oauth2 especifica un conjunto de *grant types*[34] para distintos escenarios. Los más comunes son: *Authorization Code, Implicit, Client Credentials, Password, Device Code* y *Refresh Token* (Oauth, 2018). En la capa de servicios desarrollada se utilizó el tipo de autorización *Client Credentials*, teniendo en cuenta que solo se necesita verificar qué cliente de los registrados en base de datos es el que solicita la información. Este método se basa en la solicitud de un *token*[35] al servidor de autorización usando las credenciales *client_id* y *secret_id*, para acceder a los recursos protegidos por el propietario del servidor. A continuación se ilustra el flujo para la obtención del *token*.

[34] Tipos de autorización.
[35] Cadena de caracteres con significado en un lenguaje de programación.

Figura 14 Flujo para el tipo de autorización Client Credentials

El flujo ilustrado en la figura 6 incluye los siguientes pasos (Microsoft, 2012):

1) El cliente se autentica con el servidor de autorización usando sus credenciales y solicita un token de acceso para realizar peticiones al servidor de recursos.
2) El servidor de autorización autentica al cliente y valida el tipo de autorización, si es válida, emite un token de acceso, en caso contrario especifica el tipo de error que se produjo.
3) El cliente realiza una solicitud de recurso protegido al servidor de recursos, presentando el token de acceso.
4) El servidor de recursos valida el token de acceso, y si es válido, atiende la solicitud.
5) Los pasos (3) y (4) se repiten hasta que el token de acceso caduque. Una vez caducado el token se procede a ejecutar el paso número (6).
6) Si el token de acceso no es válido, el servidor de recursos regresa un error de token inválido.
7) El cliente solicita un nuevo token de acceso al autenticarse con el servidor de autorización usando sus credenciales. Los requisitos de autenticación del cliente se basan en el tipo de cliente y en las políticas del servidor de autorización.
8) El servidor de autorización autentica al cliente y emite un nuevo token de acceso.

El módulo oauth2-server presenta una estructura dividida en cuatro directorios principales: *api, config, domain* y *service*. Las clases controladoras para la selección de la información que se mostrará en cada vista de administración se implementarán en el directorio *api*. El directorio *config* contiene las clases de configuración para el servidor oauth y el método de seguridad. Los detalles de los clientes que serán gestionados por el administrador se definen como clases dentro del paquete *domain*, donde cada una representa una tabla en base de datos que agrupará la información de cada cliente del sistema. Las vistas desarrolladas para la administración serán ubicadas en *templates*, subdirectorio de *resource.*

3.3 Módulo services

El módulo dedicado a los servicios es el de mayor complejidad estructural. El mismo se compone de tres paquetes generales: *controller*, *model* y *repository*. En el caso particular del paquete *model*, se agregan además dos subpaquetes: *entity*, que posee el esquema de datos separado por agrupaciones de clases, y *POJO*[36], que divide la estructura de los datos de respuesta en tres categorías o niveles de permisos: *lowPrivileges*, *mediumPrivileges* y *highPrivileges*.

El directorio *repository* engloba las clases de tipo interfaz que referencian los datos almacenados en las entidades del modelo, y *controller* contiene las clases controladoras para cada uno de los grupos de servicios.

3.4 Creación de un servicio

La implementación de un nuevo servicio que cumpla con los requerimentos de la arquitectura de la capa de servicios de SIGIES, conlleva a crear una clase que defina la estructura del objeto de retorno, la cual se ubicaría dentro del directorio *POJO*. Esta acción delimita los datos que se seleccionarán de un objeto que posee información general. Debe tenerse en cuenta que el servicio implementado ha de encontrarse dentro de su categoría de privilegio. El siguiente paso consiste en crear dentro del paquete *repository*, el repositorio de tipo interfaz correspondiente a la tabla que posee los datos del servicio. Finalmente se determina el controlador que corresponde al servicio y de no estar anteriormente declarado se crea dentro del paquete *controller*. A continuación se ejemplifica lo anteriormente descrito a través de la creación del servicio *obtenerCES()*.

Creación de la estructura del objeto de retorno

```
package cu.uci.cpsigies.model.POJO.lowPrivileges;
Public class CentroES {
private Long id;
private String name;
private String code;
public animal(Long id, String name, String code) {
        this. id = id;
         this. name = name;
         this. code = code;
}
public String getId() {
         return id;
}
    public String getName() {
         return name;
    }

    public String getCode() {
```

[36] Plain Old Java Object : Objetos de Java Planos Antiguos.

```
        return code;
    }
}
```

Creación del repositorio correspondiente a la clase del modelo de datos que contiene la información

```
package cu.uci.cpsigies.repository;
public interface TbCesRepository extends JpaRepository<TbCes,Long> {
// Métodos del repositorio que serán invocados desde el controlador que lo utiliza.
}
```

Creación de la clase controladora de servicios adicionales

Las clases controladoras demandan anotaciones propias que validen el correcto funcionamiento del servicio, por ejemplo la URL de acceso. Es una buena práctica definir una URL general del controlador, que constituya la cabecera de cada servicio, los cuales le añaden a la misma su identificador o palabra(s) clave(s). A continuación se ejemplifica la creación de una clase controladora y el trabajo con las URL dentro de la misma.

```
package cu.uci.cssigies.controller;
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/service/aditional") //
public class ServicesAditionalsController {
// Atributos de la clase
@GetMapping("/{ces}") // Identificador a nivel atómico del método.
    public List<TbCes> obtenerCES() {
// Procedimientos para obtener petición
// Respuesta del método.
}
```

Ficheros de configuración

A continuación se ejemplifican algunos de los archivos de configuración más importantes sobre los cuales se pueden realizar modificaciones para el correcto funcionamiento del proyecto.

application.properties

```
spring.datasource.url=jdbc:postgresql://{dirección IP de la PC que posee la base de datos}/{nombre de la base de datos}
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.datasource.driver-class-name=org.postgresql.Driver
```

```
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=false
```

application.yml

```
spring:
    application:
        name: spring-boot-oauth2-clients
    mvc:
        favicon:
            enabled: false
        throw-exception-if-no-handler-found: true
    main:
        banner-mode: 'off'
security:
    oauth2:
        resource:
            filter-order: 3
        client:
            authenticationScheme: header
logging:
    level:
      org.springframework.security: DEBUG

server:
    port: 8083
    error:
        whitelabel:
            enabled: false
```

pom.xml

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>LATEST</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security.oauth</groupId>
            <artifactId>spring-security-oauth2</artifactId>
        </dependency>
```

Los archivos *application.properties* y *application.yml* son de nivel atómico para cada módulo que se desarrolle en la aplicación, teniendo en cuenta que cada servidor debe tener su puerto de acceso y

debe corresponderse con una base de datos. Maven[37] utiliza el *pom.xml* para describir el proyecto de *software* a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Entre las tareas que realiza se encuentra la compilación del código y su empaquetado.

3.5 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez, el estándar de codificación debería establecer cómo operar con la base de código existente (MICROSOFT, 2015).

Para el desarrollo de la capa de servicios web se tuvieron en cuenta los estándares que a continuación se explican:

- *UpperCamelCase*: la primera letra de la nueva palabra es mayúscula, permitiendo que sea fácil de distinguir de un nombre lowerCamelCase. En la capa de servicios web se utiliza para nombrar las clases (TECHTARGET, 2015).
- *lowerCamelCase*: la primera letra del primer nombre es minúscula. La ventaja de CamelCase es que en cualquier sistema informático donde las letras de un nombre deben ser contiguas (sin espacios), se puede crear un nombre más significativo utilizando una secuencia descriptiva de palabras sin violar la limitación de nomenclatura. En la capa de servicios se utiliza este estilo para nombrar las variables y los métodos (TECHTARGET, 2015).

```
@GetMapping("ces")
        public List<CentroES> obtenerCES(){
CentroES plazasR= new CentroES(...);
}
```

Figura 15 F Fragmento de código que ilustra el estándar de codificación: UpperCamelCase y lowerCamelCase

- Número de declaraciones por línea: en la capa de servicios *web* se declara cada variable en una línea distinta.

[37] Herramienta de software para la gestión y construcción de proyectos Java.

```
@GetMapping("plazas")
public List<PlazaNombre> obtCarrer(){
    List<NomCareer> carreras = repositoryCarrer.findAll();
    List<RPlacesPlanCareer> PlacesPlanCareer = repositoryPlacesPlan.findAll();
    List<PlazaNombre> result = new LinkedList<>();
}
```

Figura 16 Fragmento de código que ilustra el estándar de codificación: Número de declaraciones por línea

- Sentencias de importación: no existen líneas en blanco entre ellas.

```
import cu.uci.cssigies.Modelo.POJO.lowPrivileges.Student;
import cu.uci.cssigies.Repository.*;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.math.BigInteger;
```

Figura 17 Fragmento de código que ilustra el estándar de codificación: Sentencias de importación

3.6 Tratamiento de errores

El tratamiento de errores no es más que un procedimiento llevado a cabo en un sistema informático con el fin de dar solución a los comportamientos incorrectos del sistema. Estos errores pueden ser ocasionados por problemas internos del sistema, deficiencias de red, saturación de la pila, errores en el envío de parámetros por las URL de los servicios o sobrecarga de peticiones a un servicio. A continuación se muestra una tabla que relaciona los posibles errores HTTP y su causa.

Tabla 6 Descripción de posibles errores

# error HTTP	Error	Razón
400	Petición Incorrecta	Incluye una serie de errores relacionados con las peticiones que hacen los clientes hacia la capa de servicio si no cumplen estas con las especificaciones requeridas.
401	No autorizado	Incluye una serie de errores que indican que no están autorizados los clientes a ejecutar esa petición en el sistema. Ya sea porque no están autenticados o expiró el tiempo del token.
404	No encontrado	Incluye una serie de errores en las peticiones de los clientes a la hora de pedir un recurso que no existe en el sistema.

3.7 Pruebas de software

Hoy en día, debido al aumento del tamaño y la complejidad del software, el proceso de prueba se ha

convertido en una tarea vital dentro del proceso de desarrollo de cualquier sistema (Gutiérrez, y otros, 2004). Los ingenieros de software garantizan la calidad aplicando métodos técnicos sólidos y medidos, llevando a cabo pruebas de software bien planificadas (Pressman, 2010).

El proceso de pruebas de software tiene dos objetivos fundamentales:

- Demostrar al desarrollador y al cliente que el software satisface sus requisitos.
- Descubrir defectos en el software en que el comportamiento de este es incorrecto, no deseable o no cumple su especificación (Sommerville, 2005).

3.7.1 Métodos de prueba

Se puede probar cualquier producto de ingeniería de dos formas: concociendo la funcionalidad específica para la cual fue diseñado, demostrando mediante pruebas que cada funcionalidad es operativa o conociendo el funcionamiento del producto, donde cada componente interno trabaja de la forma adecuada. Al primer enfoque se le denomina prueba de caja negra y al segundo prueba de caja blanca (Pressman, 2010).

Pruebas de caja blanca: constituyen un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de pruebas. Estos últimos aseguran que durante la prueba se han ejecutado, por lo menos una vez, todas las sentencias del programa y que se ejercitan todas las condiciones lógicas (Pressman, 2010).

Pruebas de caja negra: son diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa. Estas pruebas ignoran intencionalmente la estructura de control y se centra en el ámbito de información de un programa, de forma que se proporcione una cobertura completa de prueba (Pressman, 2010). Para la validación de la propuesta de solución será aplicado el método de caja negra, teniendo en cuenta la técnica de partición de equivalencia. A través del examen de valores válidos y no válidos, esta técnica permite examinar las entradas existentes en el software y descubrir errores. El método seleccionado permitirá corregir problemas de rendimiento, en la interfaz del usuario y se comprobará que la capa de servicios realiza las funciones requeridas por el usuario.

3.7.2 Niveles de prueba

Los niveles de prueba son diferentes ángulos de verificar y validar en determinados momentos el ciclo de vida del software. En el desarrollo de la fase de pruebas de la capa de servicios web de SIGIES se aplicarán las pruebas que abarcan los siguientes niveles:

Aceptación: las pruebas de aceptación son realizadas con el objetivo de que el cliente pruebe el software y verifique que cumpla con sus expectativas. Estas pruebas generalmente son funcionales y se basan en los requisitos definidos por el cliente y deben hacerse antes de la salida a producción. Las pruebas de aceptación son fundamentales por lo cual deben incluirse obligatoriamente en el plan de pruebas de software. Estas pruebas se realizan una vez que ya se ha probado que cada módulo

funciona bien por separado, que el software realice las funciones esperadas y que todos los módulos se integran correctamente.

Unitario: las pruebas unitarias tienen el mayor efecto en la calidad del código cuando son parte integral del flujo de trabajo de desarrollo de software. Es una forma de probar que secciones de código funcionen correctamente. Ello permite garantizar que cada sección evaluada funcione de manera eficiente por separado.

Sistema: Las pruebas de sistema se realizan cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.

3.7.3 Descripción de los tipos de prueba realizados

A partir de los niveles de pruebas anteriormente explicados se seleccionan los siguientes tipos de prueba para asegurar el correcto funcionamiento de la capa de servicios web de SIGIES.

Pruebas de aceptación: determinación por parte del cliente de la aceptación o rechazo del sistema desarrollado. Es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción. Detecta errores en el sistema bajo un ambiente controlado. Se llevan a cabo en el lugar en donde fue desarrollado el software.

La realización de las pruebas de aceptación tiene como objetivo que el producto final cuente con el menor número de errores posibles. Para ello se describen los diseños de casos de prueba. A continuación se describen algunos de los casos de pruebas utilizados.

Tabla 7 DCP Mostrar periodo de ingreso

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Mostrar periodo de ingreso	El cliente solicita una vez autenticado satisfactoriamente consultar el servicio que ofrece el periodo de ingreso.		El sistema muestra una respuesta en formato JSON que contiene los datos: identificador del periodo escolar y nombre del periodo escolar. En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL: • Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la petición al	/api/service/aditional/school_year

			servidor de recurso. • Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.	

Tabla 8 DCP Mostrar datos personales del estudiante

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Mostrar datos perosnales del estudiante	El cliente solicita una vez autenticado satisfactoriamente consultar el servicio que ofrece los datos de un estudiante a través de la URL definida para el servicio solicitado.		El sistema muestra una respuesta en formato JSON que contiene los datos: • Nombre • Apellidos • CI • Dirección particular •Provincia de residencia •Municipio de residencia • Color piel • Sexo • Índice académico • Vía de ingreso • Preuniversitario • Procedencia • Identificador de la plaza otorgada • SMA En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL: • Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la peticion al servidor de recurso. • Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.	/api/service/student/{ci}

Con el fin de detectar la mayor cantidad de no conformidades posibles fueron realizadas tres iteraciones de pruebas. A continuación se muestra una tabla que ilustra las iterciones realizadas y las no conformidades detectadas correspondiente a cada una.

Tabla 9 Resultados de las pruebas obtenidas por iteración

Iteraciones	Cantidad de DCP	No conformidades detectadas			
		Alta	Media	Baja	Total
1	15	15	28	30	73
2	15	6	12	10	28
3	15	3	5	1	9

En cada interación fue resuelta cada una de las no conformidades identificadas, contribuyendo al mejoramiento de la propuesta para obtener un producto de calidad libre de errores y satisfacer las necesidades de los usuarios finales.

Pruebas de seguridad y control de acceso: se enfocan a la seguridad en el ámbito de la aplicación, asegurando que los mecanismos de protección incorporados en el sistema lo protegerán de accesos impropios y que los usuarios solo accedan a los servicios que requieran autenticación cuando estén debidamente registrados y autenticados en el sistema mediante la capa de servicios.

Para la realización de las pruebas de seguridad a la capa de servicios web se utilizó la herramienta informática *Postman,* simulando la petición de un token al servidor Oauth. Se utilizan las credenciales de un cliente de SIGIES y sus respectivos niveles de acceso o privilegio definido.

Tabla 10 Pruebas de seguridad

Descripción	Resultado
Las primeras pruebas fueron intentos de violación a la seguridad de la capa de servicios web, tratando de acceder con credenciales de clientes no existentes o utilizando clientes reales pero con un tipo de autorización no correspondiente.	Estas pruebas de acceso arrojaron resultados favorables, ya que no se logró acceder con credenciales no registrados previamente ni con falsas autorizaciones, demostrando que la capa de servicios web cuenta con un mecanismo de autenticación seguro, lo que garantiza que solamente accedan usuarios autorizados previamente.

Se intenta acceder a funciones no permitidas para un determinado cliente. Los roles utilizados fueron los siguientes: Administrador: rol que puede ejercer cualquier funcionalidad en SIGIES. Usuario: rol que solo puede acceder a los servicios del servidor de recursos.	El cliente registrado tiene acceso solamente a las funcionalidades predefinidas por los administradores del sistema, impidiendo en todo momento que se lleven a cabo peticiones a recursos no autorizados.

La siguiente figura muestra la respuesta del sistema ante el primer caso de violación de seguridad.

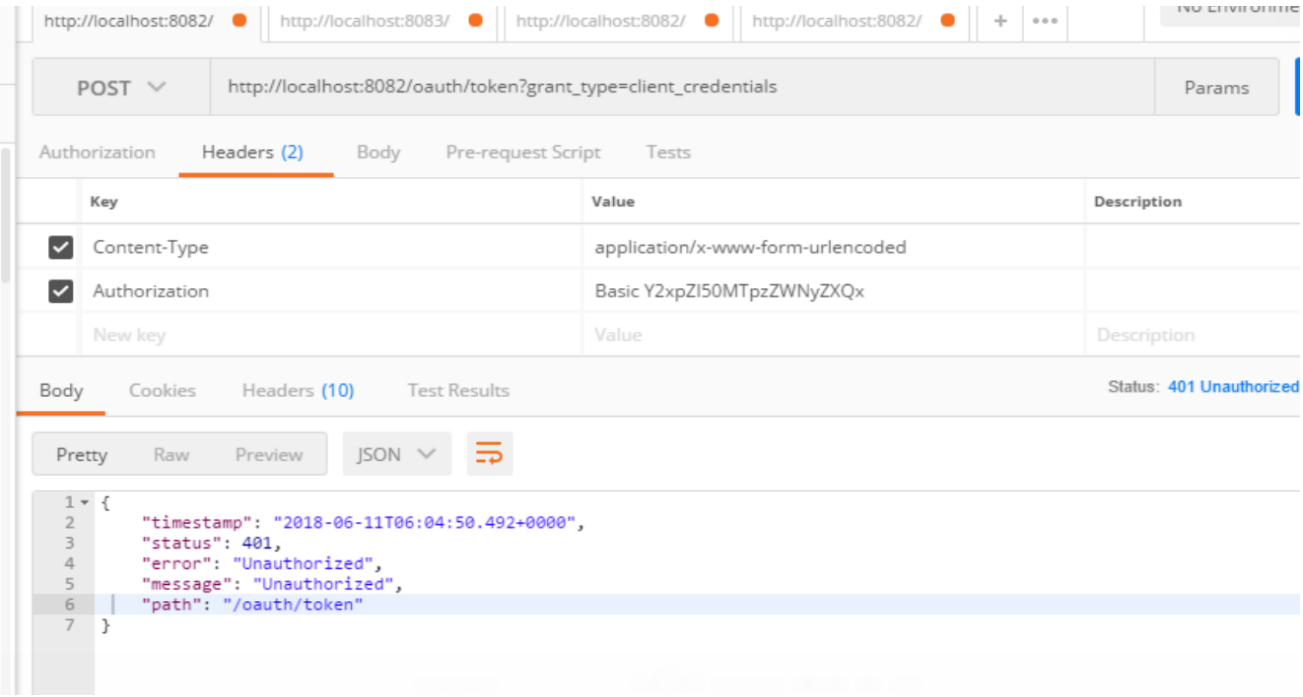


Figura 18 Respuesta del sistema ante intento de violación de seguridad con usuario inexistente

La siguiente figura muestra la respuesta del sistema ante el segundo caso de violación de seguridad.

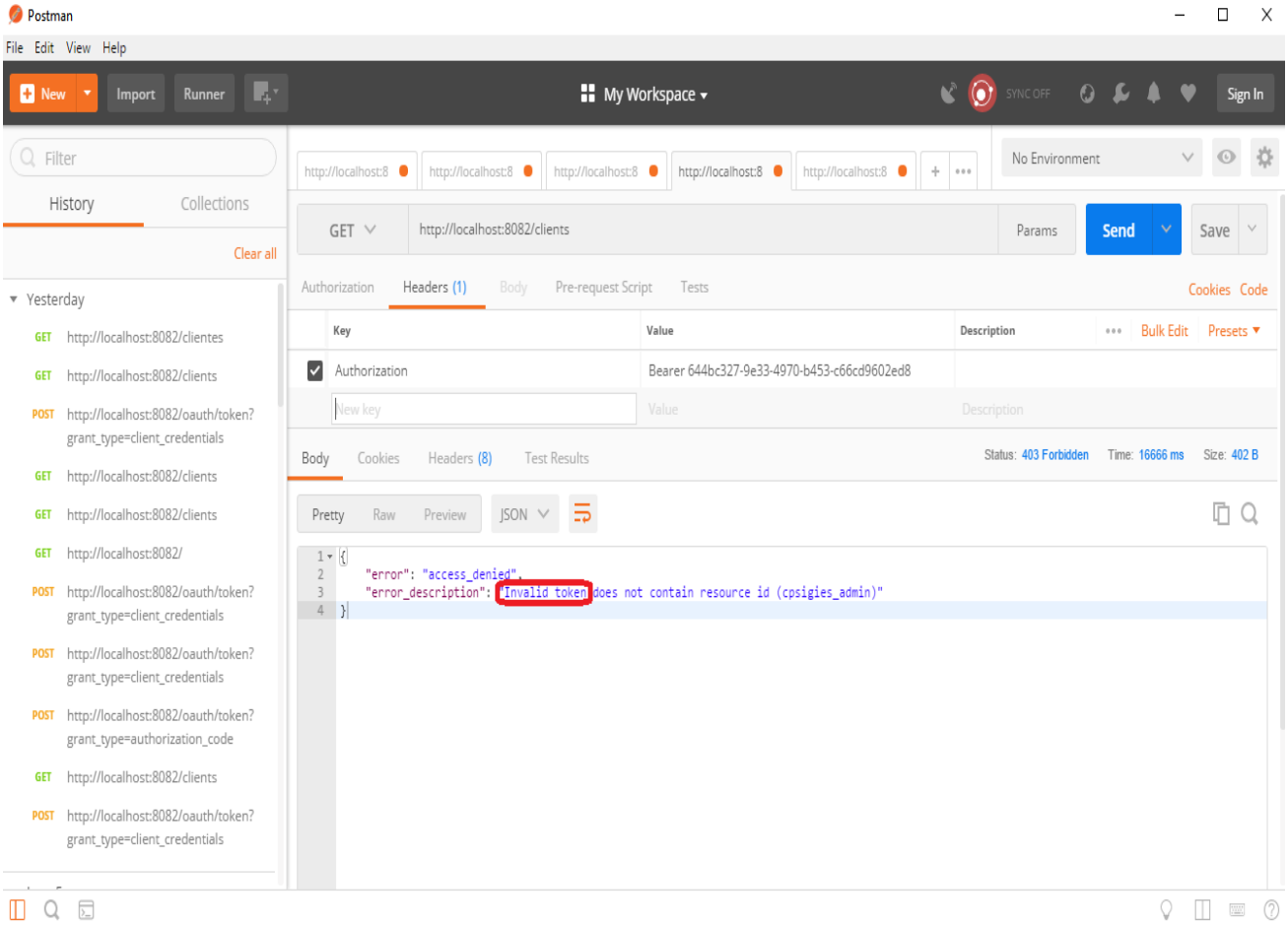


Figura 19 Respuesta del sistema ante intento de violación de seguridad con token inválido

A continuación se detallan algunos de los errores o alertas de seguridad producidos por el servidor de autorización ante intentos de violación del sistema:

invalid_request: este tipo de error es producido cuando a la solicitud le falta un parámetro para que el servidor pueda continuar con la solicitud. Esto también se puede devolver si la solicitud incluye un parámetro no compatible o repite un parámetro.

invalid_client: producido cuando la autenticación del cliente contiene un error en los paramrtros client_id o secret_id. Se envía una respuesta HTTP 401 en este caso.

invalid_scope: para solicitudes a servicios que incluyen el tipo de autorización *Password* o *Client_Credentials*, este error indica un valor de *scope* no válido en la solicitud.

cliente no autorizado: este cliente no está autorizado a utilizar el tipo de autorización solicitada. Por ejemplo, si se restringen un grupo específico de aplicaciones que pueden usar la concesión implícita, devolverá este error para las demás.

unsupported_grant_type: si se solicita un tipo de concesión que el servidor de autorización no reconoce, es devuelto este código.

Pruebas de rendimiento: se enfocan en la capacidad de recibir peticiones mediante la utilización de alguna herramienta, verificando cuantas peticiones pueda sostener el sistema sin que este se vea

afectado, así como la velocidad de respuesta del mismo. Las pruebas de rendimiento para la capa de servicios web serán realizadas utilizando la herramienta JMeter, que es una aplicación de escritorio y de código abierto, diseñada para realizar pruebas de carga y estrés, medir los tiempos de las pruebas y medir el rendimiento de un sistema. JMeter también puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web.

Para la realización de la pruebas se utilizó el entorno de *hardware* y *software* del servidor, que el mismo consiste en:

- PC con procesador i3 a 3.30 GHz y 4 GB de RAM.
- Sistema Operativo GNU/Linux.
- Servidor Web Apache Tomvat en su versión 7.0.
- Servidor de base de datos.
- PostgreSQL en su versión 9.4.1.

Se probó el sistema con un total de 1500 peticiones enviadas en 100 hilos de ejecución concurrentes en 6 servicios web, arrojando un 1,09% de error total al atender dichas peticiones a los servicios. En resumen se puede decir que la capa de servicios web cumple con el rendimiento requerido, para atender la carga y stress que se genera al anteder las solicitudes de los diferentes clientes que puedan conectarse a la capa.

Las pruebas no pueden demostrar que el software está libre de defectos o que se comportará en todo momento como está especificado. Siempre es posible que una prueba que se haya pasado por alto pueda descubrir problemas adicionales en el sistema. Generalmente, el objetivo de las pruebas de software es convencer a los desarrolladores del sistema y a los clientes de que el software es lo suficientemente bueno para su uso operacional. La prueba es un proceso que intenta proporcionar confianza en el software (Sommerville, 2005).

3.8 Conclusiones del capítulo

Mediante el diseño del diagrama de componente se representaron las relaciones de dependencia entre los componentes del sistema, para una mejor comprensión del funcionamiento de la capa de servicios y las tecnologías que la componen. El uso de Oauth2 garantizó la seguridad de la capa, posibilitando que cada cliente del sistema obtenga el acceso a la información disponible para sus permisos. La realización de pruebas permitieron comprobar que las funcionalidades fueron implementadas correctamente y que el sistema satisface las necesidades del cliente. Fueron detectadas 110 no conformidades, las cuales se solucionaron en tres iteraciones.

CONCLUSIONES

Luego de realizar el análisis, diseño, implementación y las pruebas a la capa de servicios web para el Sistema de Gestión para el Ingreso a la Educación Superior, se arribaron a las siguientes conclusiones:

- El estudio de soluciones similares posibilitó el desarrollo de una capa de servicios que garantiza el intercambio de información entre SIGIES y aplicaciones clientes de terceros. Además se pudieron definir las herramientas y tecnologías adecuadas para la implementación.
- El desarrollo de la capa de servicios, constituye una guía para la futura implementación de servicios que garanticen obtención de los datos de SIGIES por parte de sistmas externos.
- Las diferentes pruebas de software realizadas verefican el correcto funcionamiento del sistema, asegurando que la capa de servicios responderá a las peticiones de las aplicaciones clientes sin errores en las respuestas.

RECOMENDACIONES

Concluida la investigación, con el objetivo de perfeccionar la aplicación y lograr una mayor explotación de sus potencialidades se recomienda:

- Implementar los servicios asociados a la información que gestionan las herramientas SIGENU e INPES, para que puedan ser consumidos por ellas.

Bibliografía

Bibliografía

ALTOVA. 2016. ALTOVA. [En línea] 2016. [Citado el: 19 de 5 de 2018.] http://www.altova.com/es/umodel/uml-component-diagrams.html.

Bass, L., P. Clements y R. Kazman,. 1998. *Software Architecture in Practice.* s.l. : Addison-Wesley, 1998.

Booch, Grady, Rumbaugh, James y Jacobson, Ivar. 2000. *EL Lenguaje Unificado de Modelado.* Segunda edición. 2000.

Calderón, Roberto Suyomi Taype y Diego Alonso Godoy Alvarez. 2015. *Modelos de aceptación de meodologías de desarrollo de software.* [ed.] Roy Pérez. Lima : s.n., 2015. pág. 290.

Carmona, Juan García. 2012. Solid y GRASP. *Buenas prácticas hacia el éxito en el desarrollo de software.* 2012.

Catalani, E. 2012. Primeros pasos con REST(Transferencia de estado Representacional) parte 1. [En línea] 2012. [Citado el: 31 de 10 de 2017.] http://exequielc.wordpress.com/2012/10/12/primeros-pasos-con-rest-transferencia-de-estado-representacional-parte-1/..

Consortium, W3C. 2004. Web Services Architecture. [En línea] 2004. [Citado el: 30 de 10 de 17.] http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#whatis.

Costal, Dolors Costa. s.f. *Introduccion al diseño de base de datos.* s.f.

Craig, L. 1999. *UML y patrones.Una introducción al análisis y el diseño orientado.* 1999.

Craig, Larman. 1999. *UML y patrones.Una introducción al análisis y el diseño orientado.* 1999.

Erl. 2014. Service-Oriented Architecture: a Field Guide to Integrating XML and Web Services. s.l. : Prentice Hall, 2014.

Facebook. 2018. Facebook for developers. [En línea] 2018. [Citado el: 17 de 06 de 2018.] https://developers.facebook.com/docs/graph-api/overview/.

—. 2018. Newsroom. [En línea] 2018. [Citado el: 14 de 2 de 2018.] https://es.newsroom.fb.com/company-info.

Faraoni, Federico Julián Gutierrez. 2015. *Desarrollo de una aplicación web con Spring Framework para un gestor de un Recetario.* Madrid : s.n., 2015.

Fernández Romero, Yenisleidy y Díaz Gonzále, Yanette. 2012. Patrón Modelo-Vista-Controlador. La Habana : s.n., 2012, Vol. 11, págs. 47-57. Delegación Provincial del MININT, CUJAE.

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/.

García Companioni, José Luis y Noda García, Yacel. 2016. *Capa de Servicios Web para el Sistema Integrado de Gestión Estadística (SIGE).* La Habana : Universidad de las Ciencias Informáticas, 2016.

Gil Aros, Celio. 2009. Los Web Services y características de calidad. [En línea] 2009. [Citado el: 27 de 10 de 2017.] http://www.unilibre.edu.co/revistaavances/avances_10/r10_art7.pdf.

GitHub. 2016. -Git --fast-version-control. [En línea] 2016. [Citado el: 06 de 11 de 2017.] https://git-scm.com/.

Gómez, Oscar Tinoco, López, Pedro Pablo Rosales y Bacalla, Julio Salas. 2010. Criterios de selección de metodologías de desarrollo de software. [En línea] 07 de 2010. [Citado el: 31 de 10 de 2017.] http://www.redalyc.org/articulo.oa?id=81619984009.

González Rodríguez, Cesar Andrés y Almaguer Guerra, Omar. 2016. *Capa de servicios web para el Juez en Línea Caribeño.* La Habana : Universidad de las Ciencias Informáticas, 2016.

Gosling, James, y otros. 2015. *The Java Language Specification-Java SE 8 Edition. Redwood : Oracle America, Inc., 2015. Vol. 8.* 2015. Vol. 8.

Gradmann, S. 2009. Interoperabilidad. [En línea] 7 de 4 de 2009. [Citado el: 31 de 10 de 2017.] Un concepto clave a bibliotecas digitales a gran escala persistentes. . http://www.digitalpreservationeurope.eu/publications/briefs/es_interoperabilidad.pdf.

Guevara, Yurisander. 2016. Cubadebate. *Informatización en Cuba: la ruta estratégica.* [En línea] 11 de 09 de 2016. [Citado el: 05 de 10 de 2017.] Entrevista a Magda Brito D'Toste, directora de

Bibliografía

Informatización en el Ministerio de Comunicaciones (Mincom). http://www.cubadebate.cu/especiales/2016/09/11/informatizacion-en-cuba-la-ruta-estrategica/.
Gutiérrez, Agustin F, Orantes, Sandra D y López, Máximo. 2009. Arquitecturas empresariales: gestión de procesos de negocio vs. arquitecturas orientadas a servicios, ¿se relacionan?. Tecnura : s.n., 2009, Vol. 13, págs. 136-144.
Gutierrez, Javier J. 2014. ¿ Qué es un framework Web? [En línea] mayo de 2014. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.
Gutiérrez, Javier Jesús, y otros. 2004. *Pruebas Funcionales y Pruebas de Carga.* s.l. : Departamento de Lenguajes y Sistemas Informáticos, 2004.
Hibernate. 2017. Hibernate. *Hibernate.* [En línea] 2017. [Citado el: 14 de 03 de 2018.] http://hibernate.org/.
IBM. 2014. Introducción a SOA y servicios web. [En línea] 2014. [Citado el: 30 de 10 de 2017.] http://www.ibm.com/developerworks/ssa/webservices/newto/service.html.
International, ECMA. 2013. *The JSON Data Interchange Format.* 2013.
ISO. 2000. 2382-7:2000, ISO/IEC. [En línea] 2000. [Citado el: 27 de 10 de 2017.] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=7241 .
Jacobson, IvarBooch, et al. 2000. El proceso unificado de desarrollo de software/The unified software development process. *El proceso unificado de desarrollo de software/The unified software development process.* s.l. : Pearson Educación, 2000.
JOSKOWICZ, José. 2008. *Reglas y prácticas en eXtreme Programming.* 2008. Vol. 22.
Larman, Craig. 2003. Modelo de dominio. *UML y Patrones.* s.l. : Prentice Hall, 2003.
Lin, Boci, y otros. 2012. *Comparison between JSON and XML in Applications on AJAX.* 2012.
Melchor, Alex Prezi. 2012. Tutorial Diagramas de Despliegue. [En línea] 2012. [Citado el: 19 de 5 de 2018.] https://prezi.com/e_gpb7xev_im/tutorial-diagramas-de-despliegue.
MES. 2015. Ministerio de Educación Superior de la República de Cuba. [En línea] 2015. [Citado el: 30 de 09 de 2017.] http://www.mes.edu.cu.
Métodos para generar casos de prueba funcional en el desarrollo del software. **González, Liliana Palacio. 2009.** 2009, Revista Ingenierías .
MICROSOFT. 2015. Revisiones de código y estándares de codificación. [En línea] 2015. [Citado el: 24 de 4 de 2018.] https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx..
Microsoft. 2012. *The OAuth 2.0 Authorization Framework.* [ed.] Dick Hardt. 2012. pág. 76. ISSN: 2070-1721.
Netbeans. 2016. Netbeans.org. *Página Principal de Netbeans.* [En línea] 2016. [Citado el: 3 de 11 de 2017.] https://netbeans.org.
Oauth. 2018. oauth.net. [En línea] 2018. [Citado el: 6 de 6 de 2018.] https://oauth.net/.
Patrón Modelo-Vista-Controlador. **Fernández, Yenisleidy Romero y Díaz, Yanette González. 2012.** 1, La Habana : s.n., 2012, Vol. 11.
pgAdmin. 2013. PostgreSQL Tools. [En línea] 2013. [Citado el: 06 de 11 de 2017.] https://www.pgadmin.org.
Phillip Webb, et. al. 2017. *Spring Boot Reference Guide.* 2017.
PostgreSQL. 2016. PostgreSQL. [En línea] 2016. [Citado el: 06 de 11 de 2017.] http://www.postgresql.org.es/.
POSTGRESQL.ORG. 2003. PostgreSQL: File Browser. pgadmin3 . [En línea] 2003. [Citado el: 06 de 11 de 2017.] http://www.postgresql.org/ftp/pgadmin3/. .
Pressman, Roger S. 2010. *Ingeniería de Software, un enfoque práctico.* 7ma Edición. Nueva York : McGraw-Hill Interamericana, 2010. 978-607-15-0314-5.
PUTTE, G.V. de, y otros. 2004. Using Web Services for Business Integration. IBM. s.l. : iBM. ISBN SG24-6583-00, 2004.
Ramírez, Jaime. *Unidad de Programación. Métodos de Prueba del Software.*
Rodriguez, Cesar Andrés González y Omar Almaguer Guerra. 2105. *Capa de servicios web para el*

Bibliografía

Juez en Línea Caribeño. 2105.
Rodríguez, Tamara Sánchez. 2015. *Metodología de desarrollo para la actividad productiva de la UCI.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2015.
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.
SOFT112. 2014. Visual Paradigm for UML Standard 8.0. [En línea] 2014. [Citado el: 3 de 11 de 2017.] http://visual-paradigm-for-uml-standard.soft112.com/.
SOFTWAREENGINEERINSIDER. 2014. What Are Software Engineering CASE Tools? [En línea] 2014. [Citado el: 3 de 11 de 2017.] http://www.softwareengineerinsider.com/articles/case-tools.html.
Sommerville, I. 2007. Software Engineering. [En línea] 8, 2007. http://eva.uci.cu/mod/resource/view.php?id=9269&subdir=/Sommerville_8va_edicion..
Sommerville, Ian. 2005. *Ingeniería del Software 7ma edición.* Madrid, España : Pearson Educación, 2005. ISSN 84-7829-074-5.
TECHTARGET. 2015. What is CamelCase? [En línea] 2015. [Citado el: 24 de 4 de 2018.] http://searchsoa.techtarget.com/definition/CamelCase.
Trends, Google. 2018. Trends, Google. [En línea] 2018. [Citado el: 23 de 3 de 2018.] https://trends.google.com/trends/explore?date=2013-09-01%202018-05-13&q=xml%20api,json%20api.
UCI. 2016. *Metodología de desarrollo para la Actividad productiva de la UCI.* Cuba : s.n., 2016.
—. 2012. Universidad de las Ciencias Informáticas. [En línea] UCI, 2012. [Citado el: 05 de 10 de 2017.] http://www.uci.cu.
W3C. 2008. *Extensible MarKup Language (XML).* [En línea] 2008. [Citado el: 3 de diciembre de 2016.] http://www.w3.org/XML/..
Weerawarana, y otros. 2005. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL,WS-Reliable Messaging, and More.* 2005.

Anexo # 1 Diagrama de clases del diseño

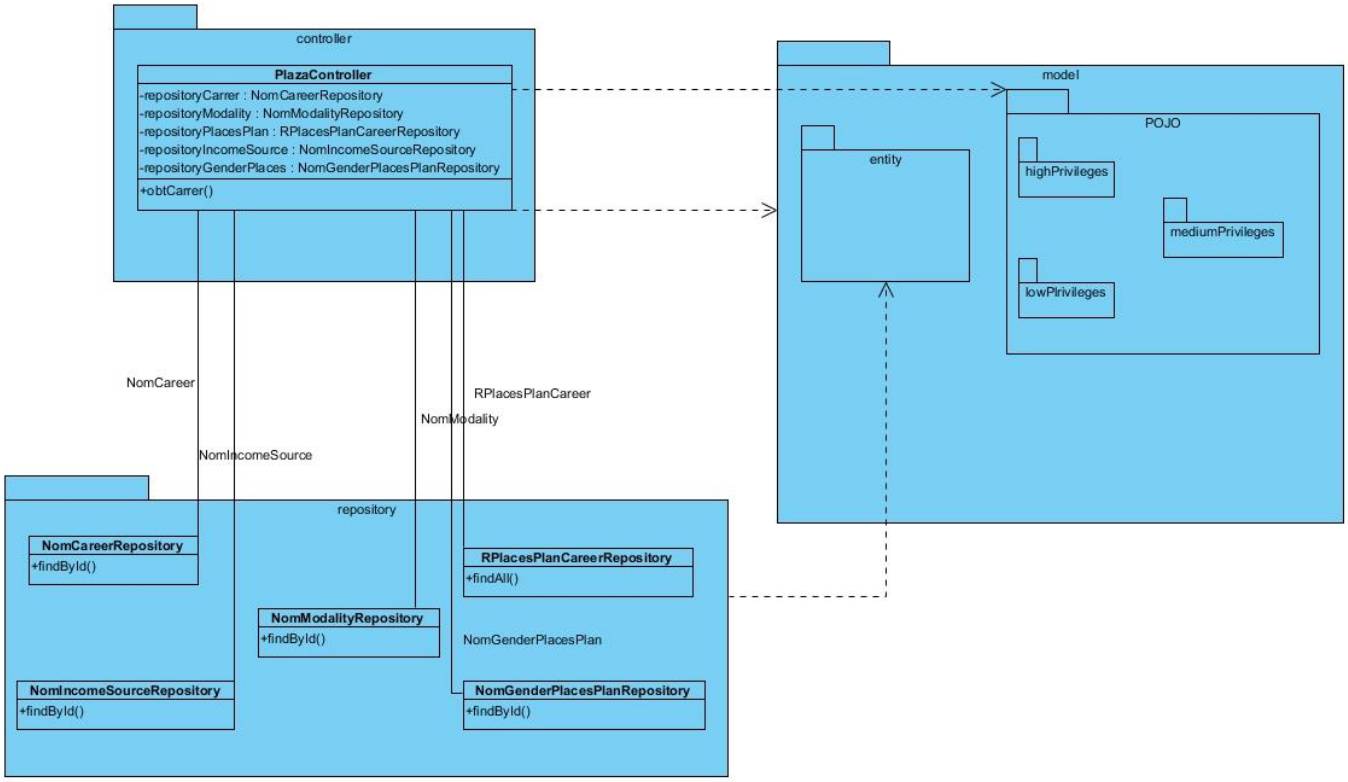


Figura 20 Diagrama de clases del diseño perteneciente a los servicios de plaza

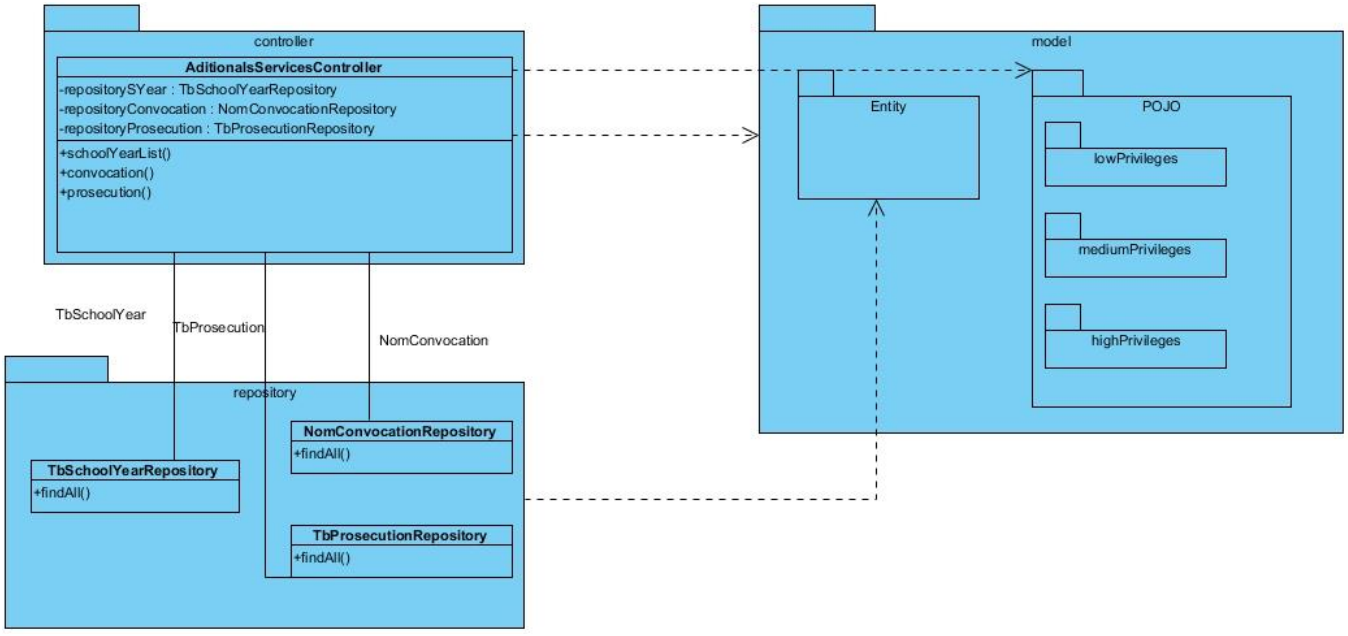


Figura 21 Diagrama de clases del siseño perteneciente a los servicios adicionales

Anexo # 2 Historias de usuarios

Tabla 11 HU_Autenticar usuario

Número: 1	**Nombre del requisito:** Autenticar usuario.
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Autenticar administrador en el sistema de administración.

2- Acciones para lograr el objetivo (precondiciones y datos):

El usuario debe introducir sus credenciales de acceso.

Debe existir el usuario en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El administrador introduce su datos para acceder a las vistas de administración.

Si el proceso se realiza de manera satisfactoria se le da la bienvenida al sistema y se redirecciona hacia la página de inicio.

Si las credenciales para el acceso presentan algún error se notifica.

4- Flujo de la acción a realizar:

- El administrador introduce su usuario y contraseña en el formulario de login.
- El sistema en caso satisfactorio lo envía hacia la página de inicio donde se muestran las acciones que permite ejecutar el sistema.
- En caso de error se le notifica al administrador que ha ocurrido un problema con los datos de entrada y se mantiene la misma en la página de logueo con el fin de que este intente nuevamente el acceso.

Tabla 12 HU_Autorizar cliente

Número: 2	**Nombre del requisito:** Autorizar cliente
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era

Prioridad: Alta	**Tiempo Estimado:** 8 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Autorizar los clientes registrados a acceder a un tipo de información solicitada, validando sus credenciales y permisos de solicitud de servicios.

2- Acciones para lograr el objetivo (precondiciones y datos):

Debe existir el cliente en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El cliente realiza una petición al servidor de autenticación enviando los siguientes datos:

- Clave identificadora de cliente.
- Tipo de contexto.

4- Flujo de la acción a realizar:

- El cliente se autentica con el servidor de autorización usando sus credenciales y solicita un token de acceso para realizar peticiones al servidor de recursos.
- El servidor de autorización autentica al cliente y valida el tipo de autorización, si es válida, emite un token de acceso, del cual se especifica su tipo y tiempo de validez, en caso contrario especifica el tipo de error que se produjo.

Tabla 13 HU_Listar clientes

Número: 3	**Nombre del requisito:** Listar clientes
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Listar los clientes del sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

El administrador debe haberse logueado satisfactoriamente.

Debe existir al menos un cliente en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Se selecciona la opción de listar clientes y se muestra una tabla con todos los clientes del sistema, especificando alguna información perteneciente a los mismos.

4- Flujo de la acción a realizar:

- Se selecciona módulo de gestión de cliente el cual posee un botón con la opción de listar los clientes contenidos en base de datos.

Tabla 14 HU_Registrar cliente

Número: 4	**Nombre del requisito:** Registrar cliente
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 1 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Registrar un cliente en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

El cliente no puede encontrarse almacenado en la base de datos. Para realizar la inclusión debe saberse del nuevo cliente a registrar:

- Client_id

- Secret_id

3- Comportamientos válidos y no válidos (flujo central y alternos):

4- Flujo de la acción a realizar:

- Se selecciona el módulo de gestión de cliente el cual posee un botón con la opción de registrar un nuevo cliente.
- El sistema muestra un formulario con los campos: client_id, secret_id, grant_type, scope, role, resource_id.

Tabla 15 HU_Eliminar cliente

Número: 5	**Nombre del requisito:** Eliminar cliente
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Eliminar un cliente del sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Debe existir una lista de cliente con alenos uno y pulsar la opción eliminar sobre el cual se desea realizar la operación.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Al pulsar el botón de eliminar debe aparecer una ventana para confirmar la acción.

El cliente es eliminado.

4- Flujo de la acción a realizar:

- Se debe dar clic en el botón de eliminar correspondiente al cliente.
- Se muestra en el sistema una ventana de confirmación de eliminación.
- Si se selecciona la opción aceptar, se elimina el cliente de la tabla.
- Si la opción seleccionada es cancelar se mantiene al usuario en la ventana de la lista de clientes.

Tabla 16 HU_Añadir escenario a un cliente

Número: 8	**Nombre del requisito:** Añadir escenario a un cliente
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Agregar escenario a un cliente.

2- Acciones para lograr el objetivo (precondiciones y datos):

El administrador debe estar logueado correctamente.

Debe existir una lista de escenarios.

3- Comportamientos válidos y no válidos (flujo central y alternos):

La tabla que contiene a los clientes muestra en la columna Escenario la opción de añadirle uno nuevo.

Si el cliente contiene todos los escenarios permitidos para él no debe aparecer la opción de agregar uno nuevo.

4- Flujo de la acción a realizar:

- Se selecciona el módulo de gestión de cliente el cual posee menú de selección.
- El sistema muestra un menú de selección que lista los escenarios disponibles para el cliente.

Tabla 17 HU_Añadir autorización a un cliente

Número: 9	**Nombre del requisito:** Añadir autorización a un cliente
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Añadir autorización a un cliente.

2- Acciones para lograr el objetivo (precondiciones y datos):

El administrador debe estar logueado correctamente.

Debe existir una lista de autorización.

3- Comportamientos válidos y no válidos (flujo central y alternos):

La tabla que contiene a los clientes muestra en la columna autorización la opción de añadir un tipo de autorización para el cliente.

Si el cliente contiene un tipo de autorización, no debe aparecer la opción de agregar uno nuevo.

4- Flujo de la acción a realizar:

- Se selecciona el módulo de gestión de cliente el cual posee menú de selección para listar clientes.
- El sistema muestra una tabla que permite añadir autorización al cliente.
- El formulario muestra un botón de tipo selección que lista los tipos de autorización disponibles, de los cuales se selecciona uno para el cliente en cuestión.

Tabla 18 HU_Mostrar estado de procesamiento de examen

Número: 11	**Nombre del requisito:** Mostrar estado de procesamiento de examen
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Mostrar estado de procesamiento de examen.

2- Acciones para lograr el objetivo (precondiciones y datos):

La aplicación cliente debe estar correctamente autenticada y ha de solicitar dicho servicio a través de la URL

definida para el mismo.

Deben existir tipos de procesamientos de examen en la base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Una vez realizada la validación de las credenciales del cliente, los permisos de acceso a datos y el token para la solicitud del servicio, el sistema muestra en formato JSON una estructura de información compuesta por:

- Nombre del procesamiento.
- Identificador del procesamiento.

En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL:

- Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la petición al servidor de recurso.
- Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.

4- Flujo de la acción a realizar:

- El cliente envía sus credenciales al servidor de autenticación especificando en la URL el tipo de autorización que presenta.
- El sistema retorna en caso satisfactorio el tipo correspondiente al token de acceso y la cadena correspondiente al mismo, además del tiempo en que expira el token obtenido. En caso contrario especifica si el error producido proviene de los datos del cliente, el tipo de autorización o permisos de acceso a datos.
- El cliente utiliza el token especificando el tipo del mismo para acceder a la información que necesita del servidor de recurso. Cada petición al servidor estará encabezada por dicho token de acceso.
- El sistema muestra en formato JSON los datos del procesamiento de exámenes.
- Una vez vencido el tiempo de validez, el sistema muestra un mensaje de error al cliente.
- El cliente debe realizar el proceso desde el inicio si desea seguir consultando información.

Tabla 19 HU_Mostrar convocatoria de examen

Número: 12	**Nombre del requisito:** Mostrar convocatoria de examen
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 1 dia
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Mostrar convocatoria del periodo de exámenes de ingreso.

2- Acciones para lograr el objetivo (precondiciones y datos):

La aplicación cliente debe estar correctamente autenticada y ha de solicitar dicho servicio a través de la URL difinida para el mismo.

Debe existir una convocatoria registrada en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Una vez realizada la validación de las credenciales del cliente, los permisos de acceso a datos y el token para la solicitud del servicio, el sistema muestra en formato JSON una estructura de información compuesta por:

- Nombre de la convocatoria.
- Identificador de la convocatoria.

En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL:

- Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la peticion al servidor de recurso.
- Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.

4- Flujo de la acción a realizar:

- El cliente envía sus credenciales al servidor de autenticación especificando en la URL el tipo de autorización que presenta.
- El sistema retorna en caso satisfactorio el tipo correspondiente al token de acceso y la cadena correspondiente al mismo, además del tiempo en que expira el token obtenido. En caso contrario especifica si el error producido proviene de los datos del cliente, el tipo de autorización o permisos de acceso a datos.
- El cliente utiliza el token especificando el tipo del mismo para acceder a la información que necesita del servidor de recurso. Cada petición al servidor estará encabezada por dicho token de acceso.
- El sistema muestra en formato JSON los datos de la convocatoria.
- Una vez vencido el tiempo de validez, el sistema muestra un mensaje de error al cliente.
- El cliente debe realizar el proceso desde el inicio si desea seguir consultando información.

Tabla 20 HU_Obtener si un estudiante esta exonerado de algún examen

Número: 14	**Nombre del requisito:** Obtener si un estudiante esta exonerado de algún examen
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Mostrar si un estudiante está exonerado de un examen de ingreso.

2- Acciones para lograr el objetivo (precondiciones y datos):

La aplicación cliente debe estar correctamente autenticada y ha de solicitar dicho servicio a través de la URL definida para el mismo.

Debe existir el estudiante en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Una vez realizada la validación de las credenciales del cliente, los permisos de acceso a datos y el token para la solicitud del servicio, el sistema muestra en formato JSON una estructura de información compuesta por:

- Nombre de la asignatura.

En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL:

- Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la petición al servidor de recurso.
- Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.

4- Flujo de la acción a realizar:

- El cliente envía sus credenciales al servidor de autenticación especificando en la URL el tipo de autorización que presenta.
- El sistema retorna en caso satisfactorio el tipo correspondiente al token de acceso y la cadena correspondiente al mismo, además del tiempo en que expira el token obtenido. En caso contrario especifica si el error producido proviene de los datos del cliente, el tipo de autorización o permisos de acceso a datos.
- El cliente utiliza el token especificando el tipo del mismo para acceder a la información que necesita del servidor de recurso. Cada petición al servidor estará encabezada por dicho token de acceso.
- El sistema muestra en formato JSON el nombre de la asignatura exonerada.

- Una vez vencido el tiempo de validez, el sistema muestra un mensaje de error al cliente.
- El cliente debe realizar el proceso desde el inicio si desea seguir consultando información.

Tabla 21 HU_Obtener resultado de la asignación de carrera

Número: 15	**Nombre del requisito:** Mostrar resultado de la asignación de carrera
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Mostrar resultado de la asignación de carrera a un estudiante.

2- Acciones para lograr el objetivo (precondiciones y datos):

La aplicación cliente debe estar correctamente autenticada y ha de solicitar dicho servicio a través de la URL definida para el mismo.

Deben existir el estudiante y la carrera en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Una vez realizada la validación de las credenciales del cliente, los permisos de acceso a datos y el token para la solicitud del servicio, el sistema muestra en formato JSON una estructura de información compuesta por:

- Nombre de la carrera otorgada.

En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL:

- Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la petición al servidor de recurso.
- Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.

4- Flujo de la acción a realizar:

- El cliente envía sus credenciales al servidor de autenticación especificando en la URL el tipo de autorización que presenta.
- El sistema retorna en caso satisfactorio el tipo correspondiente al token de acceso y la cadena correspondiente al mismo, además del tiempo en que expira el token obtenido. En caso contrario especifica si el error producido proviene de los datos del cliente, el tipo de autorización o permisos de acceso a datos.
- El cliente utiliza el token especificando el tipo del mismo para acceder a la información que necesita del servidor de recurso. Cada petición al servidor estará encabezada por dicho token de acceso.
- El sistema muestra en formato JSON el nombre de la carrera otorgada.
- Una vez vencido el tiempo de validez, el sistema muestra un mensaje de error al cliente.
- El cliente debe realizar el proceso desde el inicio si desea seguir consultando información.

Tabla 22 HU_Filtrar cliente

Número: 6	**Nombre del requisito:** Editar cliente.
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Editar un cliente del sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Debe existir el cliente en el sistema.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El administrador selecciona la opción editar.

4- Flujo de la acción a realizar:

- El administrador selecciona la opción de editar.
- El sistema muestra un formulario con los campos requeridos para editar al cliente: Contraseña, Confirmar contraseña, Tipo de autorizaciones, Escenarios.
- Además muestra las opciones Aceptar y Cancelar.

Tabla 23 HU_Filtrar cliente

Número: 7	**Nombre del requisito:** Filtrar cliente
Programador: Roberto Verdecia Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	

Descripción:

1- Objetivo:

Filtrar un cliente en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Debe existir el cliente en base de datos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Si el cliente está registrado en el sistema se mostrará dependiendo a la semejanza que posea con respecto a los vales de entrada de los campos de filtro.

4- Flujo de la acción a realizar:

- El sistema filtra por los campos: Nombre de cliente, Tipo de autorización, Escenario.
- El sistema muestra una tabla con los clientes que poseen coincidencias con los valores del filtro.
- En caso de no existir ninguna semejanza con los valores de los clientes del sistema se mantiene la tabla que lista a todos.

Anexo # 3 Diseño de casos de prueba

Tabla 24 DCP_Autenticar usuario

Escenario	**Descripción**	Usuario o correo electrónico	**Contraseña**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Acceder al sistema.	El usuario coloca en el navegador la dirección			El sistema debe permitir la autenticación del usuario a partir de los siguientes datos:	Login

	del sistema e intenta acceder.			- (*) Usuario - (*) Contraseña Y brinda además la opción: - Entrar	
EC 1.2 Opción Entrar.	El usuario introduce los datos para autenticarse y selecciona la opción Entrar.	V	V	El sistema valida los datos, brinda acceso al usuario y muestra la página de inicio.	Login/ Entrar
EC 1.3 Datos incompletos	Existen datos incompletos.	I	V	El sistema señala el/los campo(s) obligatorio(s) que no hayan sido introducidos, estos pueden ser: nombre de usuario y contraseña.	Login/ Entrar
		V	I		

Tabla 25 DCP_Autorizar cliente

Escenario	**Descripción**	Client_Id	Context_type	Tipo de autorización	**Respuesta del sistema**	**Flujo central**
EC 1.1 Acceder al sistema.	El usuario coloca en el navegador la dirección del sistema e intenta acceder.				El sistema debe permitir la autenticación del cliente a partir de los siguientes datos: - (*) Client_Id - (*) Context_type - (*) Client Credentials	/oauth/token?grant_type=client_credentials
EC 1.2 Datos correctos	El usuario resive un token de acceso, tiempo de expiración y tipo de token.	V	V	V	El sistema muestra los datos validos para acceder al servidor de recursos.	/oauth/token?grant_type=client_credentials
EC 1.3 Datos		I	V	V	El sistema muestra en	/oauth/token?grant_type=client_credentials
		V	I	V		

incorrectos	Existen datos incorrectos	V	V	I	formato JSON la ocurrencia de errores en el Client_Id, el tipo de autorizacion o tipo de contexto.	

Tabla 26 DCP_Listar clientes

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar clientes.	El usuario selecciona el módulo de cliente del menú superior y obtiene la lista de clientes contenidos en base de datos.	El sistema debe listar todos los clientes del sistema. En el listado se deben mostrar los siguientes datos: -Id -Rol -Recurso -Autorización -Escenario -Secret Y permite además, realizar las siguientes opciones: -Adicionar cliente -Eliminar cliente	Inicio/ Menú superior/Cliente
EC 1.2 Opción Eliminar cliente.	El usuario selecciona la opción Eliminar cliente.	El sistema permite eliminar un cliente X. Ver DCP_Eliminar cliente.	Inicio/ Menú superior/ Clientes/ Listar/ Elemento X/ Eliminar/Aceptar
EC 1.3 Opción Agregar un nuevo cliente.	El usuario selecciona la opción Agregar cliente.	El sistema permite agregar un nuevo cliente. Ver documento DCP_Añadir cliente.ods	Inicio/ Menú superior/ Cliente/Listar/ Agregar nuevo

Tabla 27 DCP_Registrar cliente

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Adicionar nuevo cliente.	El usuario selecciona la opción Adicionar nuevo cliente al sistema.	El sistema debe permitir adicionar un nuevo tipo de cliente. Los campos para la inclusión de un nuevo cliente son: -Client_ID -Contraseña -Tipo de autorización -Escenarios Y permite además, realizar las siguientes opciones: Cancelar Aceptar	Inicio/Menú superior/Cliente/ Adicionar cliente
EC 1.2 Opción Aceptar.	El usuario introduce los valores de los campos necesarios para la inclusión de un cliente, luego selecciona la opción Aceptar y regresar al listado.	El sistema adiciona adiciona un nuevo cliente y regresa al listado de clientes.	Inicio/Menú superior/Cliente/ Adicionar cliente/ Crear y regresar al listado
EC 1.3 Opción Cancelar	El usuario selecciona la opción Cancelar.	El sistema regresa al listado de clientes.	Inicio/Menú superior/Cliente

Tabla 28 DCP_Eliminar cliente

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Eliminar cliente.	El usuario selecciona la opción Eliminar cliente desde la tabla que los contiene	El sistema realiza una de las siguientes opciones: -Muestra el mensaje de confirmación: *¿Está seguro que desea eliminar el elemento?* Además, permite seleccionar las siguientes opciones: - Aceptar - Cancelar - Cerrar ventana de confirmación	Inicio/ Menú superior/ Clientes/ Listar/Cliente X/ Eliminar

EC 1.2 Opción Aceptar.	El usuario selecciona la opción Aceptar.	El sistema elimina al cliente, actualiza el listado y muestra el siguiente mensaje de información: *El cliente se ha eliminado satisfactoriamente*.	Inicio/ Menú superior/ Clientes/ Listar/Cliente X/ EliminarAceptar
EC 1.3 Opción Cancelar.	El usuario selecciona la opción Cancelar.	El sistema regresa al listado de clientes.	Inicio/ Menú superior/ Clientes/ Listar/Cliente X/ Eliminar/Cancelar
EC 1.4 Opción Cerrar ventana	El usuario selecciona la opción Cerrar ventana.	El sistema regresa al listado de clientes.	Inicio/ Menú superior/ Clientes/ Listar/Cliente X/ Eliminar/Cerrar

Tabla 29 DCP_Editar cliente

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Editar.	El usuario selecciona la opción Editar	El sistema debe permitir editar un cliente a través de los siguientes criterios: -Contraseña -Confirmar contraseña -Tipos de autorización -Escenarios Además el sistema muestra los botones Aceptar y Cancelar.	Inicio/Menú superior/Cliente/search/
EC 1.2 Opción Aceptar	El usuario realiza los cambios en el cliente a editar y selecciona aceptar los cambios.	El sistema valida que no existan campos requeridos sin valores y además que las contraseñas coincidan.	Inicio/Menú superior/Cliente/editar
EC 1.3 Opción Cancelar.	El usuario decide cancelar la edición.	El sistema cancela la acción de editar y muestra la vista de listado de clientes.	Inicio/Menú superior/Cliente/search

Tabla 30 DCP_Filtrar cliente

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Filtrar.	El usuario selecciona la opción Buscar	El sistema debe permitir buscar un cliente a través de los siguientes criterios: -Nombre de cliente -Tipo de autorización -Escenario	Inicio/Menú superior/Cliente/search/
EC 1.2 Buscar por campo: Nombre de cliente.	El usuario introduce el nombre o parte de este	El sistema realiza la búsqueda según el nombre del cliente o coincidencias con lo escrito en el campo de texto.	Inicio/Menú superior/Cliente/search
EC 1.3 Buscar por campo: Tipo de autorización.	El usuario selecciona el tipo de autorización para realizar la búsqueda.	El sistema realiza la búsqueda según el tipo de autorización.	Inicio/Menú superior/Cliente/search
EC 1.4 Buscar por campo: Escenario.	El usuario selecciona el escenario para realizar la búsqueda.	El sistema realiza la búsqueda según el tipo de escenario.	Inicio/Menú superior/Cliente/search

Tabla 31 DCP_Añadir esenario

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Adicionar escenario a un cliente.	El usuario selecciona los escenarios para un cliente del sistema.	El sistema debe permitir adicionar un nuevo tipo de escenario para un cliente a través de un botón de selección. Los escenarios son: -lowPrivileges -highPrivileges -mediumPrivileges -informacionPublica	Inicio/Menú superior/Cliente/Listar clientes/ Escenario

		-informacionSegura -lectura Y permite además, realizar las siguientes opciones: Cancelar Aceptar	

Tabla 32 DCP_Incluir autorización al cliente

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Adicionar autorización.	El usuario selecciona las autorizaciones que desea añadir a un cliente	El sistema debe permitir adicionar un nuevo tipo de autorización para un cliente a través de un botón de selección. Las autorizaciones son: -Implicit -Client_credentials -Autorization_code -Password	Inicio/Menú superior/Cliente/Listar/ autorización

Tabla 33 DCP_Mostrar estado de procesamiento de examen

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Mostrar estado de procesamiento de examen	El cliente solicita una vez autenticado satisfactoriamente, el servicio que muestra el estado de procesamiento de los exámenes, a través de la URL definida para el servicio solicitado.	El sistema muestra una respuesta en formato JSON que contiene los datos: • Nombre del procesamiento. • Identificador del procesamiento. En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL: • Se devuelve el error HTTP 401	/api/service/aditional/school_year

		UNAUTHORIZED, si el token de usuario es incorrecto para realizar la petición al servidor de recurso. • Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.		

Tabla 34 DCP_Mostrar convocatoria de examen

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Mostrar convocatoria de examen	El cliente solicita una vez autenticado satisfactoriamente, el servicio que muestra la convocatoria de examen, a través de la URL definida para el servicio solicitado.		El sistema muestra una respuesta en formato JSON que contiene los datos: • Nombre de la convocatoria. • Identificador de la convocatoria. En caso de ocurrir algún error en el proceso de petición del recurso a través de la URL: • Se devuelve el error HTTP 401 UNAUTHORIZED, si el token de usuario es incorrecto para realizar la petición al servidor de recurso. • Se devuelve el error HTTP 404 NOT FOUND si el identificador del servicio es incorrecto.	/api/service/aditional/convocation