



UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

**TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO
EN CIENCIAS INFORMÁTICAS**

**GENERACIÓN DE ESTADÍSTICAS SOBRE DATOS NO RELACIONALES
(J-ISIS)**


Autoras:

Isbel Torres Bidopia
Idalmis Ferrer Argüelles

Tutores:

Ing. Oigres Alvarez Pérez
Ing. Dayana Rivera Muñoz

La Habana, junio 2016
“Año 58 de la Revolución”



“El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando; lo que más estamos sembrando son oportunidades a la inteligencia (...)”

DECLARACIÓN DE AUTORÍA

Declaramos ser autores del trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas titulado “Generación de Estadísticas sobre datos no relacionales (J-ISIS)” y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Isbel Torres Bidopia

Firma del Autor

Idalmis Ferrer Argüelles

Firma del Autor

Ing. Oigres Alvarez Pérez

Firma del Tutor

Ing. Dayana Rivera Muñoz

Firma del Tutor

DATOS DE CONTACTO

Ing. Dayana Rivera Muñoz: Graduada de Ingeniero en Ciencias Informáticas en el año 2014. Desarrolladora en el proyecto ABCD del Centro de Informatización de la Gestión Documental de la UCI.

Correo electrónico: drmunoz@uci.cu

Ing. Oigres Alvarez Pérez: Graduado de Ingeniero en Ciencias Informáticas en la Facultad 2 de la Universidad de las Ciencias Informáticas (UCI), en el año 2008. Profesor de la misma que labora en el Departamento de Práctica Profesional del Centro de Informatización de la Gestión Documental impartiendo las asignaturas de Proyecto de Investigación y Desarrollo IV, VI y VII. Posee 7 años de experiencia en el desarrollo de aplicaciones empresariales implementadas en java. Actualmente, es arquitecto del proyecto Automatización de Bibliotecas y Centros de Documentación 3.0.

Correo electrónico: oaperez@uci.cu

AGRADECIMIENTOS

Isbel:

Primeramente quisiera agradecer a Dios por haberme dado la oportunidad de cumplir una de mis más grandes metas.

Agradecer a mi compañera de tesis Idalmis por brindarme su amistad, por apoyarme y ayudarme a completar una de mis mayores logros.

A mi tutores Dayana y Oigres por habernos acompañado en esta travesía, por habernos guiado, por estar siempre ahí cuando necesitamos de su ayuda.

A mi mamá Sonia por brindarme todo su cariño incondicional, por tratarme como a una hija, gracias le doy a la vida por permitirme conocer a una persona tan especial como ella.

A mi Mocosó (Lachy), por brindarme su cariño y haber echo de estos últimos años en la universidad un recuerdo inolvidable para mí. Te quiero.

A mi mejor amiga Yula, por ser como una hermana para mí y estar siempre cuando he necesitado de su apoyo a pesar de la distancia.

A mi Peluza (Grisel), por ser mi prueba de paciencia, por cuidarme y siempre estar conmigo en las buenas y en las malas.

A mi mejor amigo Niharby, por haberme brindado desde el primer momento su amistad incondicional, por ser como un hermano mayor para mí.

A mis putís, Lientz y Rosmelys por acogerme como su mojí, por escucharme siempre y aconsejarme cuando más lo he necesitado.

A mi amigo Ilsen, por brindarme su amistad y su cariño.

A mis compañeros de aula, gracias a todos por acompañarme en esta aventura, que es la universidad.

En especial quiero agradecer a mi familia por haberme apoyado desde el primer día, por permitirme tomar siempre mis propias decisiones. Por ser el motivo que me mantuvo firme a lo largo de mi carrera, a todos ustedes. Gracias!!!

AGRADECIMIENTOS

Idalmís:

Agradezco a todas las personas que de una forma u otra colaboraron con el desarrollo de este trabajo.

A todos los profesores que contribuyeron en mi formación como profesional en esta casa de altos estudios.

A mis tutores Dayana y Oigres por la guía, dedicación y ayuda que me brindaron para poder llegar al final de la carrera.

A los viejos y nuevos amigos que durante todo el recorrido de estos largos años me apoyaron y estuvieron presentes en los buenos y malos momentos.

A mi compañera de tesis Isbel por apoyarme y por estar juntas en las buenas y las malas situaciones, y por ser buena amiga.

A mi compañera de cuarto Yadís por el apoyo en esto últimos años y ser una buena amiga.

En especial a mi familia por darme la fuerza y el apoyo necesario cada vez que la necesitaba para poder cumplir mi sueño.

DEDICATORIA

Isbel:

Este logro va dedicado a toda mi familia por haber sido mi motor impulsor:

En especial va dedicado a mi madre, que en paz descanse, por darme la vida sé que hoy estarías orgullosa de verme graduada.

A mi Gruñoncito por ser un ejemplo de padre, de sacrificio y dedicación. Quiero que sepas que estoy orgullosa de tenerte como padre, te amo.

A mi Abuela Delvis, por ser lo más grande que tengo en la vida, que no he necesitado nunca decir ni una sola palabra para que sepas lo que me pasa, a ti que me has abierto tus brazos cuando el mundo me ha dado la espalda. A ti van dedicados mis logros.

A mi Tía Milagro, por ser más que una tía para mí por ser mi amiga incondicional. Por ser mi ejemplo a seguir, por haber sido la primera persona que me apoyó cuando tuve que repetir.

A mi madrastra Bety, por acogerme como a una hija. Por tratarme igual que a mis hermanos, por estar siempre pendiente de mis problemas y mis alegrías a lo largo de estos 23 añitos, por brindarme tu apoyo incondicional, gracias.

A mi hermano Osva por ser mi BB consentido, porque a pesar de ser mi hermano pequeño me cuidas como si fueras mayor. A mi hermanastra Betsy por tratarme como a una hermana, y por sobre todas las cosas por darme el sobrino más lindo del mundo Bryan.

A mi abuelo Daer y a mis abuelas Ramona, Yolanda y Nirian, por consentirme, cuidarme y brindarme todo su amor.

A mis primos Jose, Roger, Pacho, Robert, Marquito, Alejandrino, María Carla, María Eduarda y Kriss, por ser una parte fundamental en mi vida.

A mis Tíos Lázaro, Nérida, Mongo, Manolo y Rigo por protegerme y consentirme siempre.

DEDICATORIA

Idalmís:

Este trabajo va dedicado a mi familia:

A mi abuelo Tete esté donde esté, ya que fue quien insistió que estudiara y me hiciera alguien en la vida. Por no estar presente en este momento, honro su memoria dedicándole mi logro alcanzado y me despido de ti con nuestros sueños hechos realidad. A mis abuelos Edelsa y Manolo, y a mi abuela Xena que donde quieran que estén se sientan orgullosos de mí.

A mi abuelo Ramón por ser ejemplo de padre trabajador para sus hijos y nietos.

A mis hermanas Inalvís e Ilíanet por apoyarme, quererme y por ser mis ejemplos de mujeres que luchan por lo que quieren.

A mi sobrino Javier para que vea en mí un ejemplo a seguir.

A mi padrastro Joel por ser como un padre para mis hermanas y yo, por su apoyo durante todo este tiempo.

A mi papá Erik y a mis hermanos Annerys y Erito.

A mi esposo Eduardo por apoyarme, quererme, aguantarme todos estos años, por confiar en mí, por ser mi soporte, mi estímulo para seguir adelante y más que todo por ser mi amigo y confidente. Gracias por cuidar de nuestro pequeño tesorito, Rodriguito. Te amo.

En especial a la madre más linda y trabajadora del mundo, a mi mamá Sonia por su dedicación, apoyo cuando más lo necesité, por su esfuerzo por verme salir adelante, por quererme incondicionalmente, por cuidarme y guiarme en los caminos de la vida, y hacerme hacer de mí una mejor persona y quien soy hoy en día. Te quiero.

Por último le dedico mi trabajo a la personita más importante de mi vida, a mi enano Eduardo Rodrigo por él logré terminar hoy mi sueño, porque es quien me da la fuerza para luchar cada día y ver el mundo diferente. Es quien llena mi vida de felicidad infinita y todo lo que hice y hago es pensando en su bienestar.

RESUMEN

El desarrollo actual de las tecnologías ha convertido a los Sistemas de Gestión Documental en elementos indispensables para el tratamiento de la información que se almacena en las bibliotecas universitarias. En la actualidad los SIGB desarrollados, pueden incluir un módulo estadístico que permita realizar búsquedas y obtener estadísticas referentes a los registros bibliográficos. El proyecto Automatización de Bibliotecas y Centros de Documentación, perteneciente al Centro de Informatización de la Gestión Documental de la Universidad de las Ciencias Informáticas, desarrolla el Sistema Integrado de Gestión Bibliotecaria ABCD v3.0 que gestiona los principales procesos que se realizan en una biblioteca.

La presente investigación surge de la necesidad de generar estadísticas dinámicas, mediante el cruzamiento de dos variables en el módulo Estadística del sistema ABCD v3.0, que facilite la toma de decisiones. Para el desarrollo de la solución propuesta se hizo uso de la metodología, herramientas y tecnologías de desarrollo de software previamente definidas por el proyecto ABCD. Una vez concluido el proceso de desarrollo se obtuvo como resultado un componente estadístico, que permite generar estadísticas dinámicas sobre la información almacenada en las bases de datos de J-ISIS.

Palabras clave: Biblioteca, estadística e información.

ÍNDICE

INTRODUCCIÓN	1
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA	5
1.1 INTRODUCCIÓN	5
1.2 DEFINICIÓN DE ESTADÍSTICA	5
1.2.1 Tipos de estadísticas	5
1.2.2 Variables estadísticas	6
1.3 BASES DE DATOS NO RELACIONALES	7
1.3.1 Características	7
1.3.2 Aplicaciones de las bases de datos NoSQL	7
1.4 ANÁLISIS DE LOS SISTEMAS EXISTENTES	8
1.4.1 Sistema Koha	8
1.4.2 Sistema PhpMyBibli o PMB	9
1.4.3 Sistema ABCD v1.2	9
1.5 ISIS	11
1.6 SISTEMA ABCD V3.0	12
1.7 GESTOR DE BASES DE DATOS NO RELACIONAL J-ISIS-SUITE (25 AGOSTO 2014)	13
1.8 BERKELEY DB	14
1.9 CONSULTAS LUCENE	14
1.9.1 Funcionalidades básicas de Lucene	15
Estructura de una consulta Lucene empleada en J-ISIS:	15
1.10 OPEN SERVICES GATEWAY INITIATIVE (OSGi)	16
1.10.1 Bundle	18
1.11 METODOLOGÍA, HERRAMIENTAS, LENGUAJES Y TECNOLOGÍAS	19
1.11.1 Metodología de desarrollo de software Rational Unified Process (RUP)	19
1.11.2 Lenguaje Unificado de Modelado (UML) v2.1	21
1.11.3 Entorno de desarrollo integrado (Eclipse-Juno 4.2.2)	21
1.11.4 Herramienta CASE Visual Paradigm v8.0	21
1.11.5 Lenguaje de Programación Java v7.0	22
1.11.6 Servidor de Aplicaciones Virgo v3.6.3	22
1.11.7 Herramientas de automatización de pruebas de software	23
1.12 CONCLUSIONES PARCIALES	23
CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL COMPONENTE	25

2.1 INTRODUCCIÓN	25
2.2 MODELO DEL DOMINIO.....	25
2.3 REQUISITOS FUNCIONALES	26
2.4 REQUISITOS NO FUNCIONALES.....	26
2.5 DIAGRAMA DE CASOS DE USO	27
2.5.2 DESCRIPCIÓN DE LOS CASOS DE USO (CU).....	28
2.6 ARQUITECTURA DEL COMPONENTE	32
2.7 PATRONES DE DISEÑO UTILIZADOS EN EL DESARROLLO DEL COMPONENTE ESTADÍSTICO	33
2.7.1 Patrones GRASP	33
2.7.2 Patrones GOF	34
2.8 PROPUESTA DE SOLUCIÓN.....	34
2.8.1 Modelo de diseño	35
2.9 CONCLUSIONES PARCIALES	40
CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA	41
3.1 INTRODUCCIÓN	41
3.2 MODELO DE IMPLEMENTACIÓN	41
3.2.1 Diagrama de Despliegue	41
3.2.2 Diagrama de componente	42
3.3 ESTÁNDARES DE CODIFICACIÓN	43
3.4 IMPLEMENTACIONES RELEVANTES EN JAVA.....	43
2.4.1 Tratamiento de excepciones.....	47
3.5 ESTRATEGIAS DE PRUEBAS.....	48
3.6 PRUEBAS DE SOFTWARE	48
3.6.1 Tipos de pruebas usados	48
3.7 PRUEBAS UNITARIAS	49
3.7.1 Resultado de las pruebas unitarias.....	50
3.8 PRUEBAS DE RENDIMIENTO.....	52
3.8.1 Resultados de las pruebas de rendimiento.....	53
3.9 CONCLUSIONES PARCIALES	55
CONCLUSIONES GENERALES	56
RECOMENDACIONES	57
REFERENCIAS BIBLIOGRÁFICAS	58
BIBLIOGRAFÍA	60
ANEXOS	63

ÍNDICE DE FIGURAS

Figura 1. Esquema de cómo una aplicación usa Lucene (13).....	15
Figura 2. Resultado mostrado por J-ISIS al realizar una consulta simple.	16
Figura 3. Resultado mostrado por J-ISIS al realizar una consulta compuesta	16
Figura 4. Arquitectura en capas de OSGi (14).....	17
Figura 5. Ciclo de vida de un bundle (13).....	19
Figura 6. Fases de RUP (16).....	20
Figura 7. Diagrama del modelo del dominio (Fuente: Elaboración propia).....	26
Figura 8. Diagrama de casos de uso del sistema (Fuente: Elaboración propia)	28
Figura 9. Dependencia entre las capas de la arquitectura	32
Figura 10. Diagrama de Paquetes (Fuente: Elaboración propia)	35
Figura 11. Diagrama de clases del diseño (Fuente: Elaboración propia).	36
Figura 12. Clase TabularStatistic (Fuente: Elaboración propia).....	37
Figura 13. Clase JISISStatisticProviderImpl (Fuente: Elaboración propia).....	38
Figura 14. Clase StatisticParam (Fuente: Elaboración propia).....	39
Figura 15. Clase SumarizedOperation (Fuente: Elaboración propia).....	39
Figura 16. Clase CrossStatisticImpl (Fuente: Elaboración propia).....	40
Figura 17. Diagrama del despliegue (Fuente: Elaboración propia)	41
Figura 18. Diagrama de componentes (Fuente: Elaboración propia).....	42
Figura 19. Implementación del método checkIndexTags (Fuente: Elaboración propia)	44
Figura 20. Implementación del método setHeaders (Fuente: Elaboración propia).	45
Figura 21. Implementación del método buildValue (Fuente: Elaboración propia).....	47
Figura 22. Ejemplo del tratamiento de excepciones (Fuente: Elaboración propia).....	48
Figura 23. Prueba con JUnit primera iteración (Fuente: Elaboración propia)	51
Figura 24. Prueba con JUnit segunda iteración (Fuente: Elaboración propia)	51
Figura 25. No conformidades obtenidas en las pruebas unitarias (Fuente: Elaboración propia)	52
Figura 26. Resultado de la prueba de carga en la primera iteración (Fuente: Elaboración propia)	53
Figura 27. Resultado de la prueba de carga en la segunda iteración (Fuente: Elaboración propia).....	53
Figura 28. Resultado de la prueba de carga en la tercera iteración (Fuente: Elaboración propia)	54
Figura 29. Excepción mostrada por J-ISIS al realizar peticiones concurrentes (Fuente: Elaboración propia).....	54
Figura 30. Módulo Informes del sistema Koha.....	64
Figura 31. Creación de una nueva tabla en el sistema ABCD v1.2.....	64
Figura 32. Uso de una tabla existente en el sistema ABCD v1.2	65

Figura 33. Generar salida en el módulo estadística del sistema ABCD v1.2..... 65

ÍNDICE DE TABLAS

<i>Tabla 1. Aplicaciones de las bases de datos NoSQL (7)</i>	7
<i>Tabla 2. Familia extendida del software ISIS (10)</i>	11
<i>Tabla 3. Resumen de las características de las tecnologías básicas (10)</i>	12
<i>Tabla 4. Descripción de las capas de la arquitectura OSGi (13)</i>	17
<i>Tabla 5. Conceptos y descripción del modelo del dominio (Fuente: Elaboración propia)</i>	25
<i>Tabla 6. Requisitos funcionales y sus descripciones (Fuente: Elaboración propia)</i>	26
<i>Tabla 7. Descripción del autor (Fuente: Elaboración propia)</i>	28
<i>Tabla 8. Descripción del CU_Realizar búsqueda por expresión de búsqueda (Fuente: Elaboración propia)</i>	28
<i>Tabla 9. Descripción del CU_Realizar búsqueda por rango MFN (Fuente: Elaboración propia)</i>	29
<i>Tabla 10. Descripción del CU_Cruzamiento de dos variables (Fuente: Elaboración propia)</i>	30
<i>Tabla 11. Iteraciones para el caso de prueba "Rango MFN válido" (Fuente: Elaboración propia)</i>	50
<i>Tabla 12. Iteraciones para el caso de prueba "Rango MFN negativo" (Fuente: Elaboración propia)</i> ..	50
<i>Tabla 13. Entrevista</i>	63

INTRODUCCIÓN

La historia de la humanidad se basa en la recopilación de los conocimientos adquiridos a través de los años. La necesidad de salvaguardar y organizar todo el patrimonio que conforma la información, fue el detonante para el surgimiento de las primeras bibliotecas. Las bibliotecas, conjuntamente con los centros de documentación, se han convertido en uno de los organismos fundamentales en la difusión de los documentos y la información, siendo además centros de recursos para el aprendizaje, la docencia y la investigación. Los centros de documentación son unidades de información que reúnen, gestionan y difunden la documentación de un área del conocimiento determinado o la producida por un organismo o institución a la que se circunscribe.

El uso de las Tecnologías de la Información y las Comunicaciones (TIC), permiten gestionar el creciente volumen de información que se almacena en las bibliotecas universitarias. La necesidad de optimizar los procesos de una biblioteca dio paso a la creación de los Sistemas Integrados de Gestión Bibliotecaria (SIGB). Un SIGB se define como (...) *“un grupo de programas informáticos interrelacionados, que automatizan múltiples operaciones y funciones bibliotecarias basadas en datos centralizados e intercambiables; esto con el objetivo de facilitar la gestión de las actividades llevadas a cabo en una biblioteca”* (...) (1). El uso de un SIGB en una biblioteca universitaria provee las siguientes ventajas: no redundancia de los datos e información, consistencia de la información, coste mínimo de la actualización de los datos de la organización, entre otras. Ofrece además la posibilidad de gestionar de forma unificada y cómoda, para el bibliotecario, todos los recursos informativos utilizados en la biblioteca.

En la actualidad los SIGB desarrollados, incluyen un módulo Estadística que permite realizar búsquedas y obtener estadísticas referentes a los registros bibliográficos. Un ejemplo lo constituye el sistema Automatización de Bibliotecas y Centros de Documentación v1.2 (ABCD), aunque existen otros que son abordados con posterioridad en la presente investigación. El sistema ABCD v1.2, a pesar de encontrarse en explotación en varias instituciones del país, presenta problemas que pueden dificultar la actividad del usuario. Con el objetivo de corregir los errores detectados (presenta errores de validación, la forma en que está desarrollado el sistema no permite la escalabilidad, utiliza como gestor de base de datos C-ISIS), agregar nuevas funcionalidades y definir una arquitectura que se ajuste a los intereses actuales surge la idea de desarrollar la versión 3.0 del sistema.

La Universidad de las Ciencias Informáticas (UCI) (...) *“es un centro docente-productor que desarrolla aplicaciones y servicios informáticos orientados a diversos sectores de la economía y los servicios, dentro y fuera del país”* (...). Como modelo de buenas prácticas del quehacer productivo en la UCI, se

encuentra el Proyecto ABCD, perteneciente al Centro de Informatización de la Gestión Documental. En el proyecto ABCD, se implementa el SIGB ABCD v3.0 que informatiza los procesos que realiza una biblioteca.

En el sistema ABCD v3.0 la información, referente a los registros bibliográficos, es gestionada mediante la familia CDS/ISIS (Servicios de Informatización Documentada / Conjunto Integrado de Sistemas de Informatización). Parte de la información procesada en los módulos Adquisición, Catalogación, Catálogo en Línea de Acceso Público (OPAC) y Estadística es gestionada en bases de datos no relacionales, con J-ISIS.

J-ISIS, como gestor de bases de datos almacena los datos como registros. Un registro puede tener uno o más campos de diferentes tipos, tales como: alfanumérico, alfabéticos, numéricos, cadena, booleano, char, byte, int, float, doble, fecha y hora. Para identificar cada tipo de campo se le asigna un número único. Una de las ventajas que ofrece J-ISIS es que puede manipular un número ilimitado de bases de datos, cada una de las cuales pueden tener elementos completamente diferentes.

De acuerdo a los requisitos funcionales del sistema, surge la necesidad de generar estadísticas dinámicas, referentes a los registros bibliográficos, mediante el cruzamiento de dos variables cualesquiera del conjunto de datos almacenado en cada registro bibliográfico. El módulo Estadística, del sistema ABCD v3.0, no cuenta con la posibilidad de generar estadísticas dinámicas con cruzamiento de dos variables, no realiza búsquedas por rango MFN o por expresión de búsqueda sobre las bases de datos de J-ISIS; lo que trae consigo que no se pueda realizar un análisis de los datos existentes en las bases de datos, dificultando la toma de decisiones.

Debido a la situación antes mencionada, se plantea como **problema a resolver**: ¿Cómo realizar cálculos estadísticos dinámicos de los datos bibliográficos, almacenados en bases de datos no relacionales en el sistema ABCD v3.0?

Se establece como **objeto de estudio** la generación de estadísticas sobre los datos bibliográficos.

Enmarcada la investigación en el **campo de acción**: generación de estadísticas de los datos bibliográficos en el sistema ABCD v3.0.

Para dar solución al problema anteriormente mencionado se propone como **objetivo general**: desarrollar un componente para la generación de estadísticas dinámicas para el sistema ABCD v3.0.

Para dar cumplimiento al objetivo general se proponen los siguientes **objetivos específicos**:

- Realizar el estudio del estado del arte en sistemas que realicen la generación de estadísticas

sobre los registros bibliográficos.

- Realizar el análisis y diseño del componente estadístico para el sistema ABCD v3.0.
- Realizar la implementación y las pruebas al componente estadístico del sistema ABCD v3.0.

En aras de dar cumplimiento al objetivo planteado se trazaron las siguientes **tareas de investigación**:

- Análisis de los principales conceptos relacionados con el acceso a la información en bases de datos no relacionales, para obtener la base teórica necesaria para el desarrollo de la solución.
- Investigación sobre formas de obtener información a partir de datos almacenados en bases de datos no relacionales con J-ISIS.
- Estudio del estado del arte en sistemas que realicen la generación de estadísticas sobre los registros bibliográficos, para un mejor entendimiento del componente a desarrollar.
- Estudio del lenguaje de programación, metodología, herramientas y tecnologías de desarrollo para la realización del componente.
- Validación de las pruebas unitarias y de rendimiento para evaluar el funcionamiento del componente.

Para el desarrollo de esta investigación fueron utilizados diferentes **métodos y técnicas de investigación**, los cuales se especifican a continuación:

Métodos teóricos: permiten revelar las relaciones esenciales del objeto de estudio, son fundamentales para la comprensión de los hechos y para la formulación de la hipótesis de investigación.

- **Analítico-sintético:** consiste en la separación de las partes de un todo para estudiarlas en forma individual, y la unión relacional de elementos dispersos para estudiarlos en su totalidad. Este método fue utilizado para el análisis del lenguaje de programación, metodología, herramientas y tecnologías de desarrollo empleadas, identificando las características que las distinguen.
- **Modelación:** es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. Con este método se mostró la estructura, relación y características de la solución, a través de diagramas, para el diseño y la comprensión de la implementación del componente estadístico.

Técnica de investigación empírica:

- **Entrevista:** constituye una conversación planificada entre el investigador y el entrevistado para obtener información. Esta técnica de investigación fue utilizada para definir el alcance de la investigación e identificar las funcionalidades que debe tener el componente (Ver Anexo 1).

El documento cuenta con una estructura de 3 capítulos:

Capítulo 1. Fundamentación teórica: en este capítulo se presenta la definición conceptual del marco teórico de la investigación. Se realiza el estudio de la generación de las estadísticas de los registros bibliográficos en los SIGB. Se describe el lenguaje de programación, la metodología, herramientas y tecnologías de desarrollo utilizadas para el desarrollo del componente.

Capítulo 2. Análisis y diseño del componente: en este capítulo se describe la propuesta de solución del componente. Se muestra la arquitectura empleada para el desarrollo del componente. Se definen los requisitos funcionales y no funcionales, y se elabora el conjunto de artefactos que se genera durante la fase de análisis y diseño.

Capítulo 3. Implementación y prueba: en este capítulo se describe la implementación del componente, así como las garantías de su funcionamiento. Finalmente se muestran los resultados obtenidos en la ejecución de las pruebas realizadas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo, se hace referencia al marco conceptual, citando los principales conceptos relacionados con el tema propuesto. Se analiza la generación de estadísticas sobre los registros bibliográficos en SIGB. Se describe además el lenguaje de programación, la metodología, herramientas y tecnologías empleadas en el desarrollo de la aplicación.

1.2 Definición de Estadística

Desde los inicios de la humanidad se utilizan las herramientas cuantitativas para el tratamiento de datos. El empleo de la estadística en las antiguas civilizaciones como la egipcia, consistía en aplicar censos que ayudaban a la organización del estado y la construcción de las pirámides. La palabra “**estadística**” viene del latín “*statisticus*” que significa “del estado”. La enciclopedia británica define la estadística como la ciencia encargada de recolectar, analizar, presentar e interpretar datos. Sin embargo el diccionario inglés *Word Reference* define la estadística como un área de la matemática orientada a la recolección e interpretación de datos cuantitativos y al uso de la teoría de la probabilidad para calcular los parámetros de una población (2).

La estadística facilita la toma de decisiones, mediante la presentación ordenada de los datos observados en tablas y gráficos estadísticos. Permite además estimar la probabilidad del éxito que tiene cada una de las posibles decisiones. Ofrece la posibilidad de realizar cálculos para la comparación de resultados.

1.2.1 Tipos de estadísticas

Estadística descriptiva: consiste en la presentación de datos en forma de tablas y gráficas. Comprende cualquier actividad relacionada con los datos. La representación de los datos se realiza mediante la ordenación en tablas (proceso denominado tabulación) y su posterior representación gráfica. La estadística descriptiva desarrolla un conjunto de técnicas para la presentación y reducción de datos. La reducción estadística consiste en utilizar solo un número reducido de los datos posibles para facilitar las operaciones estadísticas. Esta reducción conlleva a un error que debe ser controlado, puede realizarse previamente durante el proceso de tabulación o, con mayor eficacia, utilizando las llamadas medidas estadísticas.

La utilización de las medidas estadísticas permite comparar diferentes series de datos obtenidos en distintas observaciones. La estadística descriptiva desarrolla además técnicas que estudian la dependencia que puede existir entre dos o más características observadas en una serie de individuos. Estas son las denominadas técnicas de regresión y correlación. Mientras la técnica de regresión estudia

la posible predicción de los valores de una variable a partir de otra, la correlación estudia el tipo de dependencia que existe entre ambas variables, intentando cuantificarla mediante el cálculo de coeficientes de correlación (3).

Estadística Inferencial: se deriva de muestras, observaciones hechas sólo a una pequeña parte de un conjunto numeroso de elementos y esto implica que su análisis requiere de generalizaciones que van más allá de los datos. Como consecuencia, la característica más reciente es que sirven para hacer generalizaciones. La estadística inferencial investiga o analiza una población partiendo de una muestra tomada (4).

De acuerdo a sus características, para la implementación del componente estadístico, se utilizó la estadística descriptiva. Este tipo de estadísticas permitió crear y diseñar un objeto tabular para generar las estadísticas cruzadas en el componente y su posterior representación.

1.2.2 Variables estadísticas

Una variable estadística es una propiedad que puede variar y cuya variación es susceptible de medirse u observarse” (5). Existen tres tipos de variables estadísticas: cualitativas, cuantitativas y bidimensionales. Las variables cualitativas se clasifican a su vez en nominales y ordinales; y las variables cuantitativas se clasifican en discretas y continuas. Para la implementación del componente estadístico se hace uso de las variables estadísticas bidimensionales.

Variables estadísticas cualitativas: son aquellas que no se pueden medir numéricamente (5).

- **Nominales:** son datos que corresponden a categorías que por su naturaleza no admiten un orden.
- **Ordinales:** son aquellas que corresponden a evaluaciones subjetivas que se pueden ordenar o jerarquizar.

Variables estadísticas cuantitativas: son aquellas que tienen valor numérico (5).

- **Discretas:** son aquellas que sólo pueden tomar valores enteros.
- **Continuas:** son aquellas que pueden tomar cualquier valor real dentro de un intervalo o rango.

Variable estadística bidimensional: una variable estadística bidimensional es el conjunto (X, Y) de valores, que pueden tomar dos caracteres diferentes X e Y medidos sobre cada uno de los individuos de una población o muestra. Los caracteres X e Y se denominan caracteres o variables marginales y pueden ser a su vez variables estadísticas cualitativas o cuantitativas (6).

1.3 Bases de datos no relacionales

Las bases de datos NoSQL (*Not Only SQL/ No solo SQL*) son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación de las bases de datos relacionales. No imponen una estructura de datos en forma de tablas y relaciones entre ellas. Permiten almacenar información en otros formatos como clave-valor similar a tablas *hash*, documentos o grafos. Las bases de datos NoSQL están concebidas para obtener una alta capacidad de volumen de almacenamiento y velocidad de procesamiento de la información (7).

1.3.1 Características

Consistencia eventual: no se implementan mecanismos rígidos de consistencia como los presentes en las bases de datos relacionales, donde la confirmación de un cambio implica una comunicación del mismo a todos los nodos que lo repliquen (7).

Escalabilidad horizontal: la implementación se realiza en muchos nodos de capacidad de procesamiento limitado, en vez de utilizar grandes *mainframes* (7).

Escalabilidad horizontal: la posibilidad de aumentar el rendimiento del sistema añadiendo más nodos (7).

1.3.2 Aplicaciones de las bases de datos NoSQL

En la Tabla 1 se presenta un conjunto de tipos de bases de datos NoSQL y sus aplicaciones.

Tabla 1. Aplicaciones de las bases de datos NoSQL (7)

Tipos de bases de datos	Aplicaciones
Bases de datos documentales	<ul style="list-style-type: none">➤ CouchDB, de Apache➤ MongoDB.➤ RavenDB, de Hibernating Rhinos➤ BaseX➤ eXist➤ SimpleDB➤ IBM Lotus Domino➤ Terrastore
Bases de datos en grafo	<ul style="list-style-type: none">➤ Neo4j➤ DEX➤ AllegroGraph

	<ul style="list-style-type: none"> ➤ OrientDB ➤ InfiniteGraph ➤ Sones GraphDB ➤ InfoGrid ➤ HuperGraphDB
Bases de datos clave/valor	<ul style="list-style-type: none"> ➤ Cassandra, de Apache ➤ BigTable, de Google ➤ Dynamo, de Amazon ➤ Project Voldemort, de LinkedIn ➤ Riak ➤ Berkeley
Bases de datos multivalor	<ul style="list-style-type: none"> ➤ Db4o ➤ GemStone S ➤ Objectivity/DB
Bases de datos tabular	<ul style="list-style-type: none"> ➤ Hbase, de Apache ➤ BigTable, de Google ➤ Hypertable

1.4 Análisis de los sistemas existentes

Con el objetivo de lograr un mejor entendimiento del proceso de generación de estadísticas en los SIGB, se llevó a cabo un estudio del módulo Estadística de varios sistemas utilizados a nivel nacional e internacional. A continuación, se muestra la relación de las funcionalidades del módulo estadística de cada uno de ellos:

1.4.1 Sistema Koha

Es un SIGB de código abierto que tiene licencia GPL (Licencia General Pública). Cumple con los estándares internacionales (MARC y Z39.50 XHTML 1.0 y CSS de la World Wide Web Consortium (W3C)) y se puede tener acceso desde cualquier computador o terminal con acceso a Internet. Koha funciona con una arquitectura cliente servidor, utilizando GNU/Linux, Apache, MySQL, Perl y OpenLDAP en el servidor (8).

En Koha el módulo encargado de realizar las estadísticas es el módulo Informes (Ver Anexo 2). Existen dos posibilidades de obtener informes: por un lado se encuentran los informes predefinidos que facilitan informes genéricos de uso (Adquisiciones, Usuarios, Catálogo, Préstamos), informes de inactividad (usuarios que no realizan préstamos, libros no prestados, etc.), *Top list* (usuarios que más préstamos realizan, los libros más prestados, etc.) u otros informes (libros perdidos, catálogo por tipo de

documento, préstamos por tipo de usuario, etc.). Por otro lado se facilita la confección de informes más personalizados mediante un asistente para informes, desde el mismo el usuario selecciona qué datos quiere, cómo quiere que se visualicen, etc.

El tipo de reporte que usa Koha es tabular. Permite seleccionar varios campos o todos a la vez. Da la opción de realizar operaciones matemáticas con los elementos de las filas y las columnas. Permite además escoger el orden en que se desean imprimir los datos. Una vez seleccionadas las opciones se presenta la consulta generada por el asistente de informes. Esta consulta puede ser guardada o copiada para hacerle cambios de forma manual. Además el usuario puede escribir sus propias consultas SQL (Ver Anexo 2).

1.4.2 Sistema PhpMyBibli o PMB.

PMB es un sistema de código abierto con la licencia GNU GPL diseñado para trabajar en los sistemas operativos Windows y Linux. El sistema operativo ya debe tener instalado los siguientes programas: Un servidor Apache, el administrador de bases de datos MySQL, lenguaje de programación PHP y un navegador web. Cumple con estándares como el MARC21 y permite la búsqueda de registros bibliográficos mediante el protocolo Z39.50 (9).

A los procesos predeterminados con los que cuenta PMB, se le pueden añadir nuevos procedimientos. Estos son creados en el módulo Informes, encargado de realizar las operaciones estadísticas. La estadística puede ser realizada mediante la herramienta básica de selección de campos y valores de PMB. Si se desean realizar operaciones de mayor complejidad, es necesario generar una consulta SQL con los parámetros que permitan ejecutar una operación que devuelva valores u opciones para lograr obtener los valores requeridos.

En el módulo Informes se puede acceder a estadísticas sobre préstamos, reservas, usuarios, entre otros. Una de las ventajas que brinda este módulo de PMB es que si se necesita una estadística o listado que no está contemplada en la instalación por defecto, se pueden personalizar las estadísticas o listados propios. Además en Informes se pueden hacer cambios en la visualización de los registros del catálogo, se puede generar una nueva plantilla (template) para los registros (notices), (Ver Anexo 3).

1.4.3 Sistema ABCD v1.2

ABCD v1.2 es un SIGB de código abierto que realiza las funciones principales de una biblioteca, es decir, adquisiciones, catalogación, gestión de préstamos, control de publicaciones periódicas, estadísticas y servicios de información. Es una aplicación Web, multilingüe, que soporta búsquedas en servidores Z39.50. Usa para el almacenamiento de datos la familia extendida del software ISIS.

El módulo Estadística ofrece varias funcionalidades en su pantalla principal que permiten la realización de las estadísticas. La primera de estas funciones es la creación de una nueva tabla, para ello debe definirse una tabla identificando qué valores se utilizan en la dirección horizontal (filas) y cuáles en la dirección vertical (columnas), (Ver Anexo 4). La segunda funcionalidad permite la utilización de una tabla existente, estas tablas se listan en la lista de opciones por sus criterios filas / columna. La lista de tablas disponibles se toma de un archivo de texto "tabs.cfg"; este archivo contiene, para cada tabla pre-definida, tres valores separados por el carácter '|' (nombre, campo para las filas y campo para las columnas), (Ver Anexo 5). La tercera de las funcionalidades permite la generación de la salida, esta implica dos fases: la creación de la tabla con los valores y la creación de cuadros gráficos de salida o la exportación de la tabla a otros formatos externos. Antes de la creación de la salida se debe definir un rango MFN o una expresión de búsqueda para aplicar las estadísticas sólo en ciertos sub-conjuntos de la base de datos, (Ver Anexo 6).

Principales características en las que coinciden los anteriores SIGB:

- Basados en Web.
- Plataforma multilingüe.
- Software libre.
- Los tres sistemas contienen un módulo para la generación de estadística.
- Realizan consultas para obtener los resultados estadísticos.

El estudio realizado a los módulos encargados de la generación de estadísticas de los registros bibliográficos, en los sistemas Koha, PMB y ABCD v1.2, permitió conocer el funcionamiento de cada uno de ellos. Fueron descartados primeramente los sistemas Koha y PBM debido que operan sobre bases de datos relacionales con MySQL. Posteriormente a pesar de que el sistema ABCD v1.2 presenta una mayor semejanza con la versión actual (ambas operan sobre bases de datos no relacionales) es descartada la integración pues utiliza como gestor de bases de datos C-ISIS, mientras que la actual versión utiliza como gestor de bases de datos a J-ISIS.

Luego del análisis realizado se concluye que el componente estadístico del sistema ABCD v3.0 debería mantener la entrada de los datos, la realización de las búsquedas por expresión de búsqueda o por rango MFN. Entre las limitantes encontradas se detectó que el módulo estadística del sistema ABCD v1.2 no permite la realización de las búsquedas por expresión de búsqueda y por rango MFN de forma concurrente, funcionalidad que se decide agregar al componente estadístico del sistema ABCD v3.0.

1.5 ISIS

ISIS (Integrated Set of Information Systems / Conjunto Integrado de Sistemas de Información), es un conjunto de programas, diseñado principalmente para operar bases de datos bibliográficos de volumen pequeño a mediano (10). Existen numerosas versiones de ISIS; entre ellas se encuentra (CDS/ ISIS) desarrollada por la UNESCO. En la Tabla 2 se muestra la familia extendida del software ISIS con el objetivo de mostrar la evolución de las mismas.

ISIS tiene la capacidad para cumplir tres tipos de funciones generales (10):

- Creación y mantenimiento de bases de datos.
- Recuperación de registros bibliográficos.
- Edición e impresión de información.

Familia extendida del software ISIS

Tabla 2. Familia extendida del software ISIS (10)

No. Generación	Año	Nombre
1era	1975	CDS/ISIS
		Micro/ISIS
2da	1985	Interfaces enriquecidas ISIS/Pascal
		Herramientas CISIS
3era	1995	Bases de datos múltiples, gráficas
		WinISIS ISISDLL
4ta	2005	Versiones adaptadas a www (wwwisis, isis3w, openisis, etc.)
5ta	2008	J-ISIS
		ISIS/NBP

A continuación en la Tabla 3 se realiza un resumen de las características pertenecientes a las tecnologías básicas (Clásico ISIS, J-ISIS, ISIS/NBP).

Tabla 3. Resumen de las características de las tecnologías básicas (10)

Tecnologías básicas. Resumen.			
Características principales	Clásico ISIS	J-ISIS	ISIS/NBP
Lenguaje de programación	Pascal C C++	Java	Python
Tecnología de base de datos	MST + RF	Berkeley DB	CouchDB
Estándar para el intercambio de información	ISO 2709	ISO 2909	ISO 2709
Técnica de indexación	Indexación de archivos invertida. Solo booleana	Indexación Lucene Clasificación	Indexación Lucene Clasificación

1.6 Sistema ABCD v3.0

Constituye una implementación superior del sistema ABCD v1.2. Usa tecnología Java. Actualmente se encuentra en desarrollo. Entre sus módulos se encuentran: Adquisición, Catalogación, OPAC y Estadísticas. En el sistema ABCD v3.0, las estadísticas generadas por el componente estadístico, referente a los registros bibliográficos almacenados en las bases de datos no relacionales de J-ISIS, son gestionadas y visualizadas mediante la funcionalidad Administrar Tabla.

Entre los casos de usos presentes en el módulo estadística del sistema ABCD v3.0 se encuentran:

Administrar tabla o Reporte estadístico: el caso de uso permite registrar, editar, visualizar, eliminar una tabla de reporte estadístico y generar su salida correspondiente. Además permite listar y filtrar las tablas o reportes estadísticos registrados en el sistema (11).

Administrar variables estadísticas: el caso de uso permite configurar las variables estadísticas utilizadas para la creación de las tablas de la BD J-ISIS (11).

Administrar reporte estadístico: el caso de uso permite administrar los reportes estadísticos que reflejen el comportamiento de la biblioteca (11).

1.7 Gestor de bases de datos no relacional J-ISIS-Suite (25 agosto 2014)

J-ISIS permite construir y administrar bases de datos estructuradas no numéricas, es decir bases de datos constituidas en su mayor parte por texto. Utiliza Berkeley DB v5.0 para la gestión de registros de longitud variable, Lucene para la búsqueda y recuperación de los recursos documentales y Web-JISIS para proporcionar acceso basado en web a los recursos de la biblioteca.

Tabla de extracción de campos (FST): define los criterios para la extracción de uno o más elementos (campo o sub-campo) de los registros de una base de datos de J-ISIS (12).

Número de archivo maestro (MFN): todos los registros almacenados en las bases de datos se identifican por un número único, asignado automáticamente por J-ISIS (12).

Con el objetivo de indexar adecuadamente cada uno de los campos, se utilizan técnicas de indexación. Las técnicas de indexación permiten especificar un procesamiento particular que se realiza en los datos, producidos por el formato, con el fin de identificar los elementos específicos a ser indexados. A continuación se mencionan brevemente las diez técnicas de indexación existentes en J-ISIS:

- **Técnica de Indexación 0:** construye un elemento de cada línea extraída por el formato; o sea, el formato de extracción de datos saca una línea para cada elemento que sea indexado. Esta técnica se utiliza para indexar campos enteros o sub-campos.
- **Técnica de Indexación 1:** construye un elemento de cada sub-campo o línea extraída por el formato. Para garantizar el correcto funcionamiento de esta técnica, su formato debe especificar el modo de prueba, porque es la única forma de preservar los códigos delimitadores de sub-campos. Constituye un acceso directo a la utilización de la técnica de indexación 0.
- **Técnica de Indexación 2:** construye un elemento de cada término o frase entre angulares triangulares (<...>). Cualquier texto fuera de los paréntesis no está indexado. Esta técnica requiere el modo de prueba, porque los otros modos eliminan los corchetes.
- **Técnica de indexación 3:** realiza mismo proceso de la técnica de indexación 2, excepto que los términos o frases están encerradas en barras (/.../).
- **Técnica de indexación 4:** construye un elemento de cada palabra en el texto extraído por el formato. Una palabra es cualquier secuencia contigua de caracteres alfabéticos.
- **Técnicas de indexación de la 5 a la 8:** las siguientes 4 técnicas de indexación permiten especificar un prefijo para los términos de búsqueda con las técnicas de indexación 1, 2, 3 y 4. Estas técnicas son las 5, 6, 7 y 8 respectivamente. El prefijo se especifica en el formato de extracción de datos como un literal incondicional de la siguiente forma 'dp...pd', [formato].
- **Técnicas de indexación 9:** es una técnica de indización específica J-ISIS. Permite indexar oraciones en la salida producida por el lenguaje de impresión de formateo.

- **Técnicas de indexación 10:** genera fichas N-gramas de las frases reconocidas en el texto.

Luego de analizadas las diez técnicas de indexación antes mencionadas, se decide escoger la técnica de indexación número 4 para la implementación del componente estadístico por ser la más abarcadora.

1.8 Berkeley DB

Berkeley DB es un motor de base de datos que provee a los desarrolladores una base de datos simple, rápida y segura. Puede ser utilizado en aplicaciones que requieran alto rendimiento de almacenamiento y recuperación simultánea de pares clave / valor. La familia de productos de base de datos Berkeley se compone de tres productos: base de datos *Berkeley*, *Berkeley DB Java Edition* y *Berkeley DB XML*.

1.9 Consultas Lucene

Para la realización de las búsquedas en los textos de las BD, se utilizaron las consultas Lucene. Lucene es una librería de recuperación de información de alto rendimiento y escalable que permite añadir capacidades de indexación y búsqueda a cualquier aplicación. Esta desarrollada en Java. Lucene puede indexar y realizar búsquedas sobre cualquier dato que pueda ser convertido a texto. Lucene indexa los documentos o BD que contiene la aplicación para que, a través de una consulta del usuario, busque en el índice y muestre los resultados con éxito (Ver Figura 1). Entre sus características se encuentran las siguientes: permite la indexación incremental, soporta la indexación de documentos con formato: TXT, PDF, DOC, RTF, XML, PPT y HTML, permite la concurrencia, o sea que permite realizar la búsqueda mientras se actualiza el índice, es multi-plataforma, puede realizar búsquedas por campos, rangos de fechas, entre otras.

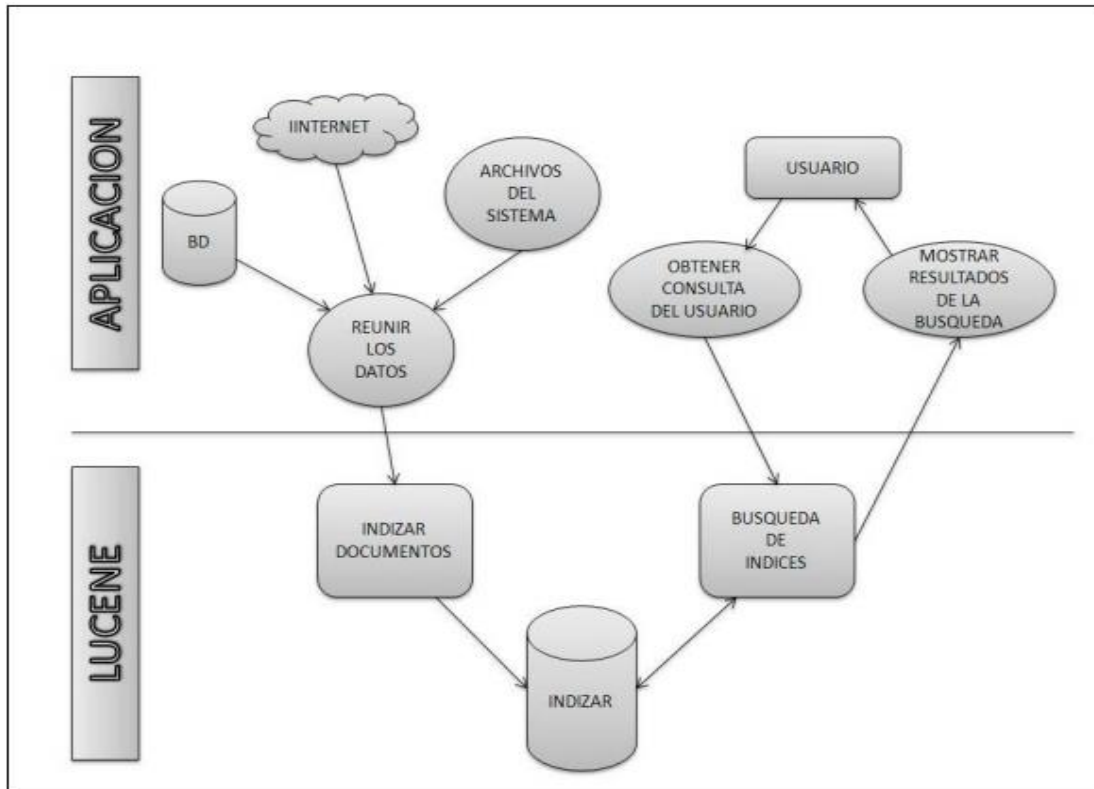


Figura 1. Esquema de cómo una aplicación usa Lucene (13)

1.9.1 Funcionalidades básicas de Lucene

Indexación: se encarga de procesar todos los datos originales en un formato que permite buscarlos rápidamente sin necesidad de escanearlos cada vez que se quiere realizar una búsqueda.

Búsqueda: capacidad de realizar consultas sobre los datos para obtener documentos que contengan información valiosa. La búsqueda tiene en cuenta dos factores principales: la destitución y la precisión. La destitución se encarga de indicar qué documentos son relevantes a la búsqueda, mientras que la precisión se encarga del filtrado de los datos.

Estructura de una consulta Lucene empleada en J-ISIS:

Una consulta Lucene está desglosada en términos y campos. Los campos hacen referencia a cada uno de los campos indexados en la BD a los cuales se acceden mediante la sintaxis “_Número del campo: Término”, el cual puede ser simple “**Mario**” o compuesto “**Mario de Andrade**”. Por otra parte se pueden realizar consultas compuestas mediante la utilización de operadores lógicos, estos pueden ser AND, OR y NOT.

Ejemplo consulta simple (Ver Figura 2): _100: Mario

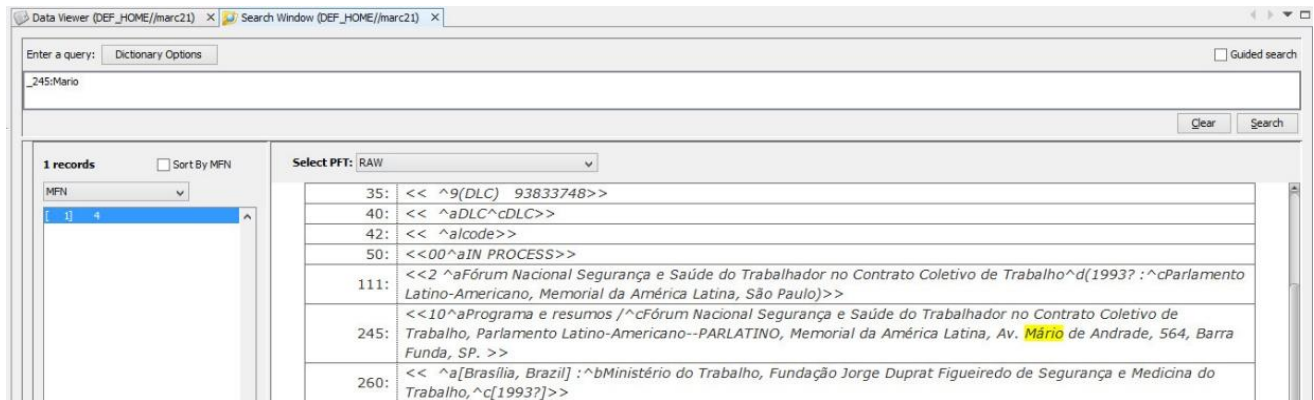


Figura 2. Resultado mostrado por J-ISIS al realizar una consulta simple.

Ejemplo consulta compuesta (Ver Figura 3): _100: García AND _245: sanitarios

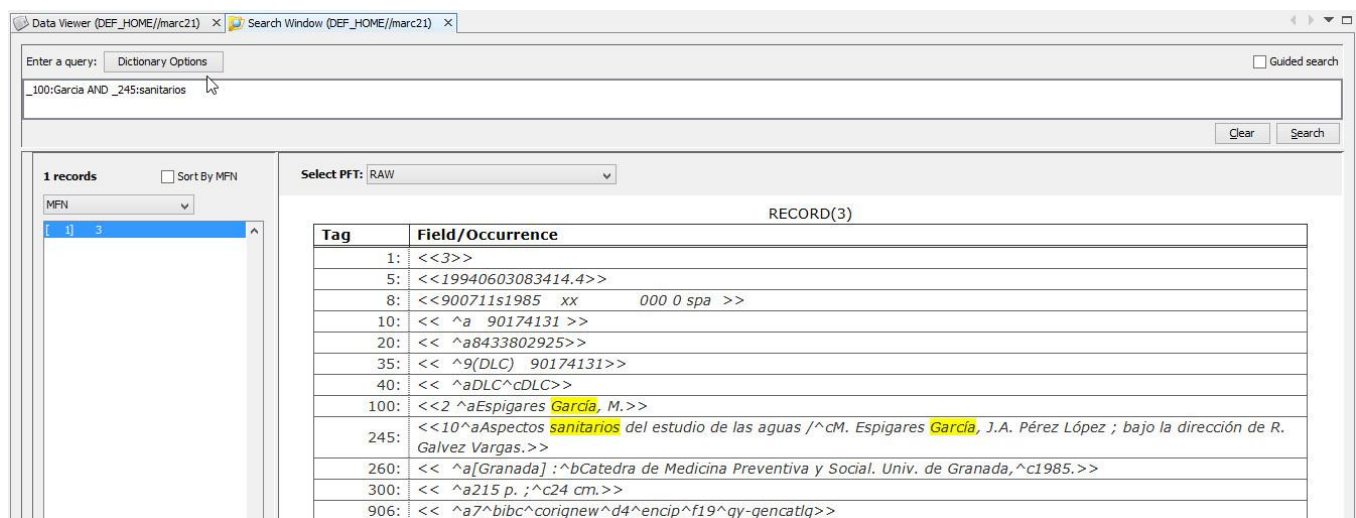


Figura 3. Resultado mostrado por J-ISIS al realizar una consulta compuesta

1.10 Open Services Gateway Initiative (OSGi)

Es un marco de trabajo modular para Java que especifica la forma de crear módulos y la manera en que estos interactúan en tiempo de ejecución. Este marco de trabajo proporciona a los desarrolladores un entorno orientado a servicios y basado en componentes (14). OSGi se divide en un conjunto de capas, cada una de las cuales proporciona una base sólida para la modularidad. Entre los contenedores de OSGi se encuentra Equinox v3.8, siendo una implementación de OSGi para la plataforma Eclipse, a su vez es usado por el proyecto ABCD v3.0. A continuación se muestra en la Figura 4 la representación de las capas de OSGi y en la Tabla 4 las descripciones correspondientes a cada una de ellas.

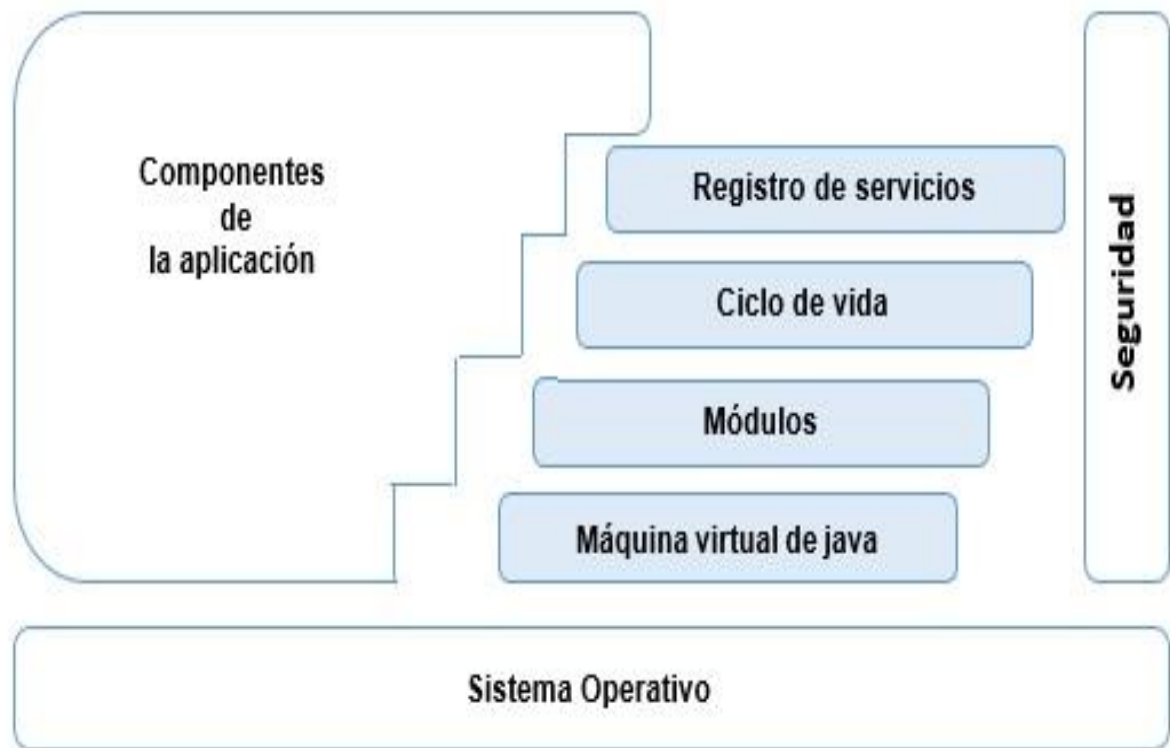


Figura 4. Arquitectura en capas de OSGi (14)

Tabla 4. Descripción de las capas de la arquitectura OSGi (13)

Capas	Descripción
Módulo	Es responsable de manipular los componentes. Su objetivo principal es proporcionar modularidad basada en la plataforma Java. Hace cumplir las normas de visibilidad entre módulos.
Ciclo de vida	Se basa en la capa Módulo para la carga de clases y proporciona un enfoque dinámico de la gestión del componente, lo que permite actualizar partes de una aplicación sin detener el componente. Cuando un <i>bundle</i> está en estado activo, es capaz de interactuar con el registro de servicios, impulsado por la capa superior de OSGi.
Registro de servicio	Permite bundles para ser utilizados y para interactuar de manera dinámica, tomando en cuenta la naturaleza del sistema. A través de los

	registros de servicio, OSGi ofrece la posibilidad de registrar uno o varios puntos de acceso a los componentes.
--	---

1.10.1 Bundle

Un *bundle* es una unidad de modularización, definida por el marco de trabajo OSGi. Los *bundles* pueden compartir los paquetes Java entre un *bundle* exportador y un *bundle* importador de paquete. (14). Puede además registrar servicios y acceder a los servicios existentes en otros *bundles* activos en el contexto de OSGi.

Un *bundle* es un archivo .jar que (14):

- Contiene los recursos necesarios para proporcionar alguna funcionalidad. Estos recursos pueden ser archivos de clase para el lenguaje de programación Java, así como otros datos (archivos HTML, archivos de ayuda, iconos, etc.).
- Contiene un archivo de manifiesto, que describe el contenido del archivo JAR y proporciona información acerca del *bundle*.
- Un *bundle* se inicia a través de su *bundle* activador.

Un *bundle* puede encontrarse en alguno de los siguientes estados (Ver Figura 5):

- **Instalado:** el *bundle* se ha instalado correctamente.
- **Resuelto:** este estado indica que el *bundle* está preparado para ser iniciado o que el *bundle* se ha detenido.
- **Iniciado:** un *bundle* se activa llamando a su objeto *Bundle Activador*, si es que existe.
- **Activo:** el *bundle* ha sido activado con éxito y se está ejecutando.
- **Inactivo:** se detiene el *bundle*.
- **Desinstalado:** el *bundle* se ha desinstalado. No se puede mover a otro estado.

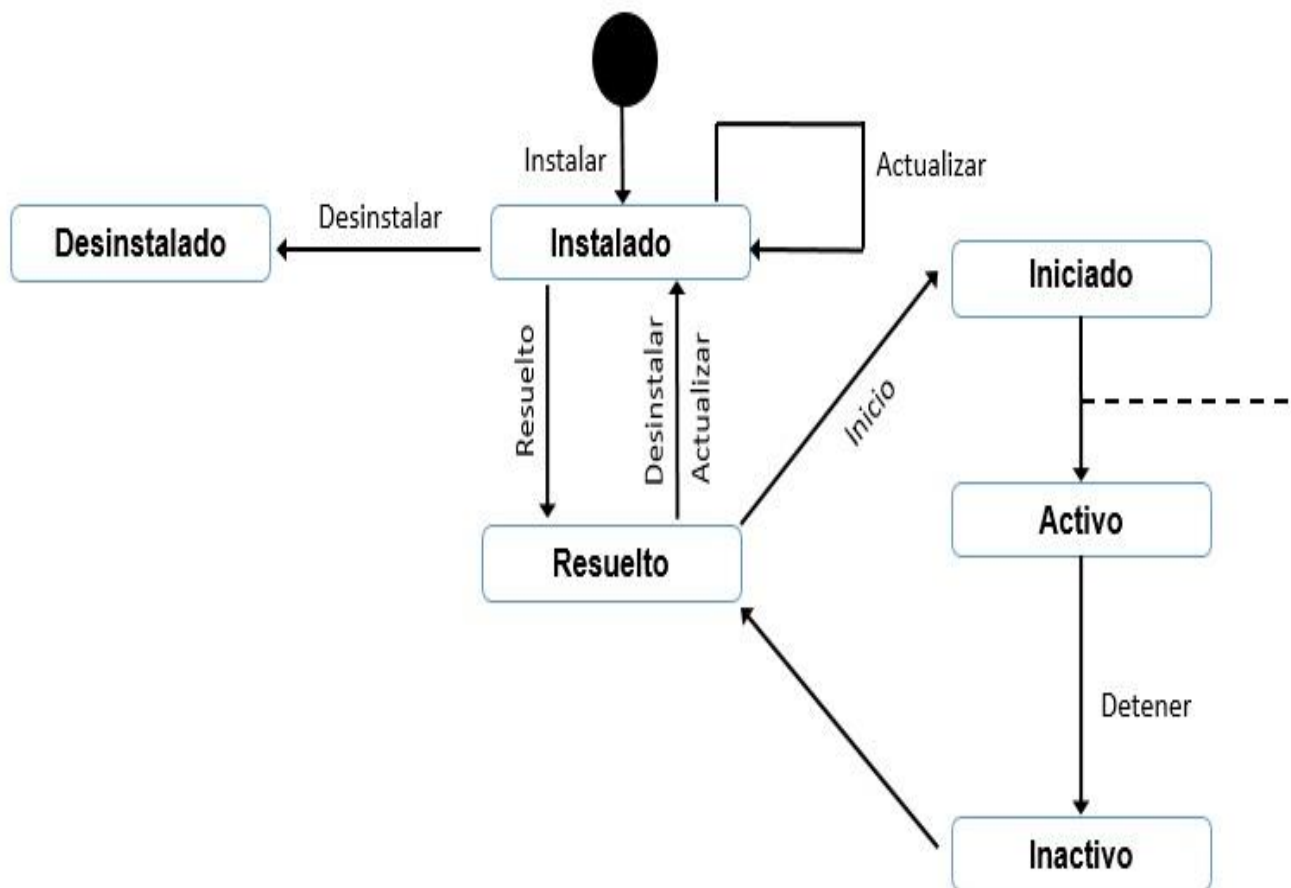


Figura 5. Ciclo de vida de un bundle (13)

1.11 Metodología, herramientas, lenguajes y tecnologías

Para el desarrollo de la presente investigación se utilizaron un conjunto de herramientas, tecnologías y una metodología de desarrollo de software, previamente establecidas como políticas del proyecto ABCD, por lo que su selección queda fuera del alcance del presente trabajo.

1.11.1 Metodología de desarrollo de software Rational Unified Process (RUP)

RUP es un proceso de ingeniería de software que proporciona un enfoque disciplinado para la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Tiene como propósito asegurar la producción de software de alta calidad, que se ajuste a las necesidades de sus usuarios finales con costos y calendarios predecibles (15). RUP intenta integrar los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos de software. En RUP, la necesidad de contar con una secuencia de actividades realizadas por diferentes roles para definir el proceso, obliga al uso de los flujos de trabajo;

Fases de RUP:

- **Inicio:** esta fase tiene como objetivo establecer el ámbito del proyecto y sus límites, encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad, mostrar al menos una arquitectura candidata para los escenarios principales, así como estimar el coste en recursos y tiempo de todo el proyecto (15).
- **Elaboración:** tiene como propósito analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final (15).
- **Construcción:** la finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones (15).
- **Transición:** la finalidad de esta fase de transición es poner el producto en manos de los usuarios finales. Para ello se requerirá desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y usabilidad del producto (15).

A continuación, en la Figura 6, se muestra el flujo de trabajo de RUP. Se encuentran seleccionadas las fases y disciplinas en las que se basa la presente investigación.

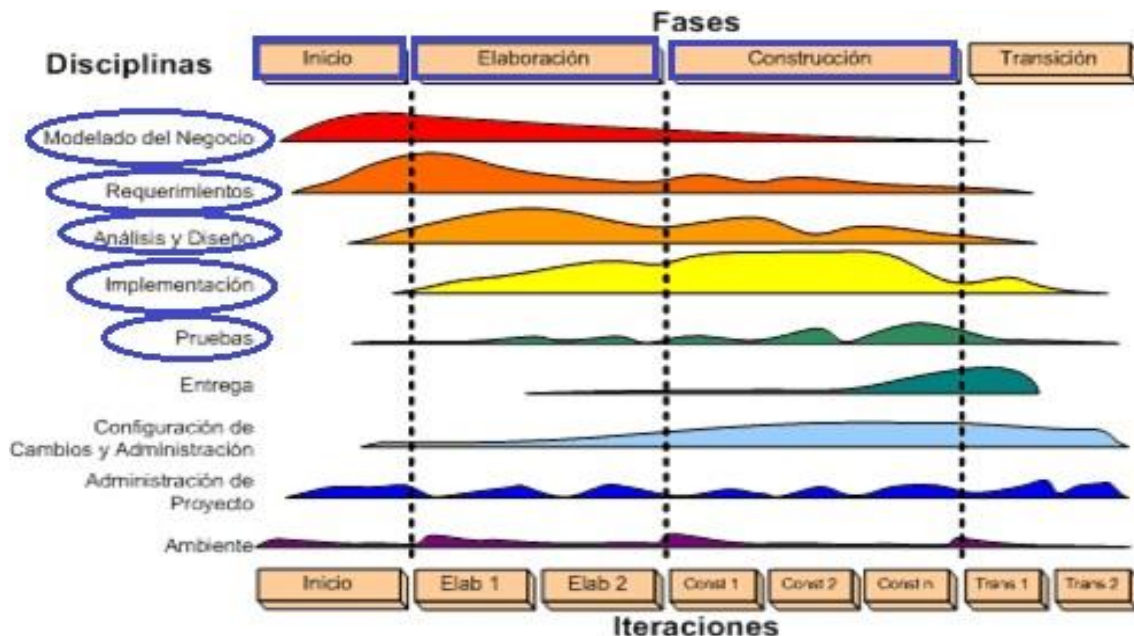


Figura 6. Fases de RUP (16)

1.11.2 Lenguaje Unificado de Modelado (UML) v2.1

UML es un lenguaje de modelado diseñado para visualizar, especificar, construir y documentar los artefactos de los sistemas de software, así como para el modelado del negocio (17). UML estandariza nueve tipos de diagramas para representar gráficamente un sistema desde distintos puntos de vista.

Tipos de Diagrama UML:

- Diagramas de clases.
- Diagramas de objetos.
- Diagramas de interacción.
 - Diagrama de secuencia.
 - Diagrama de colaboración.
- Diagramas para representar aspectos dinámicos del sistema.
 - Diagrama de casos de uso.
 - Diagrama de estados.
 - Diagrama de actividades.
- Diagramas para representar aspectos físicos del sistema.
 - Diagrama de componentes.
 - Diagrama de despliegue.

1.11.3 Entorno de desarrollo integrado (Eclipse-Juno 4.2.2)

Eclipse-Juno v4.2.2 es un entorno de desarrollo integrado (IDE). Eclipse está diseñado para afrontar las siguientes necesidades: soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML y GIF), facilitar una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta el proveedor, proporcionar entornos de desarrollo (GUI) o no gráficos, ejecutarse en gran variedad de sistemas operativos, incluidos Windows y Linux (18). Tiene además pruebas unitarias con JUnit y control de versiones con subclipse.

1.11.4 Herramienta CASE Visual Paradigm v8.0

Es una herramienta CASE ¹para el modelamiento UML. Concebida para soportar el ciclo de vida completo de desarrollo del software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue (19). Permite modelar diversos tipos de diagramas UML (Diagramas de clases, diagrama

¹ CASE: Por sus siglas en inglés Computer Aided Software Engineering, en español Ingeniería de Software Asistida por Computadoras.

de paquetes, diagrama de componentes, diagrama de despliegue, entre otros). Permite revertir y generar código fuente desde los diagramas. Soporta Java, C++, DotNet Exe / dll, XML, XML Schema, y Corba IDL. Genera reportes y documentación en HTML / PDF. Permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software.

1.11.5 Lenguaje de Programación Java v7.0

Java es un lenguaje orientado a objetos. Es interpretado por la JVM (*Java Virtual Machine / Máquina Virtual de Java*) esto lo hace más lento pero sin embargo trae consigo sus ventajas, en particular el hecho de no tener que recompilar un programa Java de un sistema a otro porque basta para cada uno de los sistemas tener su propia máquina virtual. Además es robusto ya que la declaración de las variables debe ser obligatoriamente explícita, se verifica además el código (sintaxis, tipos) en el momento de la compilación y de la ejecución lo que permite reducir los errores y los problemas de incompatibilidad de versiones.

Java es independiente de las arquitecturas, pues no produce un código específico para un tipo de arquitectura. Otra de sus características es que es portable ya que se trata de un lenguaje interpretado, además los tipos de datos primitivos tienen el mismo tamaño sea cual sea la plataforma en la cual se ejecuta el código. También es multihilo pues permite desarrollar aplicaciones que ponen en marcha la ejecución simultánea de varios hilos, esto permite ejecutar simultáneamente varias tareas con el fin de aumentar la velocidad de las aplicaciones ya sea compartiendo el tiempo del CPU o compartiendo las tareas entre varios procesadores (20). En la versión 7 de Java se agregan nuevas características que permiten incrementar la productividad del desarrollador y simplificar las tareas comunes de programación disminuyendo la cantidad de código necesario.

A continuación se especifican algunas de las características integradas a la versión 7 (21):

- Inferencia de tipos u operador diamante para simplificar el uso de *generics*: no es necesario tener que indicar los objetos en la inicialización.
- Número en literales binarios: con Java 7 es posible declarar literales en formato binario.
- Números con guiones: java proporciona la opción de poder escribir separando con guiones para facilitar la lectura al programador.
- Multi-catch: útil para evitar código duplicado de manejo de errores.

1.11.6 Servidor de Aplicaciones Virgo v3.6.3

Virgo es un servidor de aplicaciones basado en OSGi originalmente desarrollado por *Spring DM Server* y actualmente mantenido por la Fundación Eclipse. Virgo soporta el despliegue de los *bundles* de OSGi y no modifica el funcionamiento de las aplicaciones web. Permite además compartir bibliotecas y

servicios. Virgo proporciona una forma de agrupar un conjunto de *bundles* OSGi y otros artefactos que comprenden una sola aplicación (22).

1.11.7 Herramientas de automatización de pruebas de software

JUnit v4.10

JUnit es un marco de trabajo que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado, si la clase cumple con la especificación entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba. En caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

Ventajas y modificaciones de JUnit 4, respecto a versiones anteriores (23):

- Las clases que contienen los métodos que representan las pruebas ya no tienen que ser subclases de **junit.framework.TestCase**
- Los métodos que representan las pruebas a realizar ya no tienen que tener el prefijo “test” en su nombre sino que se indican con la anotación **@Test**.
- Sustitución del método ‘*setUp*’ por la anotación **@Before**.
- Sustitución del método ‘*tearDown*’ por la anotación **@After**.
- Se pueden realizar pruebas en donde el tiempo de ejecución es crítico. De manera que una prueba falle si tarda más de X milisegundos en ser ejecutada.
- Se pueden realizar pruebas en donde se debe controlar que una excepción es lanzada.
- Se pueden desactivar pruebas, a través de la anotación **@Ignore**.

Apache JMeter v3.0

Es una herramienta Java desarrollada dentro del proyecto Jakarta, que permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web. Puede ser usada para realizar pruebas de estrés, por ejemplo en un servidor, y poner en prueba su rendimiento. Permite simular una gran cantidad de usuarios mediante la utilización de hilos que acceden a los servicios de forma concurrente o cada cierto intervalo de tiempo (24).

1.12 Conclusiones parciales

En este capítulo se realizó un análisis de los principales conceptos relacionados con el objeto de estudio, que proporcionó la base teórica para el desarrollo del componente. El estudio realizado a los SIGB sobre el proceso de generación de estadísticas referente a los registros bibliográficos permitió conocer el funcionamiento de los mismos, contribuyendo a la posterior implementación del componente

estadístico. Se hizo un estudio sobre las funcionalidades básicas y la estructura de las consultas Lucene, que permitió obtener el conocimiento necesario para la realización de consultas en las bases de datos de J-ISIS.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL COMPONENTE

2.1 Introducción

En este capítulo se define el modelo de dominio. Se identifican los requisitos funcionales y no funcionales del sistema. Se realiza el diagrama de casos de uso del sistema, así como la descripción de los mismos. Se definen además el estilo arquitectónico y los patrones de diseño utilizados. Se plantea la propuesta de solución del componente y se elabora el conjunto de artefactos que se genera durante la fase de análisis y diseño.

2.2 Modelo del dominio

Un modelo de dominio captura los tipos de objetos más importantes en el contexto del sistema. Los objetos del dominio representan los eventos que suceden en el entorno en que trabaja el sistema. Este modelo permite mostrar de manera visual los principales conceptos que se emplean, facilitando a los usuarios, desarrolladores e interesados la utilización de un vocabulario común para poder entender y comprender el contexto en que se desarrolla el sistema (25). El modelo de dominio se representa mediante diagramas de clases UML, estos diagramas muestran a los clientes, usuarios, revisores y otros desarrolladores las clases del dominio y como se relacionan mediante asociaciones. A continuación, en la Tabla 5 se muestra la descripción de los principales conceptos relacionados en el modelo de dominio.

Tabla 5. Conceptos y descripción del modelo del dominio (Fuente: Elaboración propia)

Conceptos	Descripción
Módulo Estadística	Es el módulo, del sistema ABCD v3.0, que va a hacer uso del componente para realizar estadísticas dinámicas cruzadas.
Componente	Es el <i>bundle</i> que permite realizar la generación de estadísticas dinámicas con cruzamiento de variables.
FST ²	Define los campos de la base de datos para realizar búsquedas a través del archivo invertido, utilizando técnicas de indexación.
Berkeley DB	Es el motor de BD usado por J-ISIS, para la persistencia de los datos.

En la siguiente Figura 7, se muestra una representación de las clases del modelo de dominio del componente y las relaciones entre ellas.

² FST: Field Selection Table/ Tabla de Extracción de Campos

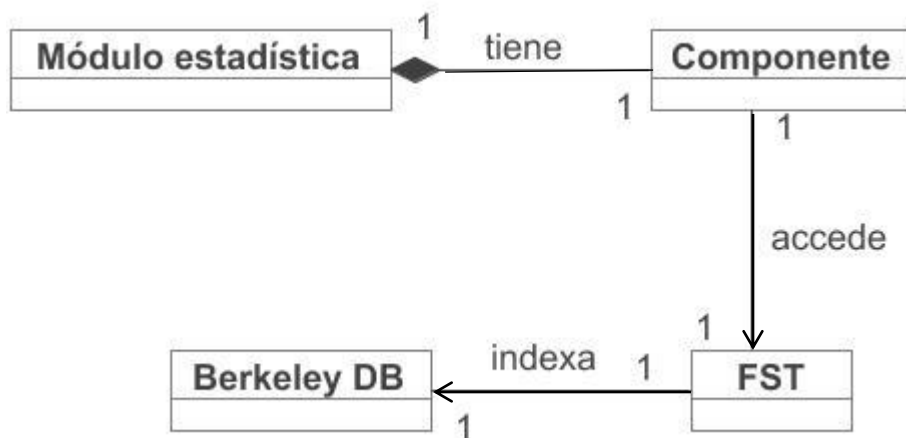


Figura 7. Diagrama del modelo del dominio (Fuente: Elaboración propia)

2.3 Requisitos funcionales

Los requisitos funcionales (RF) de un *software* definen lo que el sistema será capaz de realizar, funcionalidades requeridas y restricciones que son aprobadas en mutuo acuerdo con el usuario final, describen además las transformaciones que el sistema realiza sobre las entradas para producir salidas. Estos requisitos, a medida que avanza el proyecto de *software* se convierten en los algoritmos, la lógica y gran parte del código del sistema. A continuación, se muestra la Tabla 6 que contiene la descripción de los requisitos funcionales del componente.

Tabla 6. Requisitos funcionales y sus descripciones (Fuente: Elaboración propia)

Requisitos	Descripción
RF_1: Realizar búsquedas sobre las bases de datos J-ISIS por rango MFN.	Permite al usuario realizar búsquedas mediante un rango MFN, previamente definido.
RF_2: Realizar búsquedas sobre las bases de datos J-ISIS por expresiones de búsqueda.	Permite realizar búsquedas en las bases de datos de J-ISIS haciendo uso de expresiones de búsqueda Lucene.
RF_3: Realizar suma de las coincidencias de las variables estadísticas.	Permite realizar la suma, horizontal y vertical, de las coincidencias de las variables estadísticas.
RF_4: Obtener estadísticas cruzadas.	Permite obtener el resultado del cruzamiento de dos variables estadísticas en forma de matriz.

2.4 Requisitos no funcionales

Los requisitos no funcionales (RNF) son cualidades que el producto debe cumplir, características que hacen al producto usable, rápido, confiable y son de vital importancia para una puesta en marcha exitosa del *software*, además de lograr que este responda a las expectativas del usuario (26).

RNF_1: Hardware: son los elementos que se deben disponer, para que el componente implementado cumpla con sus funcionalidades.

PC Servidor:

- Memoria RAM: 2 GB
- Espacio libre en el disco duro: 80GB
- Microprocesador: 2.16GHz

RNF_2: Software: son las tecnologías que debe tener instalada el servidor donde se va a desplegar el componente.

PC Servidor:

- Servidor de aplicaciones Virgo en la versión 3.6.3.
- Servidor de base de datos no relacional J-ISIS-Suite (25 agosto 2014).
- Java Runtime Environment de Oracle 7.70.

RNF_3: Usabilidad: describe los niveles apropiados de usabilidad, como los factores humanos, ayudas y documentación del componente.

- Para una mejor visibilidad de los datos se debe poder asegurar que el número de filas sea mayor que el número de columnas.

RNF_4 – Rendimiento: es el modo mediante el cual los usuarios examinan con detenimiento hasta qué punto el proyecto se ajusta a sus expectativas a los tiempos de respuesta, disponibilidad y es capaz de prestar servicio adecuadamente de acuerdo al tipo y tamaño para el que ha sido concebido.

- Debe proporcionar apropiados tiempos de respuesta y procesamiento, así como tasas de producción de resultados, al realizar su función bajo condiciones establecidas. Al efectuar acciones de cargar un registro el sistema no debe exceder los 20 segundos y al efectuar acciones de cálculos su tiempo debe ser menor o igual a 1 minuto.

2.5 Diagrama de casos de uso

Un diagrama de casos de uso explica gráficamente el contexto de un sistema, los actores, la relación entre estos y los casos de uso. Tiene como objetivo ofrecer un diagrama contextual que permita conocer los actores externos de un sistema y las formas básicas en que lo utilizan (17). Sirve como herramienta de comunicación que resume el comportamiento de un sistema y sus actores. A continuación en la Tabla 7 se muestra la descripción del actor y en la Figura 8 el diagrama de casos de uso del componente.

Tabla 7. Descripción del actor (Fuente: Elaboración propia)

Actor	Descripción
Gestor de reportes estadísticos	<ul style="list-style-type: none"> ➤ Usuario que interactúa con el sistema. ➤ Genera un reporte con las estadísticas definidas. ➤ Utiliza todas las funcionalidades que brinda el módulo estadística.

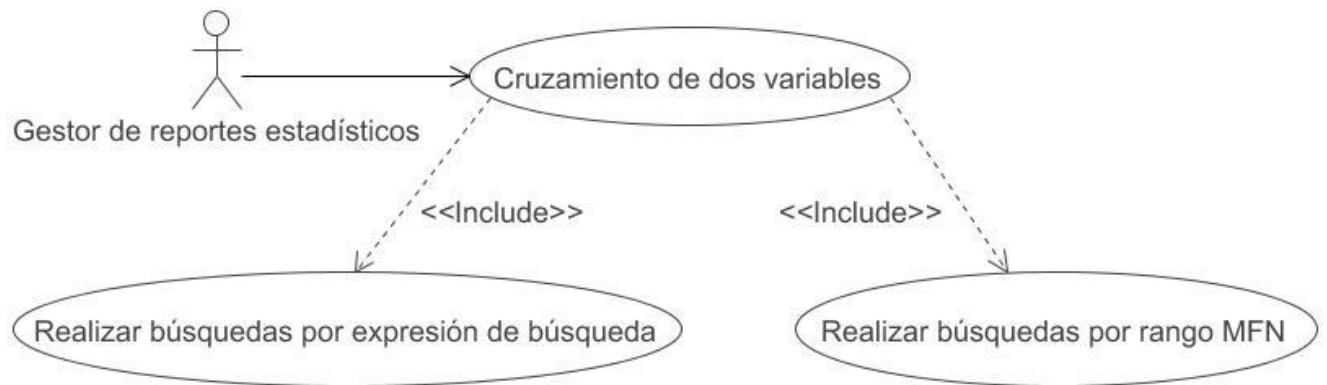


Figura 8. Diagrama de casos de uso del sistema (Fuente: Elaboración propia)

2.5.2 Descripción de los casos de uso (CU)

Las descripciones de los casos de uso, detallan las acciones que tienen lugar durante la interacción actor-sistema, es decir, describen el flujo de actividades que realiza el actor al hacer uso del sistema y las correspondientes respuestas del mismo. A continuación en las Tablas 8, 9 y 10 se describen los casos de uso: Realizar búsqueda por expresión de búsqueda, realizar búsqueda por rango MFN y cruzamiento de dos variables.

Tabla 8. Descripción del CU_ Realizar búsqueda por expresión de búsqueda (Fuente: Elaboración propia)

Caso de uso	Realizar búsquedas por expresión de búsqueda.
Objetivo	Realizar búsquedas sobre bases de datos J-ISIS mediante una expresión de búsqueda especificada.
Actores	Gestor de reportes estadísticos

Resumen	El caso de uso comienza cuando el Gestor de reportes estadísticos, inserta una expresión de búsqueda, posteriormente selecciona la opción generar salida y obtiene el resultado. Finaliza el caso de uso.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	Los registros bibliográficos tienen que estar indexados en Lucene.	
Postcondiciones	Se obtiene una matriz con los valores correspondientes a la expresión de búsqueda definida.	
Relaciones	CU incluidos	Ninguno
	CU extendidos	Ninguno
Requisitos no funcionales		No aplica.
Asuntos pendientes		No aplica.

Tabla 9. Descripción del CU_ Realizar búsqueda por rango MFN (Fuente: Elaboración propia)

Caso de Uso	Realizar búsqueda por rango MFN.
Objetivo	Realizar búsquedas sobre bases de datos J-ISIS mediante un rango MFN especificado.
Actores	Gestor de reportes estadísticos
Resumen	El caso de uso comienza cuando el Gestor de reportes estadísticos, define un rango MFN, posteriormente selecciona la opción realizar búsqueda y obtiene el resultado. Finaliza el caso de uso.
Complejidad	Alta
Prioridad	Alta
Precondiciones	El par de variables seleccionadas debe estar indexada en la base de datos.

Postcondiciones	Se obtiene una matriz con los valores correspondientes al rango MFN especificado.	
Relaciones	CU incluidos	Ninguno
	CU extendidos	Ninguno
Requisitos no funcionales	No aplica.	
Asuntos pendientes	No aplica.	

Tabla 10. Descripción del CU_Cruzamiento de dos variables (Fuente: Elaboración propia)

Caso de Uso	Cruzamiento de dos variables.	
Objetivo	Realizar el cruzamiento del par de variables seleccionado y obtener el resultado estadístico correspondiente.	
Actor	Gestor de reportes estadísticos	
Resumen	El caso de uso comienza cuando el Gestor de reportes estadísticos selecciona el par de variables a ser cruzadas, luego define el tipo de búsqueda a realizar (por rango MFN o por expresión de búsqueda), posteriormente se muestran los resultados estadísticos del par de variables seleccionadas. Finaliza el caso de uso.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	El par de variables seleccionadas deben estar indexadas en la base de datos.	
Postcondiciones	Se obtiene una matriz con los valores correspondientes a la expresión de búsqueda o el rango MFN especificado.	
Flujo de eventos		
Flujo básico “obtener estadísticas cruzadas”		
	Actor	Sistema

1	Selecciona el par de variables a ser cruzadas.	
2		El sistema verifica si el par de variables seleccionadas se encuentran indexadas.
3		Se obtiene la matriz con los valores estadísticos correspondientes a la expresión de búsqueda o rango MFN especificado.
4		Termina el caso de uso.
Flujos alternos		
1ero "El par de variables seleccionadas no están indexadas"		
	Actor	Sistema
1		Si alguna o ambas variables no se encuentran indexadas se entra al FST Manager (donde son indexados los datos), ahí se encuentran dos arreglos, una de ellos contiene los datos que están indexados y el otro los datos no indexados.
2		Se buscan las variables especificadas en el arreglo de datos no indexados y se adicionan al arreglo de los datos indexados.
3		Se guardan los cambios realizados.
4		Se re-indexa la base de datos.
5		Regresa al paso 1 del flujo básico.
Relaciones		CU incluidos
		Realizar búsqueda por expresión de búsqueda. Realizar búsqueda por rango MFN.
		CU extendidos
		Ninguno
Requisitos no funcionales		No aplica.
Asuntos pendientes		No aplica.

2.6 Arquitectura del componente

La arquitectura de software es la estructura del sistema e incluye los componentes fundamentales del mismo. Proporciona un marco de referencia para guiar de manera más organizada la construcción del software entre los analistas, diseñadores, programadores y demás miembros del equipo de desarrollo (15). Para su definición es posible hacer uso de diferentes estilos y patrones arquitectónicos, basándose en los objetivos fundamentales del software a desarrollar.

Para el desarrollo del componente, se toma como base la arquitectura del sistema ABCD v3.0. Es una arquitectura que propone el desarrollo de componentes, basados en OSGi. Los componentes, conocidos como *bundles*, interactúan entre ellos mediante servicios dentro de la Máquina Virtual de Java una vez desplegados en el servidor de aplicaciones Virgo v3.6.3 que utiliza como implementación de OSGi Equinox 3.8.

Para facilitar la publicación y el consumo de servicios se realiza su configuración mediante archivos XML utilizando SpringDM. Cada servicio publicado en el contexto de OSGi debe proveer una interfaz pública que es utilizada por los *bundles* externos. La implementación de cada servicio es privada y solo puede ser accedida mediante su interfaz. Los *bundles* implementados en la arquitectura según sus responsabilidades pueden estar ubicados en la capa de acceso a datos, la capa de negocio o capa de presentación. Los *bundles* de determinada capa solo acceden a los servicios de *bundles* de capas inferiores. La Figura 9 muestra la dependencia entre las capas de la arquitectura.

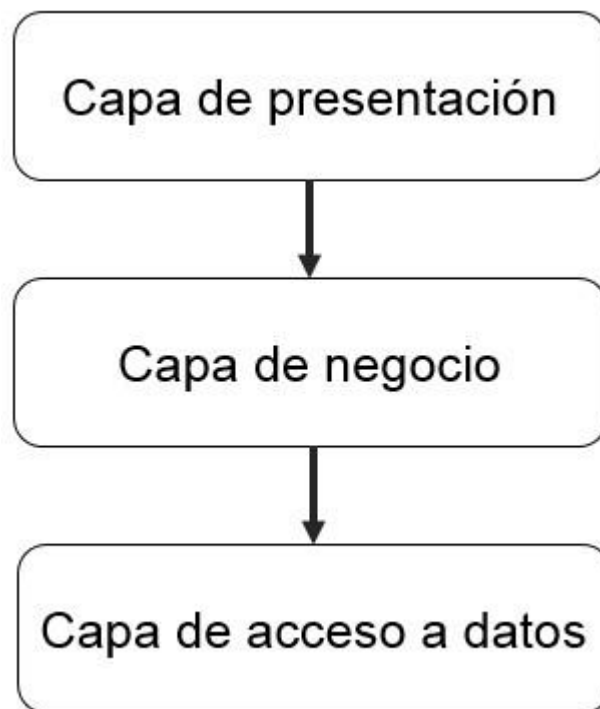


Figura 9. Dependencia entre las capas de la arquitectura

El componente es un *bundle* de negocio del módulo Estadística usado para realizar la generación de estadísticas dinámicas. Este tiene dependencia de la biblioteca *jisis-core* v2.0 para la conexión al servidor de base de datos de J-ISIS. El componente es usado por el *bundle* *cu.uci.abcd.statistic*.

2.7 Patrones de diseño utilizados en el desarrollo del componente estadístico

Un patrón de diseño se define como una solución a un problema que se usa repetidamente en contextos similares. Se encarga de identificar, abstraer y nombrar los aspectos elementales de una estructura de diseño, donde los componentes son las clases y objetos, y sus mecanismos de interacción son mensajes. Ayudan a elegir diseños alternativos que hacen un sistema reutilizable y evitan alternativas que comprometan la reutilización (17).

2.7.1 Patrones GRASP

Los patrones generales de software para asignación de responsabilidades (GRASP), describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Ayudan a entender el diseño del objeto esencial y aplicar el razonamiento para el diseño de una forma sistemática, racional y explicable. Se pueden destacar 5 patrones principales: bajo acoplamiento, alta cohesión, experto, creador y controlador.

Experto: este patrón indica que se debe asignar responsabilidades al experto de la información, es decir, a la clase que tiene la información necesaria para realizar las responsabilidades. Se ve reflejado en la clase ***TabularStatistic*** que es la que contiene la información necesaria para realizar la tabla de estadística.

Creador: este patrón nos ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases. La nueva instancia debe ser creada por la clase que tiene la información necesaria para realizar la creación del objeto. Se ve reflejado en la clase ***CrossStatisticImpl*** cuando se crea un objeto de tipo ***TabularStatistic***.

Interface: este patrón define los parámetros de un tipo interfaz, todas las clases o instancias que quieran utilizar ese comportamiento deben implementar dicha interfaz. Se ve reflejado en la clase ***CrossStatisticImpl*** donde se implementa su interfaz ***CrossStatistic***.

Alta Cohesión: asigna una responsabilidad de manera que la cohesión permanezca alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. La alta cohesión caracteriza a las clases con responsabilidades altamente relacionadas y que no lleva a cabo gran cantidad de trabajo. Se ve reflejado en la clase ***JISISStatisticProviderImpl*** que utiliza la clase ***Option*** para realizar las consultas Lucene.

Bajo Acoplamiento: asigna una responsabilidad para mantener bajo acoplamiento. El acoplamiento indica que tan fuertemente está conectada una clase con otra. Por tanto, este patrón asigna responsabilidades de manera que el acoplamiento permanezca bajo, es decir que una clase no dependa de otras clases. Se ve reflejado en la clase **CrossStatisticImpl** que depende de la interfaz **JISISStatisticProvider** la cual no se encuentra acoplada a la implementación de dicha interfaz.

Polimorfismo: cuando el comportamiento relacionado varía según el tipo (clase), asigna la responsabilidad para el comportamiento utilizando operaciones polimórficas a los tipos para los que varía el comportamiento. Se ve reflejado en las clases **OptionAND**, **OptionNOT** y **OptionOR** que heredan de la clase **Option**.

2.7.2 Patrones GOF

Por sus siglas en inglés Gang of Four (Banda de los Cuatro), son un conjunto de patrones de diseño creados con el objetivo de codificar y hacer reutilizables una serie de principios referentes al diseño de aplicaciones de alta calidad. Se dividen en tres categorías principales (15):

Creacionales: Encierran conocimientos acerca de cuáles son las clases concretas que usa el sistema.

Estructurales: Tratan con la composición de las clases y objetos.

Comportamiento: Caracterizan el modo en que las clases u objetos interactúan y distribuyen responsabilidades.

Patrón empleado en el diseño de la solución:

Instancia única (Singleton): este patrón se encuentra dentro de la categoría Creacionales. La utilidad de este patrón es asegurar que una clase tiene una sola instancia y a su vez proporcionar un punto de acceso global a ella. Se ve reflejado cuando se registra un servicio en el contexto de OSGi con SpringDM al hacer una única instancia de la clase **CrossStatisticImpl**.

Iterador: este patrón se encuentra dentro de la categoría Comportamiento. Este patrón tiene como intención proporcionar una forma de acceder a los elementos de una colección de objetos de manera secuencial sin revelar su representación interna. Se ve reflejado en la clase **JISISStatisticProviderImpl** en el trabajo de colección.

2.8 Propuesta de solución

Se propone la creación de un *bundle* con el principal objetivo de obtener las estadísticas cruzadas de dos variables, para ello se hace necesario realizar consultas Lucene por alguna expresión de búsqueda sobre el API de J-ISIS, se podrán además realizar búsquedas por rango MFN, así como por ambas de forma concurrente. El componente es capaz de realizar los cálculos, transformaciones de datos y comparaciones pertinentes.

2.8.1 Modelo de diseño

Es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto a otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. El modelo de diseño sirve de abstracción a la implementación y se utiliza como una entrada fundamental de las actividades de implementación *“El modelo de diseño proporciona detalles sobre la arquitectura del software, estructura de datos, interfaces y componentes que se necesitan para implementar el sistema”* (15).

Diagrama de paquetes:

Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando dependencias entre ellas y suministran una descomposición de la jerarquía de un sistema. A continuación se muestra en la Figura 10 el diagrama de paquetes del componente estadístico.

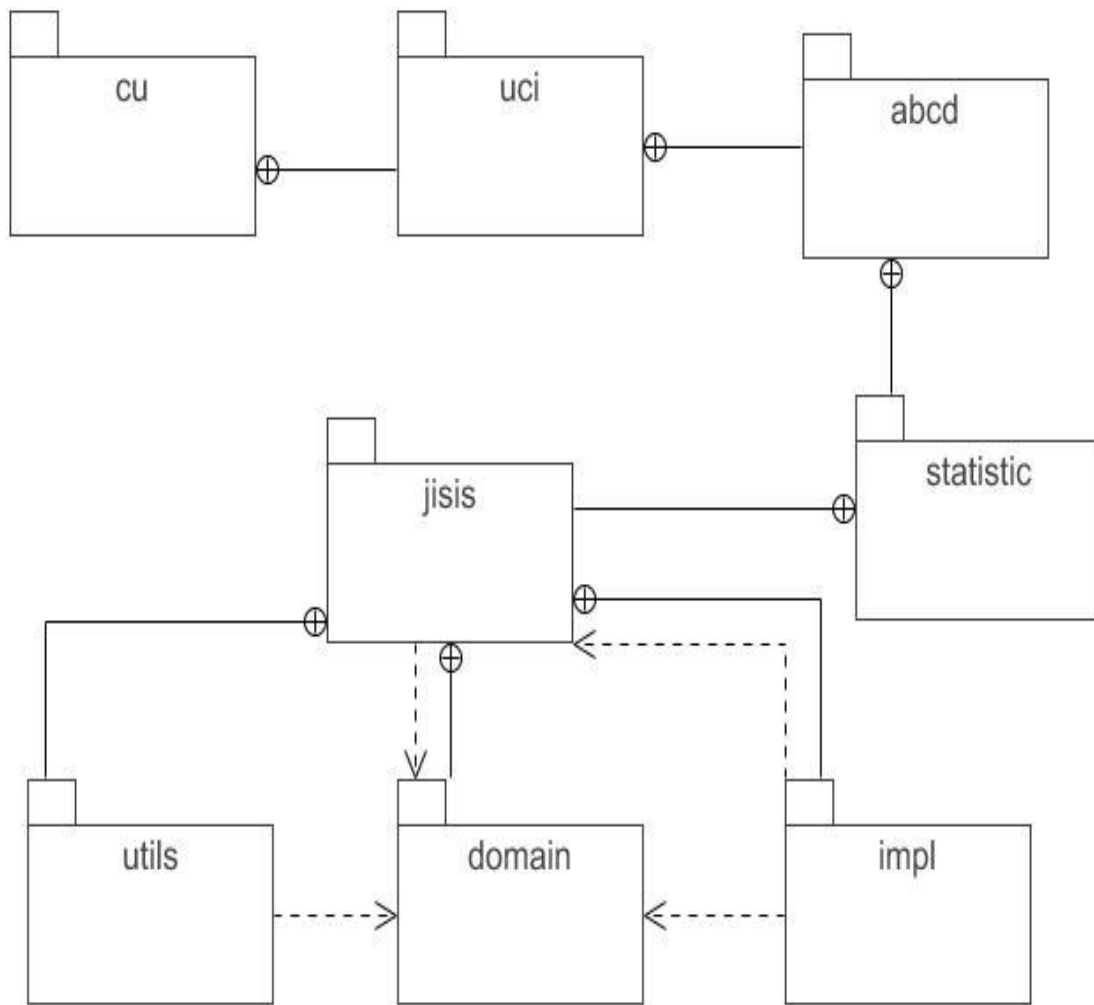


Figura 10. Diagrama de Paquetes (Fuente: Elaboración propia)

Diagrama de clases del diseño:

El diagrama de clases del diseño es una representación correcta de lo que se debe implementar. Estos diagramas representan la parte estadística del sistema a través de la representación de las clases y

sus relaciones (16). A partir de la descripción detallada de los casos de uso del sistema se modeló el diagrama de clases del diseño que se muestra en la Figura 11.

Primeramente se muestra una representación de los paquetes, el nombre de las clases contenidas en ellos y sus relaciones.

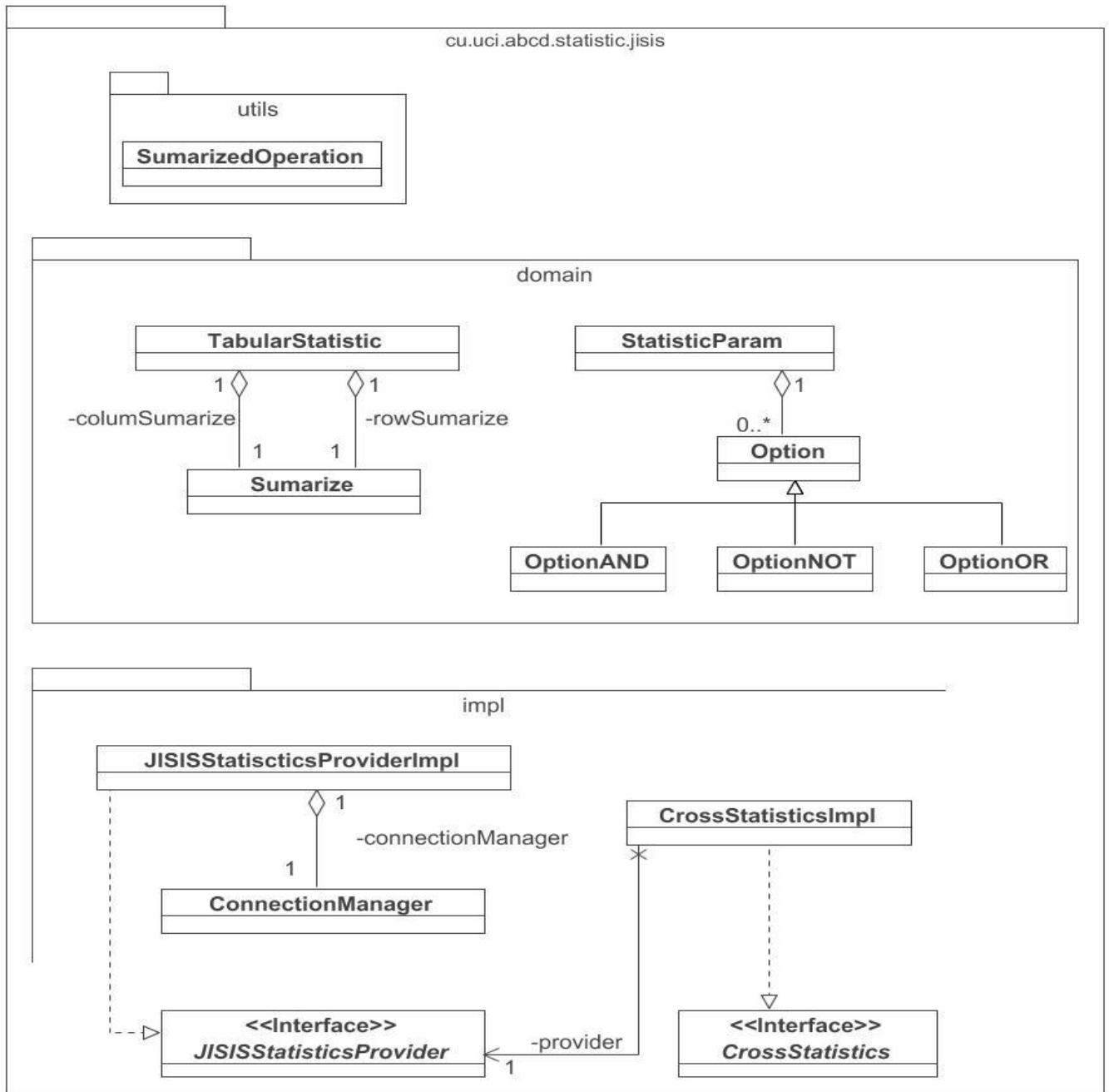


Figura 11. Diagrama de clases del diseño (Fuente: Elaboración propia).

A continuación, se muestran algunas de las clases más importantes del componente.

La clase **TabularStatistic** (ver Figura 12) contiene una serie de atributos, entre los que se encuentran: dos listas con los nombres que corresponden a las variables seleccionadas ubicadas por las filas y las

columnas, la matriz donde serán mostrados los datos, la suma de los valores de las filas, la suma de los valores de las columnas y una suma total. Contiene además los métodos de acceso, utilizados por los objetos externos para mostrar (*get*) o modificar (*set*) el valor de los atributos. La matriz que muestra los datos, se inicializa con el tamaño de las listas de los nombres de las filas y las columnas. En la clase **TabularStatistic** se encuentran además los métodos **addRowName** y **addColumnName** que son utilizados para adicionar valores a la lista de los nombres de las filas y las columnas.

TabularStatistic
-columnNames : List -rowNames : List -data : double[][] -rowSumarize : Sumarize -columnSumarize : Sumarize -total : double
+getRowSumarize() : Sumarize +getColumnSumarize() : Sumarize +getColumnNames() : List +setColumnNames(columnNames : List) : void +getRowNames() : List +setRowNames(rowNames : List) : void +TabularStatistic() +getData() : double [][] +setData(data : double [][]) : void +addColumnName(value : String) : void +addRowName(value : String) : void +setRowSumarize(rowSumarize : Sumarize) : void +setColumnSumarize(columnSumarize : Sumarize) : void +getTotal() : double +setTotal(total : double) : void +setRowSumarize(rowSumarize : Sumarize) : void +setColumnSumarize(columnSumarize : Sumarize) : void

Figura 12. Clase TabularStatistic (Fuente: Elaboración propia)

La clase **JISISStatisticProviderImpl** (ver Figura 13) contiene los atributos que permiten establecer la conexión con la base de datos. Entre sus principales métodos se encuentran:

- Método **checkIndexTags** permite validar si las variables entradas por parámetro están indexadas en la BD (de no estarlo una o ambas variables, se indexan, se guardan los cambios realizados y se re-indexa la BD).
- Método **setHeaders** permite obtener de la lista de *Records* los encabezados de las filas y las columnas (en este método, mediante el uso de las listas Set, se valida que los nombres ubicados en las filas y las columnas no se repitan).
- Método **buildValues** permite obtener un arreglo bidimensional con todos los valores estadísticos para conformar la tabla estadística.

Estos métodos son utilizados por los métodos **getValuesFromLucene**, **getValuesFromMfn** y **getValueFromMfnAndExpression**.

JISISStatisticProviderImpl
<pre> -proxy : ClientDbProxy -connectionManager : ConnectionManager +JISISStatisticProviderImpl() +getFieldSelectionTable(statisticParam : StatisticParam) : FieldSelectionTable +getValueFromMfnAndExpression(statisticParam : StatisticParam, tab : TabularStatistic) : TabularStatistic +getValuesFromLucene(statisticParam : StatisticParam, tab : TabularStatistic) : TabularStatistic +getValuesFromMfn(statisticParam : StatisticParam, tab : TabularStatistic) : TabularStatistic +getConnectionManager() : ConnectionManager +setConnectionManager(connectionManager : ConnectionManager) : void -findMfnByOptions(statisticParam : StatisticParam) : long [] -findRecordsByOptions(statisticParam : StatisticParam) : List -getRecords(statisticParam : StatisticParam, begin : int, end : int) : List -getRecords(statisticParam : StatisticParam, mfn : long []) : List -buildLuceneQuery(statisticParam : StatisticParam) : String -getProxy(statisticParam : StatisticParam) : ClientDbProxy -setHeaders(statisticParam : StatisticParam, records : List, rows : List, columns : List) : void -buildValues(statisticParam : StatisticParam, tab : TabularStatistic, rows : List, columns : List, records : List) : TabularStatistic +checkIndexTags(statisticParam : StatisticParam) : void -RowsColumsChanged(rows : List, colums : List, statisticParam : StatisticParam) : void </pre>

Figura 13. Clase JISISStatisticProviderImpl (Fuente: Elaboración propia)

La clase **StatisticParam** (ver Figura 14) posee los siguientes atributos: el valor de la variable fila, el valor de la variable columna, inicio y fin del rango MFN, el nombre de la base de datos, la dirección de la base de datos y la lista de opciones correspondiente. Cuenta además con los métodos de acceso, utilizados por los objetos externos para mostrar (*get*) o modificar (*set*) el valor de los atributos. Esta clase es la que captura los parámetros de entrada según el constructor que es ejecutado.

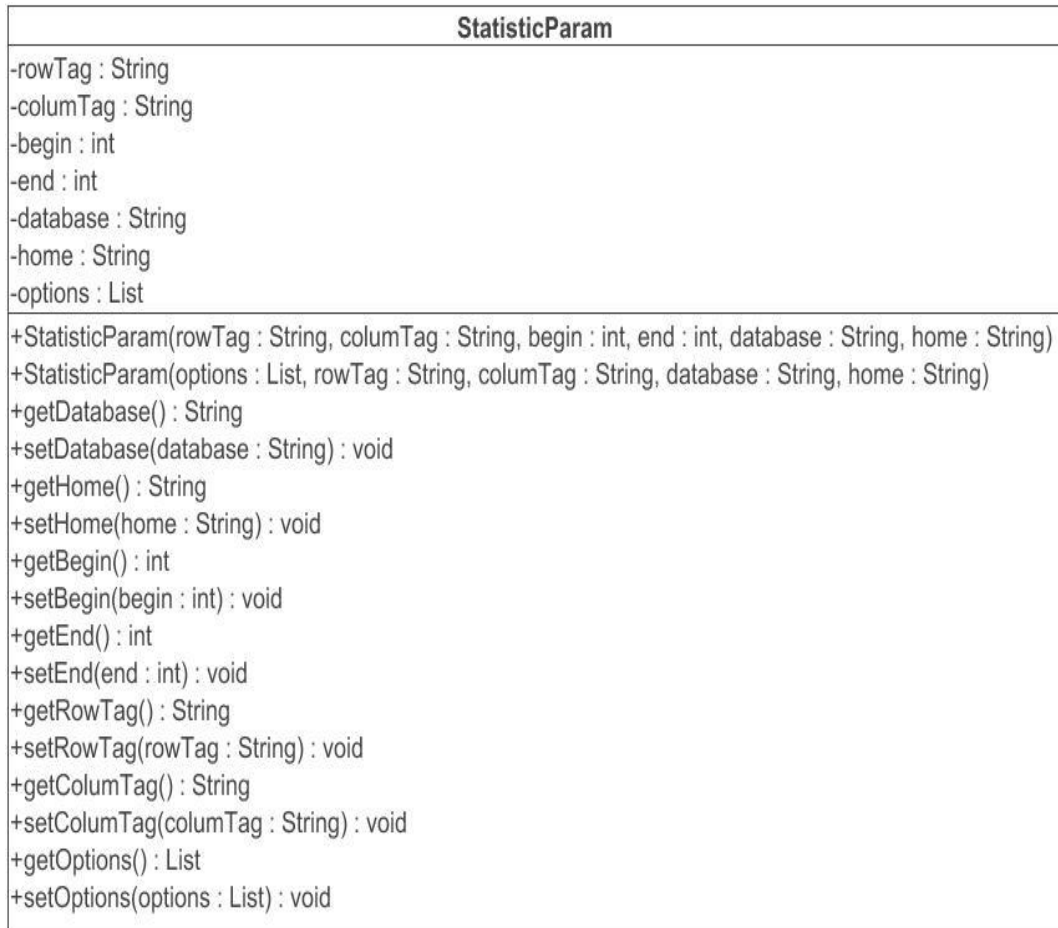


Figura 14. Clase *StatisticParam* (Fuente: Elaboración propia)

La clase **SumarizedOperation** (ver Figura 15) es la que contiene los métodos necesarios para obtener la suma de los valores de las listas de los totales de las filas y las columnas. Obtiene la suma del total general de ambas listas de totales.

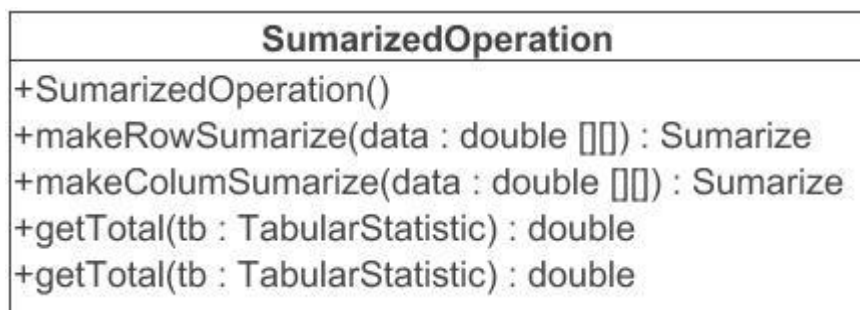


Figura 15. Clase *SumarizedOperation* (Fuente: Elaboración propia)

La clase **CrossStatisticImpl** (ver Figura 16) es donde se conforma la estadística completa mediante el método **buildStatistics**, este método devuelve un objeto **TabularStatistic**. Además se encuentran otros métodos que se utilizan para obtener el resultado final como **sumarize** y **getStructure**, este último

es el que especifica según los parámetros de entradas que tipo de búsqueda es realizada para obtener la estadística.

CrossStatisticsImpl
-provider : JISISStatisticsProvider
+CrossStatisticsImpl() +buildStatistics(statisticParam : StatisticParam) : TabularStatistic -sumarize(tb : TabularStatistic) : TabularStatistic -getStructure(statisticParam : StatisticParam) : TabularStatistic +getProvider() : JISISStatisticsProvider +setProvider(provider : JISISStatisticsProvider) : void

Figura 16. Clase CrossStatisticImpl (Fuente: Elaboración propia)

2.9 Conclusiones parciales

En este capítulo se realizó una caracterización de la propuesta de solución del componente, que facilitó la comprensión de la generación de estadísticas dinámicas para su posterior implementación. El diagrama del modelo del dominio propició una representación visual de las clases conceptuales del entorno. Se realizó la revisión de los CU, lo cual facilitó el entendimiento del flujo del sistema. Se elaboraron los diagramas de clases del diseño y el diagrama de paquete, lo que propició un modelo estructurado de las clases encargadas de ejecutar las responsabilidades del negocio.

Capítulo 3. Implementación y prueba

3.1 Introducción

En este capítulo se desarrolla el flujo de trabajo de la implementación y se describen sus principales artefactos. Se realiza el diagrama del despliegue y el diagrama de Componentes. Se describen paso a paso algunos de los métodos principales en el desarrollo del componente. Se realizan pruebas unitarias y de rendimiento al componente, mostrando además los resultados obtenidos.

3.2 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, se implementan en términos de componentes. Representan la organización de los mismos de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizado, así como las dependencias y los recursos necesarios para poder ejecutar el sistema desarrollado. Esta descripción es de gran utilidad a la hora de implementar el componente, facilita la organización del trabajo y lo hace más entendible a los desarrolladores.

3.2.1 Diagrama de Despliegue

Un diagrama de despliegue muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución en tiempo real en las instancias de los nodos de proceso (27). El diagrama de despliegue se utilizó para capturar los elementos de configuración del procesamiento y las conexiones entre estos. Se aplicó además, para visualizar la distribución de los componentes de software en los nodos físicos (Ver Figura 17). Se cuenta con un servidor web Virgo v3.6.3 donde es montado el componente, el cual se conecta al servidor de bases de datos J-ISIS, mediante el protocolo TCP/IP y puerto 1111.



Figura 17. Diagrama del despliegue (Fuente: Elaboración propia)

3.2.2 Diagrama de componente

Los diagramas de componentes describen como se organizan los componentes, de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes unos de otros (Ver Figura 18). Debido a que los diagramas de componentes son más parecidos a los diagramas de casos de usos, éstos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema (27).

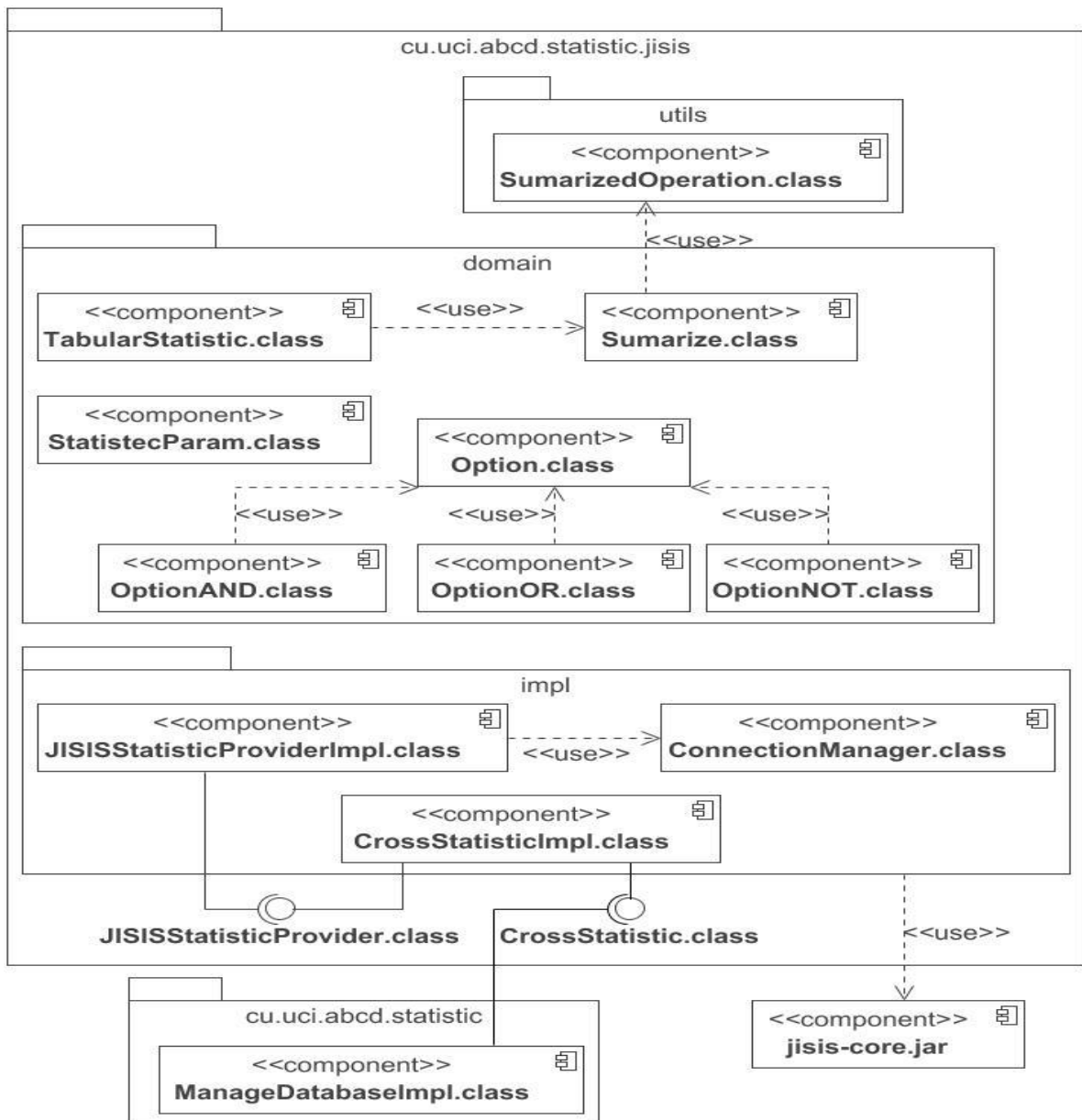


Figura 18. Diagrama de componentes (Fuente: Elaboración propia)

3.3 Estándares de codificación

Los estándares de codificación se definen por el equipo de desarrollo para lograr estandarización en la programación del software. Estos se basan en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. La generalización de aspectos tan simples como el trato de las mayúsculas, ayuda a eliminar conflictos de funcionalidades implementadas con nombres iguales y guían de forma clara el proceso de desarrollo.

A continuación se muestran algunos de los estándares de codificación utilizados en el desarrollo del componente estadístico:

- La nomenclatura de las clases está definida por la notación *PascalCasing*, la cual define que los nombres e identificadores pueden estar compuestos por múltiples palabras juntas y la primera letra de cada palabra debe ir siempre en mayúscula, además se obvia el uso de artículos.

Ejemplo: *JISISStatisticProvider*, en este caso el nombre de la clase está compuesta por 3 palabras y cada una de ellas inicia con letra mayúscula.

- De manera general el nombre de todos los métodos y los atributos de las clases se escriben con la inicial del identificador en minúscula, en casos de nombres compuestos es similar con la diferencia de que la letra inicial de la segunda palabra comienza con letra mayúscula, para ellos se utiliza la notación *CamelCasing*.

Ejemplo: `getProxy` y `fstNames`.

3.4 Implementaciones relevantes en Java

A continuación se describen en pasos la implementación de los métodos *checkIndexTags*, *setHeaders* y *buildValues* los cuales se encuentran en la clase *JISISStatisticProviderImpl*, perteneciente al *bundle cu.uci.abcd.statistic.jisis.impl* donde se implementa el servicio que brinda acceso a la base de datos no relacional:

Método **checkIndexTags** (Ver Figura 19):

- Se obtiene el arreglo de nombres del FST de la base de datos de J-ISIS.
- Se toma el primer FST de la base de datos.
- Se convierte a entero el número del campo de la fila y la columna.
- Se toma el arreglo de los números de los campos indexados en el FST.
- Se crean variables booleanas para saber si están indexados los campos.

- Se recorre el arreglo de número de campos indexados y se pregunta si los campos de los indicadores están indexados.
- Si el campo del indicador de la fila no está indexado, se indexa.
- Si el campo del indicador de la columna no está indexado, se indexa.
- Luego de indexar los indicadores, se guarda el FST y se re-indexa la base de datos.

```

public void checkIndexTags(StatisticParam statisticParam) throws Exception {

    String[] fstNames = getProxy(statisticParam).getFstNames();
    FieldSelectionTable fst = getProxy(statisticParam).getFst(fstNames[0]);
    int rowTag = Integer.parseInt(statisticParam.getRowTag());
    int columnTag = Integer.parseInt(statisticParam.getColumnTag());

    int tags[] = fst.getEntriesTag();

    boolean existRowTagInFst = false;
    boolean existColumnTagInFst = false;

    for (int j = 0; j < tags.length; j++) {
        if (rowTag == tags[j] && !existRowTagInFst)
            existRowTagInFst = true;
        if (columnTag == tags[j] && !existColumnTagInFst)
            existColumnTagInFst = true;
    }

    if (!existRowTagInFst)
        fst.addEntry(rowTag, "", 4, "v" + rowTag);

    if (!existColumnTagInFst)
        fst.addEntry(columnTag, "", 4, "v" + columnTag);

    if (!existRowTagInFst || !existColumnTagInFst) {
        getProxy(statisticParam).saveFieldSelectionTable(fst);
        getProxy(statisticParam).buildIndex();
    }
}
}
}

```

Figura 19. Implementación del método `checkIndexTags` (Fuente: Elaboración propia)

Método **setHeaders** (Ver Figura 20):

- Primeramente se crean dos listas Set para que los nombres de las filas y las columnas no se repitan.

- Se recorre la lista de Records para llenar las Listas Set con los valores de los indicadores.
- Se validan que los valores correspondientes a los indicadores no sean null ni vacíos.
- Se crean dos iteradores para recorrer las listas Set.
- Se adiciona a las listas de las filas y columnas, en la primera posición, el valor: "No hay Datos".
- Se adiciona a las listas de las filas y las columnas los elementos de las listas Set.

```
private void setHeaders(StatisticParam statisticParam, List<Record> records, List<String> rows, List<String> columns) {
    Set<String> rowsTemp = new HashSet<>();
    Set<String> columnsTemp = new HashSet<>();

    for (Record record : records) {
        String value;
        try {
            value = record.getField(Integer.parseInt(statisticParam.getRowTag())).getStringFieldValue();
            if (value != null && !value.isEmpty()) {
                rowsTemp.add(value);
            }
        } catch (NullPointerException | NumberFormatException | DbException e) {
            e.printStackTrace();
        }

        try {
            value = record.getField(Integer.parseInt(statisticParam.getColumnTag())).getStringFieldValue();
            if (value != null && !value.isEmpty()) {
                columnsTemp.add(value);
            }
        } catch (NullPointerException | NumberFormatException | DbException e) {
            e.printStackTrace();
        }
    }

    Iterator<String> rowIT = rowsTemp.iterator();
    Iterator<String> colIT = columnsTemp.iterator();

    rows.add("No hay Datos");
    columns.add("No hay Datos");

    for (int i = 0; i < rowsTemp.size(); i++)
        rows.add(rowIT.next());

    for (int i = 0; i < columnsTemp.size(); i++)
        columns.add(colIT.next());
}
}
```

Figura 20. Implementación del método setHeaders (Fuente: Elaboración propia).

Método **buildValues** (Ver Figura 21):

- Primeramente se pasa al objeto `TabularStatistic` el valor de los nombres de las filas y las columnas.
- Se crea un arreglo bidimensional y se inicializa con el tamaño de las listas de las filas y las columnas.
- Se recorre la lista de `Records` para obtener los valores a tener en cuenta en la estadística.
- Si el valor de la fila y la columna, del `Record` en cuestión, son vacíos se adiciona en la intercepción de `No hay Datos [0] [0]`.
- Si el valor de la columna es vacío se recorre la fila para saber en qué posición sumar `[i] [0]`.
- Si el valor de la fila es vacío se recorre la columna para saber en qué posición sumar `[0] [i]`.
- Si los dos valores contienen datos, se recorren las filas y las columnas para añadir el valor estadístico.
- Se le pasa al objeto `TabularStatistic` el arreglo bidimensional con todos los valores llenos.

```

private TabularStatistic buildValues(StatisticParam statisticParam, TabularStatistic tab, List<String> rows, List<String> columns,
    List<Record> records) throws IOException {
    tab.setRowNames(rows);
    tab.setColumnNames(columns);
    double[][] data = tab.getData();

    for (Record record : records) {

        if (**.equals(record.getField(Integer.parseInt(statisticParam.getRowTag())).getStringFieldValue())
            && **.equals(record.getField(Integer.parseInt(statisticParam.getColumnTag())).getStringFieldValue())) {
            data[0][0] += 1;

        } else if (**.equals(record.getField(Integer.parseInt(statisticParam.getColumnTag())).getStringFieldValue())) {

            for (int i = 1; i < data.length; i++)
                if (rows.get(i).equals(record.getField(Integer.parseInt(statisticParam.getRowTag())).getStringFieldValue()))
                    data[i][0] += 1;

        } else if (**.equals(record.getField(Integer.parseInt(statisticParam.getRowTag())).getStringFieldValue())) {

            for (int i = 1; i < data[0].length; i++)
                if (columns.get(i).equals(record.getField(Integer.parseInt(statisticParam.getColumnTag())).getStringFieldValue()))
                    data[0][i] += 1;

        } else {
            for (int i = 1; i < data.length; i++)
                for (int j = 1; j < data[i].length; j++)
                    if (rows.get(i).equals(record.getField(Integer.parseInt(statisticParam.getRowTag())).getStringFieldValue())
                        && columns.get(j).equals(record.getField(Integer.parseInt(statisticParam.getColumnTag())).getStringFieldValue(
                            )))
                        data[i][j] += 1;
        }
    }

    tab.setData(data);
    return tab;
}

```

Figura 21. Implementación del método buildValue (Fuente: Elaboración propia)

2.4.1 Tratamiento de excepciones.

Una excepción es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias, estas no se pueden ignorar porque harán que el programa aborte. El tratamiento de excepciones es la detección de errores que una aplicación debería manejar de forma razonable (28). En el sistema el proceso de detección y tratamiento de errores se realiza a través de la utilización del bloque de instrucciones try-catch (Ver Figura 22), el mecanismo es utilizado en aquellas partes del código que puedan generar algún tipo de error, a continuación se muestra su utilización. En este caso específico el bloque de instrucciones captura y trata las excepciones que puedan ocurrir en el proceso de obtener la variable de la fila validando que el valor pasado por parámetro no sea nulo ni vacío.

Ejemplo del tratamiento de excepciones en el desarrollo del componente:

```
try {
    value = record.getField(
        Integer.parseInt(StatisticParam.getRowTag()))
        .getStringFieldValue();
    if (value != null && !value.isEmpty()) {
        rowsTemp.add(value);
    }
} catch (NullPointerException | NumberFormatException | DbException e) {
    e.printStackTrace();
}
```

Figura 22. Ejemplo del tratamiento de excepciones (Fuente: Elaboración propia).

3.5 Estrategias de pruebas

Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir (15).

Objetivo de realizar las estrategias de pruebas: Las estrategias de software se hacen con el objetivo de que el producto del software que se encuentre en desarrollo, reúna todos los requisitos planteados por el cliente mediante la lógica del negocio (15).

3.6 Pruebas de software

Las pruebas de software son un elemento esencial y crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. La prueba se enfoca sobre la lógica interna del software y las funciones externas. Es un proceso que tiene como objetivo la ejecución de un programa con la intención de descubrir un error. La fase de pruebas es una de las más valiosas del ciclo de vida de un software y en ese sentido, deben evaluarse todos los artefactos generados, lo que incluye especificaciones de requisitos, diagramas de diversos tipos, el código fuente y el resto de productos que forman parte de la aplicación. Según Pressman *“Las pruebas no pueden asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software”* (15).

3.6.1 Tipos de pruebas usados

Pruebas de unidad: Las pruebas de unidad o pruebas unitarias centran el proceso de verificación en la menor unidad del diseño del software: el componente software o módulo. Una prueba de unidad

pretende probar cada función en un archivo de programa simple. Esto quiere decir que un módulo que tiene una prueba de unidad se puede probar independientemente del resto del sistema (15).

Pruebas de rendimiento: Las pruebas de rendimiento se basan en comprobar que el sistema puede soportar el volumen de carga definido en la especificación, es decir, hay que comprobar su eficiencia. Son utilizadas para evaluar el cumplimiento por parte de un sistema o componente, de los requisitos de rendimiento especificados (15). Con ellas se puede determinar cómo responde un sistema ante una cierta carga, así como validar otros atributos relacionados con la calidad, como pueden ser la escalabilidad o el uso de recursos entre otros. Dentro de este tipo se encuentran las pruebas de carga y de estrés, pruebas que serán aplicadas al componente desarrollado.

- **Pruebas de carga:** Las pruebas de carga someten el sistema a cargas de trabajo extremas, determinando la capacidad límite de resistencia del programa. Se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga.
- **Prueba de estrés:** Estas pruebas son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento mediante con la ejecución de un número de usuarios muy superior al esperado. Tienen como finalidad determinar la robustez de una aplicación cuando la carga es extrema y ayuda a determinar si la aplicación se comportará debidamente ante diferentes situaciones.

3.7 Pruebas unitarias

Las pruebas realizadas al componente estadístico fueron automatizadas, para ello se hizo uso del marco de trabajo JUnit para las pruebas unitarias. JUnit brinda un conjunto de bibliotecas que se integran fácilmente al IDE de desarrollo Eclipse, permite además la realización de las pruebas a los métodos de las clases implementadas.

Para hacer uso de JUnit se definieron los siguientes pasos:

- Definir los métodos a probar dentro de cada caso de prueba.
- Realizar pruebas a los métodos.

A continuación en la Tabla 11 se describen las dos iteraciones realizadas para comprobar el correcto funcionamiento del rango MFN, por su parte en la Tabla 12 se describe la iteración realizada al caso de prueba “Rango MFN negativo“:

Tabla 11. Iteraciones para el caso de prueba "Rango MFN válido" (Fuente: Elaboración propia).

#	Caso de prueba	Descripción	Cant. No conformidades	Descripción
1	Rango MFN válido	Se realiza una iteración para comprobar si la aplicación valida correctamente los rangos MFN.	1	Al poner los rangos al revés, o sea en vez de poner de 1 a 5 se puso de 5 a 1, a esto el sistema respondió con normalidad.
2		Se realiza una segunda iteración para comprobar si se le dio correcta solución a la no conformidad detectada en la primera iteración.	0	La no conformidad detectada en la primera iteración fue corregida.

Tabla 12. Iteraciones para el caso de prueba "Rango MFN negativo" (Fuente: Elaboración propia)

#	Caso de Prueba	Descripción	Cant. No conformidades	Descripción
1	Rango MFN negativo	Se realiza una iteración para comprobar si la aplicación valida correctamente los rangos MFN.	0	El componente valida correctamente el rango MFN. Ejemplo: Si el gestor de reportes estadístico define un rango MFN de -1 a 20, el componente solo mostrará el resultado desde el primer MFN de la base de datos hasta el MFN número 20.

3.7.1 Resultado de las pruebas unitarias

A continuación se muestran los resultados de aplicar las pruebas a la clase más significativa que presenta el componente, *JISISStatisitcProviderImpl*, en general el resultado de la prueba realizada a todos los métodos de esta clase fue satisfactorio. Se realiza una primera iteración donde fue detectada

una no conformidad (Ver Figura 23), posteriormente se realiza una segunda iteración para verificar que la no conformidad detectada fue corregida (Ver Figura 24).

Se detectó una única no conformidad, la misma fue resuelta en la iteración donde fue encontrada. La Figura 25 muestra una gráfica con la distribución por iteraciones de las no conformidades encontradas durante la ejecución de las pruebas unitarias al componente estadístico.

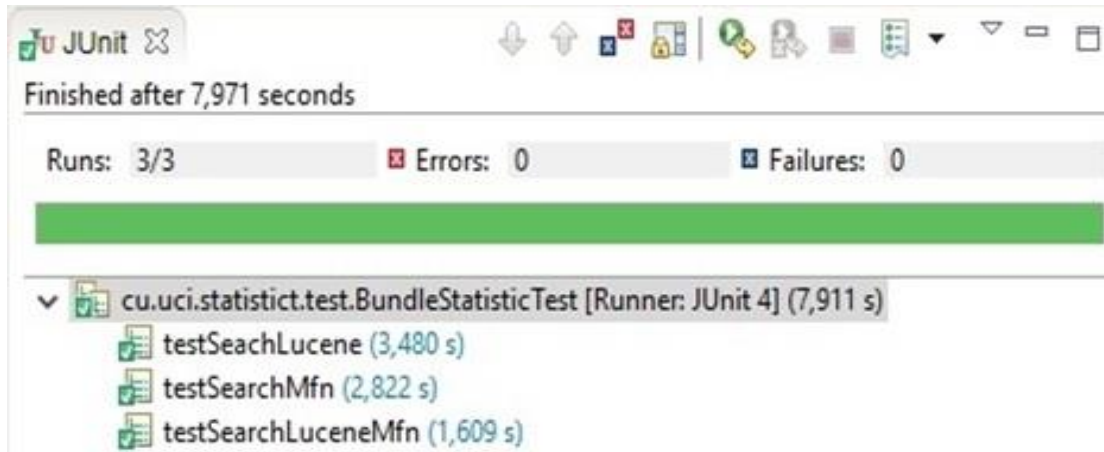


Figura 23. Prueba con JUnit primera iteración (Fuente: Elaboración propia)

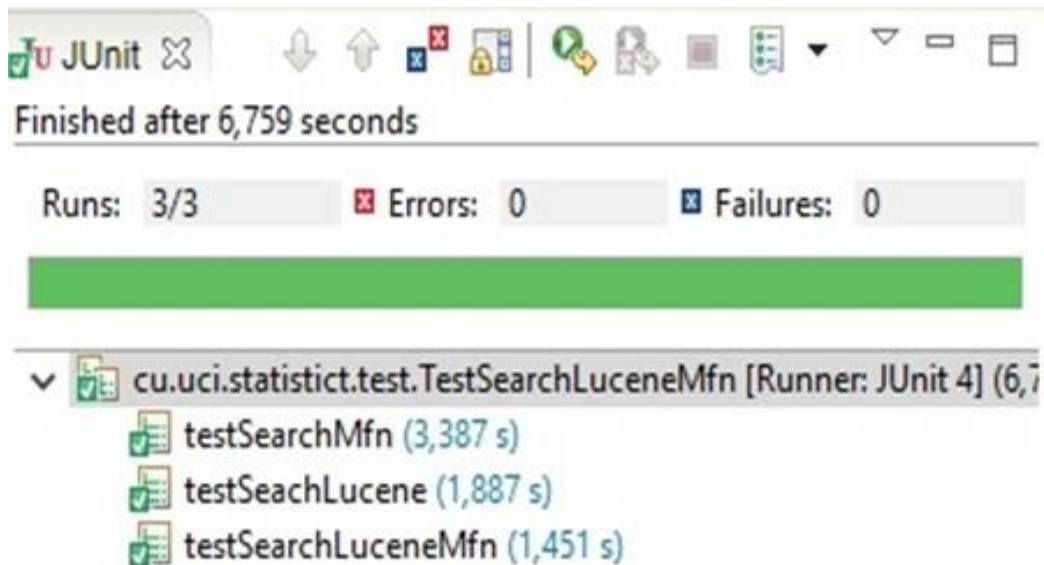


Figura 24. Prueba con JUnit segunda iteración (Fuente: Elaboración propia)

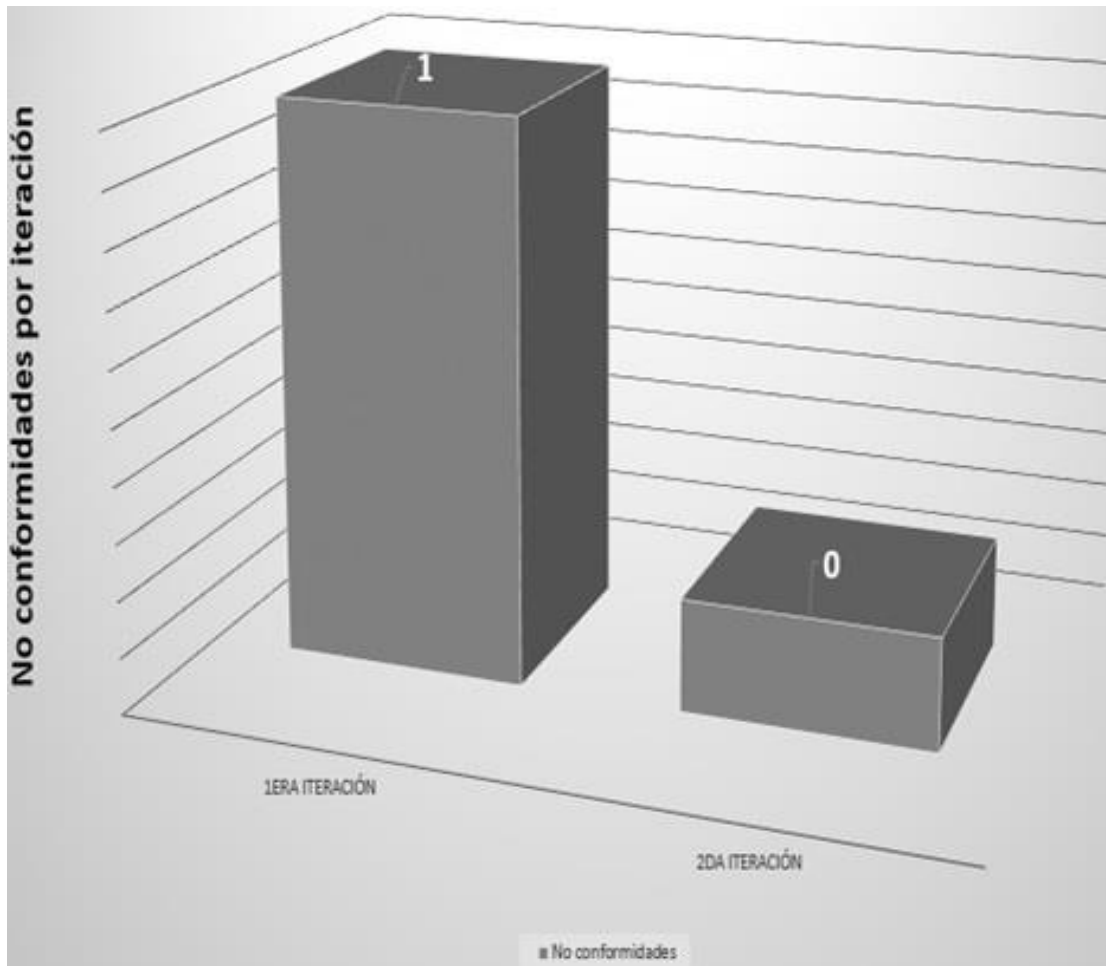


Figura 25. No conformidades obtenidas en las pruebas unitarias (Fuente: Elaboración propia)

3.8 Pruebas de rendimiento

Para la realización de las pruebas se hizo necesario tener en cuenta las condiciones del escenario, tanto del hardware como software, donde se encuentra la aplicación; para obtener una correcta información de comportamiento y resultados en general. Por tanto se hizo necesario simular las pruebas en un escenario con las características siguientes:

Hardware:

- **Tipo de procesador:** Intel (R) Celeron (R) CPU N2830 @ 2.16GHz
- **Memoria:** 4.00GB
- **Tipo de red:** Ethernet 10 / 100Mbps.

Software:

- **Tipo de servidor:** Virgo v3.6.3
- **Máximo de hilos concurrentes:** 5 y 10 (respectivamente a cada escenario de prueba).
- **Plataforma:** Windows 10

➤ **Lenguaje:** Java

3.8.1 Resultados de las pruebas de rendimiento

A continuación se muestran los resultados obtenidos luego de ejecutadas las pruebas de rendimiento sobre el componente desarrollado por cada una de las iteraciones. La información fue obtenida por medio del componente “Informe Agregado” que proporciona la herramienta de pruebas JMeter.

Primera iteración: la primera iteración fue realizada simulando 5 usuarios conectados al componente realizando búsquedas por rango MFN. En la Figura 26 se muestran los valores obtenidos:

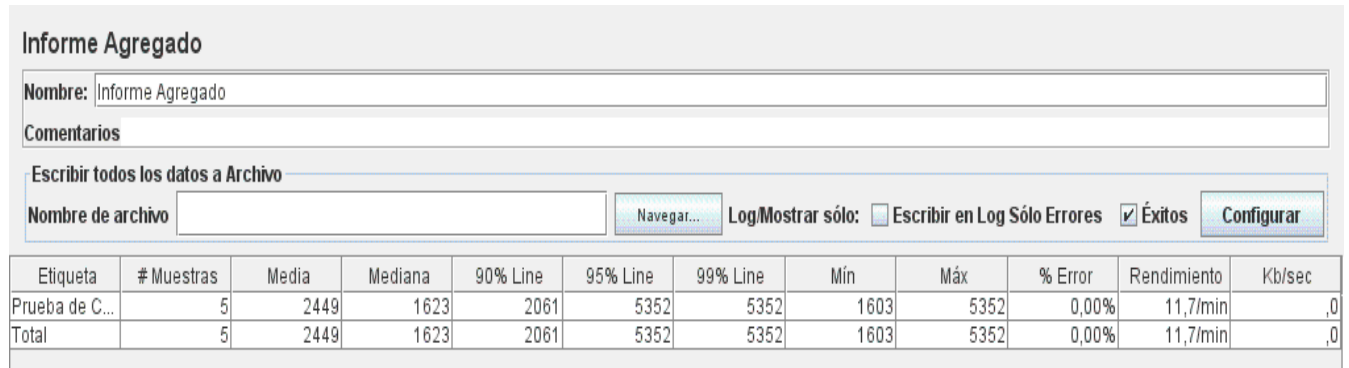


Figura 26. Resultado de la prueba de carga en la primera iteración (Fuente: Elaboración propia)

La primera iteración fue realizada con un total de 5 usuarios conectados en un período de 10 segundos, aunque no se produjeron errores, se obtuvo un tiempo desfavorable de 11.7min.

Segunda iteración: la segunda iteración fue realizada simulando a 10 usuarios conectados al componente realizando búsquedas por rango MFN. De igual forma, en la Figura 27 se muestran los resultados obtenidos:

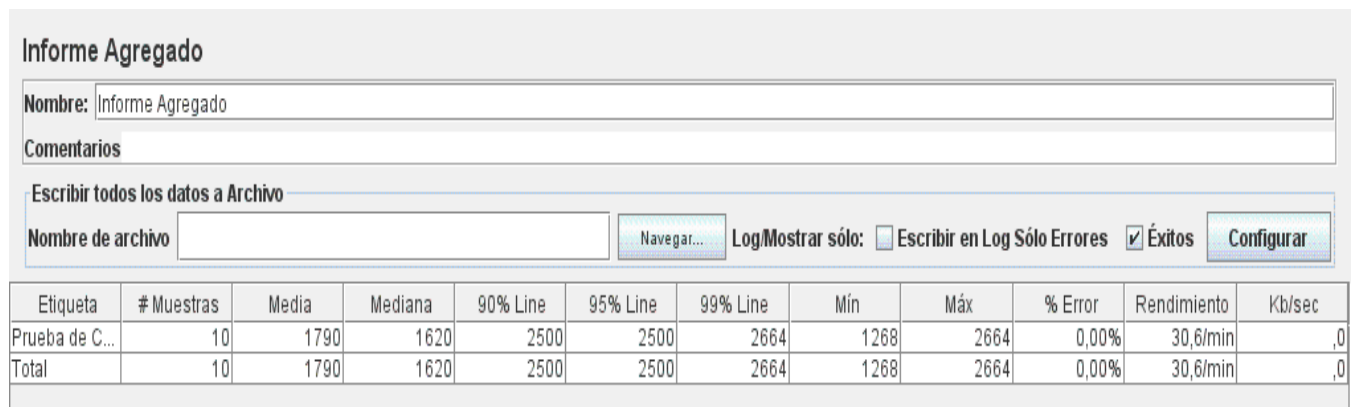


Figura 27. Resultado de la prueba de carga en la segunda iteración (Fuente: Elaboración propia)

La segunda iteración realizada con un total de 10 usuarios conectados en un período de 20 segundos, aunque no se produjeron errores, se obtuvo un tiempo desfavorable de 30.6 min.

Tercera iteración: la tercera fue realizada simulando a 5 usuarios conectados de forma concurrente al componente realizando búsquedas por rango MFN. De igual forma, en la Figura 28 se muestran los resultados obtenidos:

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
Prueba de C...	5	17912	17919	18038	18315	18315	17526	18315	100,00%	16,3/min	,0
TOTAL	5	17912	17919	18038	18315	18315	17526	18315	100,00%	16,3/min	,0

Figura 28. Resultado de la prueba de carga en la tercera iteración (Fuente: Elaboración propia)

La tercera iteración realizada con un total de 5 usuarios conectados de forma concurrente al sistema, arrojó un 100% de error, debido a que el servidor J-ISIS no soporta peticiones concurrentes. A continuación en la Figura 29 se presenta un mensaje de error mostrado por J-ISIS al realizar la tercera iteración:



Figura 29. Excepción mostrada por J-ISIS al realizar peticiones concurrentes (Fuente: Elaboración propia)

3.9 Conclusiones parciales

En el presente capítulo se definieron los estándares de codificación utilizados, lo que permitió organizar el código durante la implementación. La modelación de los diagramas de componentes y de despliegue facilitó la implementación del componente estadístico. La aplicación de las pruebas unitarias permitió comprobar el correcto funcionamiento de los métodos implementados en el desarrollo del componente. La aplicación de las pruebas de rendimiento permitió conocer que el servidor J-ISIS no soporta peticiones concurrentes.

CONCLUSIONES GENERALES

Conclusiones generales:

La investigación desarrollada y los resultados obtenidos permiten a los autores plantear las siguientes conclusiones:

- Se realizó un análisis de los principales conceptos relacionados con el objeto de estudio, lo que proporcionó la base teórica para el desarrollo del componente.
- El estudio sobre las funcionalidades básicas y la estructura de las consultas Lucene, permitió obtener el conocimiento necesario para la realización de consultas en las bases de datos de J-ISIS.
- El estudio del estado del arte sirvió de guía para la implementación del componente desarrollado.
- El desarrollo de los artefactos correspondientes a la metodología de desarrollo RUP, permitió un mejor entendimiento de la generación de estadísticas dinámicas.
- La modelación de los diagramas de componentes y de despliegue facilitó la implementación del componente estadístico.
- Se implementó el componente estadístico, lo que permitió la realización de las estadísticas dinámicas con cruzamiento de dos variables.
- La validación del componente mediante la aplicación de las pruebas de software arrojó resultados satisfactorios, demostrándose su fiabilidad.

RECOMENDACIONES

Recomendaciones

- Integrar el componente al módulo estadística del Sistema ABCD v3.0.

REFERENCIAS BIBLIOGRÁFICAS

1. Universidad de las Ciencias Informáticas. *UCI*. [En línea] 2015. [Citado el: 05 de 15 de 2016.] <http://www.uci.cu/sites/default/files/Cat%C3%A1logo%202015.pdf>.
2. Quesada Ibarguen, Victor Manuel y Vergara Schmalbach, Juan Carlos. *Estadística básica con aplicaciones en Ms Excel*. 2009. pág. 178. ISBN: 978-84-690-5503-8.
3. Berdina, Xavier y Ferré, Mercé. *Estadística Descriptiva*. Barcelona : Servei de publicacions, 2009. 978-84-2590-7.
4. M. Ross, Sheldon. *Introducción a la Estadística*. España : Editorial Reverté, 2007. 978-84-291-5039-1.
5. S. Morroe, David. *Estadística aplicada básica*. Barcelona : Antoni Bosh, 2000. 84-95348-04-7.
6. González Rodríguez, Benito J, y otros. *Variables Estadísticas Bidimensionales*. s.l. : Uni>ersia, 2013.
7. Ramírez Valenzo, Marvin, Cuevas Valencia, Rene E y Martínez Castro, José Mario. *Integración de búsquedas de texto completo en Bases de Datos NoSQL*. Unidad Académica de Ingeniería, Universidad Autónoma de Guerrero. México : Revista Vínculos, 2011.
8. Koha. Koha Library Software. [En línea] 2016. [Citado el: 21 de 06 de 2016.] <https://koha-community.org/>.
9. SAS, PMB Services. PMB services. [En línea] 2004. [Citado el: 21 de 06 de 2016.] <http://www.sigb.net/index.php?lvl=cmspage&pageid=15>.
10. Smet, Egbert. *The abc of ABCD*. 2013.
11. Proyecto ABCD. *Casos de Uso módulo estadística. Proyecto de Automatización y Centros de Documentación.ABCDv3.0*. Ciudad de La Habana : CIGED.
12. Dauphin, Jean-Claude. *J-ISIS Reference Manual*. 2014.
13. Ramos Hernández, Juan Pablo y Alor Hernández, Giner. *Indización y Búsqueda a través de Lucene*. Sinaloa : Univrsidad Autónoma de Sinaloa, 2008.
14. Cogoluegnes, Arnaud, Templier, Thierry y Piper, Andy. *Spring Dynamic Modules in Action*. s.l. : Manning Publications Co., 2011.
15. Pressman, Roger S. *Ingeniería de Software, un enfoque práctico. 7th Edición*. s.l. : McGraw-Hill Companies, 2012.

16. —. *Ingeniería de Software, un enfoque práctico. 5th Edición.* s.l. : McGraw-Hill Companies, 2002.
17. Larman, Craig. *UML y Patrones una introducción al análisis y diseño orientado a objeto y al proceso unificado, Segunda edición.* Madrid : PEARSON EDUCIACIÓN, S.A., 2003.
18. Montero Garrido, Jasús Manuel. *Plataforma Eclipse. Introducción Técnica.*
19. VP. Visual Paradigm. [En línea] 2015. [Citado el: 2 de 2 de 2016.] <https://www.visual-paradigm.com>.
20. Belmonte Fernández, Oscar. *Introducción al lenguaje de programación Java. Una guía básica.* 2005.
21. Oracle. [En línea] Oracle, 2016. [Citado el: 10 de 06 de 2016.] <http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html>.
22. Foundations, The Eclipse. Eclipse. [En línea] 2016. [Citado el: 3 de 2 de 2016.] www.eclipse.org/virgo/documentation.
23. JUnit 4. [En línea] 18 de 4 de 2016. [Citado el: 2 de 6 de 2016.] <http://junit.org/junit4/>.
24. Foundation, The Apache Software. Apache JMeter. [En línea] ApacheCon , 9 de Mayo de 2013. [Citado el: 13 de 5 de 2016.] <http://jmeter.apache.org/>.
25. Fowler, Martin. *"UML Distilled: A brief Guide to the standard object modeling language".* s.l. : Addison-Wesley, 2004.
26. Pytel, Pablo, y otros. *Ingeniería de requisitos basada en técnicas de ingeniería del documento.* Costa Rica : RedUNCI, 2011. 978-950-673-892-1.
27. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El lenguaje unificado del modelado.* España : Addison-Wesley, 2000.
28. León, Pedro J. Ponce de. *Gestión de errores.* s.l. : Alicante, 2011.
29. Martínez, Ana María, Aguirre, Natalia y Fager, José. *Manual de catalogación automatizada.* Montevideo : CSE-UCUr, 2011. 978-9974-0-0761-1.
30. Hagos Berche, Helen. *Extension with Digital Library Technology of UNESCO's J-ISIS Database Software.* s.l. : Virje Universiteit Brussel, 2012.
31. Parrado Prieto, José Ángel. *Sistemas Integrados de Gestión Bibliotecaria: libres y de código abierto.* Facultad de Filosofía y Letras. España : Universidad de León, 2011/2012. pág. 11, Tesis.

BIBLIOGRAFÍA

1. Overgaard, Karin. (2004). Use cases Patterns and blueprints. Addison Wesley Professional.
2. Paz de Erickson, Ana María. (1983). Revista de la asociación interamericana de bibliotecarios y documentalistas agrícolas. VI.5
3. Molpeceres, Alberto. (2003). Proceso de desarrollo: RUP, XP y FDD.
4. Sánchez Rodríguez, Tamara. (2015). Metodología de desarrollo para las actividades productivas de la UCI. La Habana, Cuba.
5. Nowicki, Zbigniew M. (2001) Manual de usuario del WinISISv1.4. Editorial Universitaria. Ciudad de la Habana.
6. S. Pressman, Roger. (2002). Ingeniería de Software. Un enfoque práctico. Addison Wesley. McGrawGil.
7. Hernández, Giner Alor; Ramos Hernández, Juan Pablo (1998). Indexación y búsqueda a través de Lucene. Sinaloa.
8. F. Javier Díaz, Claudia M. Tzancoff Banchoff, Anahí S. Rodríguez, Valeria Soria. Usando JMeter para pruebas de rendimiento. La Plata: s.n.
9. The Apache Software Foundation. [En línea] 9 de mayo de 2013. [Citado el: 2 de 6 de 2016] <http://jmeter.apache.org/>.
10. Rodríguez, Luis Ernesto. CDS/ISIS para Windows. Colombia: s.n., 2000.
11. UCI. Universidad de las Ciencias Informáticas. Catálogo de Productos y Servicios. [En línea] [Citado el: 3 de 2 de 2016] <http://www.uci.cu/sites/default/files/Cat%C3%A1logo%202015.pdf>.
12. Berdina, Xavier y Ferré, Mercé. *Estadística Descriptiva*. Barcelona: Servei de publicacions, 2009.978-84-2590-7.
13. Collins-Sussman, Ben, W. Ditzpatrick, Brian y Michael Pilato, C. *El control de Versiones con Subversion*. 2004.
14. Díaz, F. Javier, y otros. Usando JMeter para pruebas de software. La Plata: s.n., 1990.
15. Fernández, Santiago, Cordero Sánchez, José María y Córdoba Largo, Alejandro. *Estadística descriptiva*. 2da edición. Madrid: Escuela Superior de Gestión Comercial y Marketing (ESIC), 2002. 84-7356-306-9.
16. Kohan, N y Carró. *Estadística Aplicada*. Buenos Aires: s.n., 1968.
17. Kruchten, Philippe. *The Rational Unified Process an Introduction*. España: Addison Wesley, 2001.
18. Marianela Díaz, Rosales. *Arquitectura orientada a servicios*. La Habana: s.n., 2008.
19. Medina, Ing. Andy Cabrera. *Especificación de casos de uso*. Módulo Estadística. Ciudad de la Habana: ABCD v3.0, 2015.
20. Opera, Luisa. *Reflexiones sobre el concepto de biblioteca*. S.I.: Universidad de Zaragoza.

21. Ramos Hernández, Juan Pablo y Alor Hernández, Giner. *Indexación y Búsqueda a través de Lucene*. Facultad de informática, Universidad Autónoma de Sinaloa: s.n. pág. 19.
22. Salinas, Jesús. *Innovación docente y uso de las TIC en la enseñanza universitaria*. España: RU&SC. Revista de Universidad y Sociedad del Conocimiento, 2004, Vol. 1.
23. Parrado Prieto, José Ángel. *Sistemas Integrados de Gestión Bibliotecaria: libres y de código abierto*. Facultad de Filosofía y Letras. España : Universidad de León, 2011/2012. pág. 11, Thesis.
24. Quesada Ibarquén, Víctor Manuel y Vergara Schmalbach, Juan Carlos. *Estadística básica con aplicaciones en Ms Excel*. 2009. pág. 178. ISBN: 978-84-690-5503-8.
25. M. Ross, Sheldon. *Introducción a la Estadística*. España : Editorial Reverté, 2007. 978-84-291-5039-1.
26. S. Morroe, David. *Estadística aplicada básica*. Barcelona : Antoni Bosh, 2000. 84-95348-04-7.
27. González Rodríguez, Benito J, y otros. *Variables Estadísticas Bidimensionales*. s.l. : Uni>ersia, 2013.
28. González Rodríguez, Benito J, y otros. *Variables Estadísticas Bidimensionales*. s.l. : Uni>ersia, 2013.
29. Ramírez Valenzo, Marvin, Cuevas Valencia, Rene E y Martínez Castro, José Mario. *Integración de búsquedas de texto completo en Bases de Datos NoSQL*. Unidad Académica de Ingeniería, Universidad Autónoma de Guerrero. México : Revista Vínculos, 2011.
30. Martínez, Ana María, Aguirre, Natalia y Fager, José. *Manual de catalogación automatizada*. Montevideo : CSE-UCUr, 2011. 978-9974-0-0761-1.
31. Smet, Egbert. *The abc of ABCD*. 2013.
32. Proyecto ABCD. *Arquitectura de Software. Vista entorno de desarrollo tecnológico. Proyecto de Automatización y Centros de Documentación.ABCDv3.0*. Ciudad de La Habana : CIGED.
33. Dauphin, Jean-Claude. *J-ISIS Reference Manual*. 2014.
34. Hagos Berche, Helen. *Extension with Digital Library Technology of UNESCO's J-ISIS Database Software*. s.l. : Virje Universiteit Brussel, 2012.
35. Pressman, Roger S. *Ingeniería de Software, un enfoque práctico. 7th Edición*. s.l. : McGraw-Hill Companies, 2012.
36. Larman, Craig. *UML y Patrones una introducción al análisis y diseño orientado a objeto y al proceso unificado, Segunda edición*. Madrid : PEARSON EDUCIACIÓN, S.A., 2003.
37. Montero Garrido, Jesús Manuel. *Plataforma Eclipse. Introducción Técnica*.
38. Fowler, Martin. *"UML Distilled: A brief Guide to the standard object modeling language"*. s.l. : Addison-Wesley, 2004.

39. JUnit 4. [En línea] 18 de 4 de 2016. [Citado el: 2 de 6 de 2016.] <http://junit.org/junit4/>
40. Fowler, Martin. *"UML Distilled: A brief Guide to the standard object modeling language"*. s.l. : Addison-Wesley, 2004.
41. Chávez, Michel Arias. *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos del software*. Costa Rica : Universidad de Costa Rica, 2006.
42. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El lenguaje unificado del modelado*. España : Addison-Wesley, 2000.

ANEXOS

Anexo 1

Datos de la entrevista realizada:

Tabla 13. Entrevista

Entrevista	
Objetivo	<ul style="list-style-type: none">➤ Definir los requisitos funcionales y no funcionales del sistema.➤ Definir la arquitectura del componente estadístico.
Día de la entrevista	10 de febrero del 2016
Persona entrevistada	Ing. Oigres Alvarez Pérez (Arquitecto del proyecto)
Lugar de la entrevista	Laboratorio 102, Docente 1, UCI (Puesto de Trabajo)
Modo de realización	Diálogo
Aspectos a tener en cuenta	<ul style="list-style-type: none">-Arquitectura del sistema ABCDv3.0.-Funcionamiento del módulo Estadística, del ABCDv1.2 y ABCDv3.0.-Características del componente a desarrollar.-Herramientas, lenguaje y metodología a utilizar.

Preguntas abordadas en la entrevista:

- 1) ¿Por qué se desea desarrollar un componente estadístico?
- 2) ¿Qué arquitectura presenta el sistema ABCD v3.0?
- 3) ¿Cómo realiza el módulo estadística, del sistema ABCD v1.2 el cruzamiento de las variables?
- 4) ¿Cuáles son las características del componente que se desea desarrollar?
- 5) ¿Cuáles son los requisitos funcionales?
- 6) ¿Qué herramientas usa el proyecto ABCD?
- 7) ¿Qué metodología de desarrollo de software emplea el proyecto ABCD?
- 8) ¿Cuál es el lenguaje de programación a emplear en el desarrollo del componente?

Anexo 2

Circulación Socios Buscar Cesta Más ▾ Biblioteca Kobli (Seleccionar Biblioteca)

Inserte el número de carnet del socio o parte de su apellido

Préstamo Devolución Buscar en el catálogo

Inicio > Informes

Informes

Informes guiados

- Asistente de informes guiados
- Crear nuevo
- Use guardados
- Crear a partir de SQL

Diccionario de informes

- Ver diccionario

Estadísticas Compuestas

- Adquisiciones
- Socios
- Catálogo
- Circulación
- Publicaciones periódicas
- Reservas

Listas principales

- Socios con más préstamos
- Ítems más prestados

Inactivo

- Socios que no han pedido prestado
- Ítems nunca prestados

Otro

- Ítems perdidos
- Catálogo por tipo de ítem
- Tiempo de préstamo promedio
- Esquema de base de datos de Koha
- Biblioteca de reportes Koha

Prezi

Figura 30. Módulo Informes del sistema Koha

Anexo 3

Use an existing table

Create a table

Rows

Figura 31. Creación de una nueva tabla en el sistema ABCD v1.2

Anexo 4

Statistics: marc

[Help](#) [Edit help file](#) Script: tables_generate.php

Use an existing table

List of tables:

Create a table

Generate output

Figura 32. Uso de una tabla existente en el sistema ABCD v1.2

Anexo 5

Generate output

By Mfn

From: To: [Clear](#) (Max. Mfn: 214)

By search

[Clear](#)

Figura 33. Generar salida en el módulo estadística del sistema ABCD v1.2

GLOSARIO DE TÉRMINOS

API: interfaz de programación de aplicaciones.

BD: base de datos.

CASE: ingeniería de software asistida por computación.

HTTP: protocolo de transporte de hiper texto.

HTML: *hypertext markup language*.

IDE: entorno integrado de desarrollo.

IP: protocolo de internet.

MDL: modo de datos.

MHL: modo de partida.

PDF: formato de documento portable.

RUP: proceso unificado de desarrollo

Spring: marco de trabajo de la capa de lógica de negocio para Java.

Tablas *hash*: es un contenedor asociativo que permite un almacenamiento y posterior recuperación eficientes de elementos a partir de otros objetos, llamados claves.

TCP: protocolo de control de transmisión.

XHTML: *extensible hypertext markup language*.

WWW: *world wide web*.