APLICACIÓN WEB PARA EL APRENDIZAJE DE FIGURAS RETÓRICAS

FREDY REINALDO ACOSTA PEÑA

LUIS FELIPE GARCÍA FERNÁNDEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

DE BUCARAMANGA

2015

APLICACIÓN WEB PARA EL APRENDIZAJE DE FIGURAS RETÓRICAS

FREDY REINALDO ACOSTA PEÑA

LUIS FELIPE GARCÍA FERNÁNDEZ

Trabajo de Grado para optar al título de Ingeniero de Sistemas

Directora

SONIA CRISTINA GAMBOA SARMIENTO

Doctora en Educación

Co-Director

MANUEL ARTURO MORENO TARAZONA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

DE BUCARAMANGA

2015

CONTENIDO

Pág.

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

TÍTULO

APLICACIÓN WEB PARA EL APRENDIZAJE DE FIGURAS RETÓRICAS.[1]

AUTORES

FREDY REINALDO ACOSTA PEÑA, LUIS FELIPE GARCÍA FERNÁNDEZ.[2]

PALABRAS CLAVES

Aplicación web, tecnologías de la información, figuras retóricas, software, enseñanza, filosofía.

DESCRIPCIÓN

Hoy en día las aplicaciones web con fines educativos son de gran importancia para las personas involucradas en el proceso del aprendizaje y enseñanza, porque estas brindan un entorno autónomo con amplias funcionalidades y diferentes tipos de enfoque que ayudan a complementar la formación académica del usuario. Este trabajo de grado propone el desarrollo de una aplicación web para el aprendizaje de figuras retóricas partiendo de la necesidad que surge por parte del grupo de Investigación, Filosofía y Enseñanza de la Filosofía con relación a herramientas obsolescentes, las cuales con el tiempo han ido perdiendo usabilidad, diseño y estructura. Por estas razones, este grupo decide dar cabida a estándares globales de programación (JavaEE) y al trabajo de tipo modular que estos ofrecen.

El desarrollo de este proyecto se funda en una metodología basada en la construcción de continuos prototipos, la cual permite visualizar funciones específicas del proyecto por separado, con el fin de identificar nuevos requerimientos y realizar cualquier tipo de modificación sobre la marcha; además de esto, se realiza un análisis del estado del arte actual del proyecto a realizar, con el fin de encontrar funcionalidades reutilizables o que puedan de algún modo, aportar al desarrollo de habilidades para la herramienta del software.

El enfoque principal de este proyecto son las figuras retoricas del silogismo, analogía, falacia y símil, construyendo un modelo pedagógico basado en teoría, ejemplo y ejercicios, profundizando en la generación, construcción y análisis de los diferentes modos de silogismos.

[1] Trabajo de grado

[2] Facultad de Ingeniería Físico-mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Sonia Cristina Gamboa Sarmiento, Doctora en educación.

ABSTRACT

TITLE

WEB APPLICATION TO LEARNING RHETORICAL FIGURES.[3]

AUTHORS

FREDY REINALDO ACOSTA PEÑA, LUIS FELIPE GARCÍA FERNÁNDEZ.[4]

KEYWORDS

Web application, information technology, rhetorical figures, software, teaching philosophy.

DESCRIPTION

Today web applications for educational purposes are very important for persons involved in learning and teaching process, because these provide a standalone environment with extensive functionality and different kind of approach that helps supplement academic training of the user. This thesis proposes a web application development to learning rhetorical figures based on the need that arises from the research group, Philosophy and Philosophy Teaching regarding obsolescent tools, which over time have been lost usability, design and structure. Consequently, this group decides to accommodate global programming standards (JavaEE) and modular work they offer.

The development of this project is grounded in a methodology based on the continuous construction of prototypes, which can display specific project functions separately, in order to identify new requirements and make any modifications along the way; Besides, an analysis of the current state of the art project to be undertaken is made, in order to find reusable functionality or that may in some way contribute to the development of skills for software tool. The main focus of this project are the rhetorical figures of the syllogism, analogy, simile and fallacy, building an educational model based on theory, examples and exercises deepening the generation, construction and analysis of the different syllogisms modes.

[3] Bachelor Thesis.

[4] Facultad de Ingeniería Físico-mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Sonia Cristina Gamboa Sarmiento, Doctora en educación.

INTRODUCCIÓN

Nos encontramos en una época donde prima el uso de la tecnología como herramienta para el servicio de la humanidad, buscando siempre facilitar nuestro trabajo diario, ya sea desde una aplicación que nos permita encontrar trabajo u otras que nos permitan hacer compras desde la comodidad de nuestro hogar. Es así como, también de esta manera, podemos encontrar una inmensa gama de herramientas o software de alta capacidad que nos brinda un apoyo a los estudiantes en el proceso de obtener un aprendizaje más rápido y efectivo de cualquier tema que nos podamos imaginar, en los cuales se busca ofrecer un mayor apoyo o un fortalecimiento de conocimientos, queriendo igualar la interacción que se puede encontrar entre un maestro y sus estudiantes. A medida que pasan los años estas herramientas de la tecnología acompañan la enseñanza y han ido evolucionando siendo una gran aliada tanto para los docentes, facilitándoles trabajo, como para los estudiantes a la hora de reforzar múltiples falencias presentadas.

Es entonces que, mediante este proyecto, se busca integrar el conocimiento sobre figuras retoricas junto al uso de las nuevas tecnologías en una aplicación web que le permita a los estudiantes servirse de una puerta de entrada a las formas de razonamiento argumentativo y con ello se puedan enfrentar de mejor manera a la vida pública y que los desafié a poner a prueba, a consolidar y a obtener nuevos conocimientos, basándose en la incorporación de teoría, ejemplos, ejercicios y problemas que favorezcan el entendimiento de la temática.

Este proyecto consta de seis capítulos. En el primer capítulo se realiza un planteamiento del problema y justificación. En el segundo capítulo se plantean los objetivos propuestos. En el tercer capítulo se mostrará un marco teórico que es la información que se utiliza para el desarrollo de este proyecto. En el cuarto capítulo se encuentra el estado del arte. En el quinto capítulo se encuentra la estructura lógica, en la cual encontraremos los requerimientos funcionales y no funcionales para la hora de la construcción de nuestro software, la explicación clara de la metodología usada (Modelo Vista Controlador) y diagramas de flujo del funcionamiento de nuestro software. En el sexto capítulo se encuentra el diseño computacional (actores y la narrativa de los casos de uso).Para finalizar expresamos nuestras conclusiones obtenidas por la elaboración de este proyecto y una respectiva bibliografía detallada

1. PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

En la enseñanza de la filosofía resulta fundamental el aprendizaje de habilidades propias del filosofar, como son el uso de la lógica, del pensamiento crítico y de la argumentación. Como es conocido, el campo de la lógica se extiende, desde la lógica clásica de primer orden hasta sus aplicaciones multivalentes que corresponden más con las formas de expresión humana. El campo de la lógica se constituye también en la base de toda forma de razonamiento humano que, a su vez, es una de las habilidades fundamentales para la construcción de argumentos, en especial, de argumentos filosóficos; por esto es importante el estudio del uso de las figuras retoricas tales como silogismos, falacias, etc., para la solución efectiva de problemas, extraer conclusiones y aprender de manera consciente de los hechos estableciendo relaciones lógicas entre ellos. Es ahí donde nuestro problema aparece y se pretende buscar un método mediante el uso de las nuevas tecnologías para proporcionar el buen entendimiento de estas herramientas filosóficas.

1.2 JUSTIFICACIÓN

El Grupo Filosofía y Enseñanza de la Filosofía, en el marco de la línea de investigación de pedagogía computacional, en la que se diseñan y desarrollan herramientas computacionales para el aprendizaje de habilidades mentales, ha avanzado en la representación teórica y computacional de estructuras de *lo mental* (*Vargas, 2004; Gamboa, 2011*), especialmente en el razonamiento lógico y la construcción de silogismos como estructuras sintácticas que puedan ser automatizadas en algoritmos de reconocimiento de lenguaje natural.

Este proyecto se propone desarrollar una aplicación web que ofrezca a los usuarios finales (estudiantes y profesores de filosofía) un espacio para la comprensión, el análisis y la práctica de las reglas lógicas y sintácticas de las figuras retóricas, mediante la realización de actividades como: revisión de teoría, revisión de ejemplos y realización de ejercicios. Para su realización se requiere un análisis de proyectos realizados anteriormente, en este mismo campo, con el fin de contemplar la posibilidad de reutilizar o readecuar funcionalidades existentes para, por una parte, mejorar aspectos como la lógica a nivel de negocio, las interfaces y el código spaghetti, pero por otra parte, aprovechar las concepciones lógicas y sintácticas que el Grupo ha concebido de antemano.

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVOS GENERAL

Diseñar e implementar una aplicación web para el apoyo del aprendizaje de figuras retóricas, mediante la comprensión, el análisis y la práctica de las reglas silogísticas.

2.2 OBJTIVOS ESPECÍFICOS

- Representar las reglas silogísticas en una estructura de datos que permita el almacenamiento, modificación, consulta y operación de las figuras retóricas.
- Diseñar una estrategia didáctica para garantizar el aprendizaje con elementos y actividades propios de las figuras retóricas.
- Determinar los requerimientos de la aplicación web en cuanto a: servicios a ofrecer, roles de usuarios y niveles de complejidad de actividades propuestas.
- Desarrollar los módulos de la aplicación de manera que permita:
    - Uso de la herramienta.
    - Administración de contenido de las actividades propuestas: agregar teoría, ejemplos y ejercicios.
    - Generación de reportes sobre uso de la herramienta.
    - Generación de backups para el aseguramiento de la estructura de la aplicación web.
- Instalar la herramienta en un servidor web de manera que permita su uso público con fines de evaluación por parte de los miembros del Grupo de Filosofía y Enseñanza de la Filosofía.

3. MARCO TEÓRICO

3.1 FIGURAS RETORICAS[5]

En el lenguaje literario las palabras son un fin en sí mismas, el autor selecciona el lenguaje para enriquecer la capacidad léxica de una lengua. Se entiende por "figura" en su acepción más amplia, cualquier tipo de recurso o manipulación del lenguaje con fines retóricos, antiguamente se aplicaba a la oratoria, pero al entrar ésta en decadencia pasó a la literatura y actualmente se aprecia con mayor énfasis en la publicidad.

Según la RAE las figuras retóricas o recursos estilísticos de la lengua literaria consisten en una desviación del uso normal del lenguaje con el fin de conseguir un efecto estilístico: reiteración o repetición de elementos, intensificación, embellecimiento del mensaje, etc.

Las figuras retoricas son características de la función poética del lenguaje y propios de los textos literarios tanto en prosa como en verso, más abundantes en la poesía. Pueden aparecer también en otro tipo de textos como en el lenguaje publicitario, en ciertos textos periodísticos y en la lengua coloquial.

La disciplina que se ha encargado tradicionalmente de su estudio es la Retórica, o "arte del bien decir, de embellecer la expresión de los conceptos, de dar al lenguaje escrito o hablado eficacia para deleitar, persuadir o conmover"[6]. De modo general, se pueden decir que la retórica tradicional llama figuras literarias a "cierta forma de hablar con la cual la oración se hace más agradable y persuasiva, sin respeto alguno por las reglas de la gramática". La figura es un adorno del estilo, un resultado de una voluntad de forma por parte del escritor.

3.1.1 SILOGISMO

El silogismo del griego "syllogismos" (razonamiento) es un argumento que consta de tres proposiciones, la última de las cuales se deduce necesariamente de las otras dos.[7]Es un razonamiento deductivo, las tres proposiciones se dividen en dos premisas y una conclusión, siendo la conclusión necesariamente la inferencia deductiva de las dos premisas anteriores.

[5] Las Figuras retóricas. El lenguaje literario 2. José Luis García Barrientos.
[6] Real Academia Española. (2001). Diccionario de la lengua española (22.a ed.). Consultado en http://www.rae.es
[7] Real Academia Española. (2001). Diccionario de la lengua española (22.a ed.). Consultado en http://www.rae.es

Este término fue formulado por primera vez por Aristóteles el cual consideraba la lógica como una relación de términos, los cuales son separados por juicios aristotélicos que se consideran como la unión o separación de términos, un sujeto (S) y un predicado (P), estos pueden ser tomados es su extensión particular o universal a sí mismos en su relación negativa o positiva. Estos juicios se pueden clasificar en:

Clase	**Denominación**	**Esquema**	**Expresión Ejemplo**	**Extensión de los términos**
A	Universal Afirmativo	Todo S es P	Todos los gatos son animales	S:Universal P:Particular
E	Universal Negativo	Todos los S no son P	Ningún colombiano es europeo	S:Universal P:Universal
I	Particular Afirmativo	Algún S es P	Algún americano es Argentino	S:Particular P:Particular
O	Particular Negativo	Algún S no es P	Alguno hombre no es mortal	S:Particular P:Universal

TABLA 1 CLASIFICACIÓN DE LOS JUICIOS EN SILOGISMO

3.1.1.1 FIGURA Y MODOS SILOGISTICOS[8]

Partiendo de las siguientes definiciones:

Término mayor: Es el predicado de la conclusión. La premisa en la que se encuentra se llama Premisa mayor. Se representa como P.

Término menor: Es el sujeto de la conclusión. La premisa en la que se encuentra se llama Premisa menor. Se representa como S.

Término medio: Que sirve de comparación (tertium comparationis) y no puede estar en la conclusión. Se representa como M.

Se puede dar las siguientes figuras silogísticas:

[8] A Parte Rei. Revista de Filosofía. El Silogismo Historia y Desarrollo. Silvia Carnero.

1er Figura	2da Figura	3era Figura	4ta Figura	
MP	PM	MP	PM	Premisa Mayor
SM	SM	MS	MS	Premisa Menor
SP	SP	SP	SP	Conclusión

TABLA 2 FIGURAS SILOGÍSTICAS

Estas son conocidos como modos y son las distintas combinaciones que se pueden realizar con los juicios(A, E, I, O) alcanzando hasta 64 combinaciones posibles de las cuales y luego de aplicar reglas silogísticas solo 19 modos son válidos.

3.1.1.2 REGLAS SILOGÍSTICAS

Sobre los términos

1. El silogismo sólo debe constar de tres términos
2. Ningún término debe tener suposición o significación más universal en la conclusión que en las premisas.
3. El medio debe tener suposición distributiva en alguna de las premisas
4. El medio no debe entrar en la conclusión

Sobre las premisas

1. De dos premisas afirmativas no se puede inferir una conclusión negativa
2. Si alguna de las premisas es negativa, la conclusión debe serlo también; y si alguna de aquellas es particular, debe ser particular la conclusión (1).
3. De dos premisas negativas nada se puede inferir legítimamente
4. De dos premisas particulares nada se puede inferir legítimamente

3.1.1.3 MODOS VALIDOS SILOGÍSTICOS

Estos modos es la forma que toma el silogismo de acuerdo a la cualidad de las premisas y conclusión. De las distintas combinaciones resultan solo 19 combinaciones validas que al aplicarse las reglas silogísticas generan un silogismo valido.

Los modos validos se expresan en la siguiente tabla:

	Modos validos	**Nombre modos validos**

De la primera figura	AAA,EAE,AII,EIO	*BARBARA, CELARENT, DARII, FERIO*
De la segunda Figura	EAE, AEE, EIO, AOO	*CESARE, CAMESTRES, FESTINO, BAROCO*
De la tercera figura	AAI, IAI, AII, EAO, OAO, EIO	*DARAPTI, DISAMIS, DATISI, FELAPTON, BOCARDO, FERISON*
De la cuarta figura	AAI, AEE, IAI, EAO, EIO	*BAMALIP, CAMENES, DIMATIS, FESAPO, FRESISON*

TABLA 3 MODOS VALIDOS DE LOS SILOGISMOS

3.1.2 SILOGISMOS HIPOTÉTICOS, ATLERNATIVOS Y DISYUNTIVOS[9]

El silogismo hipotético es un esquema de razonamiento que contiene tres proposiciones: la primera premisa, o premisa mayor, es una proposición hipotética; la segunda premisa o premisa menor, y la conclusión son proposiciones categóricas. Dentro de los silogismos hipotéticos, se encuentran los silogismos hipotéticos puros, que contienen dos proposiciones hipotéticas como premisas y otras como conclusión. El valor de este tipo de razonamiento es muy evidente, pues a menudo no es más fácil establecer la verdad de una proposición hipotética y la de su antecedente que la del consecuente, que puede, ser establecido indirectamente, como conclusión de la inferencia.

Ahora, los silogismos alternativos mixtos, son inferencias validas que pueden extraerse de una proposición alternativa como premisa mayor y una categórica como premisa menor. Un silogismo alternativo es válido, cuando la premisa menor niega una de las alternantes y la conclusión afirma la verdad de la otra. Esta clase de silogismo se emplea frecuentemente con el objeto de eliminar explicaciones o soluciones propuestas para diversos problemas.

Por último hablar de silogismos disyuntivos, se refiere a hablar de razonamientos donde la premisa mayor es una disyunción y la menor una proposición categórica; afirmar una disyunción significa que una, por lo menos, de las disyuntivas, es falsa. Un silogismo disyuntivo es válido si la premisa menor afirma una de las disyuntivas y la conclusión niega la otra. La forma esquemática del razonamiento

[9] Curso gratis de Filosofía Fácil. Silogismos Hipotéticos y disyuntivos. AulaFacil.com

es: no se da el caso de que A sea B y C sea D; A es B; por lo tanto, C no es D. Se dice que la inferencia está en el modus *ponens y en el modus tollens*, pues al afirmar (en la premisa menor), negamos (en la conclusión)

3.1.3 REGLAS O AXIOMAS DE VALIDEZ

Estudiando el silogismo como una forma de inferencia por medio de la cual se fija la conexión entre dos términos sobre las bases de sus relaciones con un tercer común a ambas, se puede concluir que estos axiomas expresan realmente las condiciones de validez. Los axiomas del silogismo categórico se dividen en dos clases: los que tratan de la cantidad o la distribución de los términos, y los que tratan de la calidad de las proposiciones.

Axiomas de cantidad:

1. El término medio debe estar distribuido por lo menos una vez.
2. En la conclusión no puede figurar ningún término distribuido que no lo esté en las premisas.

Axiomas de calidad:

1. De dos premisas negativas no se obtiene ninguna conclusión.
2. Si una premisa es negativa, la conclusión debe ser negativa.
3. Si ninguna de las premisas es negativa, la conclusión debe ser afirmativa.

3.1.4 TEOREMAS GENERALES DEL SILOGISMO

Teorema I: El número de términos distribuidos en la conclusión debe ser menor en por lo menos una unidad al número total de términos distribuidos en las premisas.

Teorema II: De dos premisas particulares no se desprende ninguna conclusión. Dos premisas particulares pueden ser: a) ambas negativas, b) ambas afirmativas, c) una afirmativa y una negativa.

Teorema III: Si una premisa es particular, la conclusión debe ser particular.

Teorema IV: Si la premisa mayor es una proposición particular afirmativa y la menor una proposición universal negativa, no puede haber conclusión.[10]

[10] Nagel, M.C. (s.f.). Introducción a la lógica y al método científico. II Lógica aplicada y método científico. Buenos Aires: Amorrortu editores.

3.2 RAZONAMIENTO

El razonamiento es una operación lógica mediante la cual, partiendo de uno o más juicios, se deriva la validez, la posibilidad o la falsedad de otro juicio distinto. Por lo general, los juicios en que se basa un razonamiento expresan conocimientos ya adquiridos o, por lo menos, postulados como hipótesis. Cuando la operación se realiza rigurosamente y el juicio derivado se desprende con necesidad lógica de los juicios antecedentes, el razonamiento recibe el nombre de inferencia. Los juicios que sirven como punto de partida son denominados premisas y desempeñan la función de ser las condiciones de la inferencia. El resultado que se obtiene, o sea, el juicio inferido como consecuencia, es llamado conclusión.

La inferencia permite extraer de los conocimientos ya establecidos, otro conocimiento que se encuentre implícito en las premisas o que resulte posible de acuerdo ellas. Cuando en la conclusión se llega a un conocimiento menos general que el expresado en las premisas, se habrá efectuado una inferencia deductiva. Cuando la conclusión constituye una síntesis de las premisas y, por consiguiente, un conocimiento de mayor generalidad, se habrá practicado una inferencia inductiva. Finalmente la conclusión tiene el mismo grado de generalidad o de particularidad que las premisas, entonces se habrá ejecutado una inferencia transductiva.[11]

3.2.1 RAZONAMIENTO LÓGICO

De manera axiomática se puede decir que los razonamientos pueden ser válidos (correctos) o no válidos (incorrectos). En general, se considera válido un razonamiento cuando sus premisas ofrecen soporte suficiente a su conclusión. Puede discutirse el significado de "soporte suficiente", aunque cuando se trata de un razonamiento no deductivo, el razonamiento es válido si la verdad de las premisas hace probable la verdad de la conclusión. En el caso del razonamiento deductivo, el razonamiento es válido cuando la verdad de las premisas implica necesariamente la verdad de la conclusión.

Los razonamientos no válidos que, sin embargo, parecen serlo, se denominan falacias.

El término razonamiento es el punto de separación entre el instinto y el pensamiento, el instinto es la reacción de cualquier ser vivo. Por otro lado el razonar nos hace analizar, y desarrollar un criterio propio, el razonar es a su vez la

[11] ¿Qué es el razonamiento? Ramón Ruiz Limón, El conocimiento silencioso.

separación entre un ser vivo y el hombre.

3.2.2 RAZONAMIENTO DEDUCTIVO E INDUCTIVO

Tradicionalmente, el razonamiento deductivo, se ha considerado que va de lo general a lo particular y, el inductivo, en sentido inverso. Actualmente, esta definición es pobre. Hay otros conceptos que diferencian ambos tipos de razonamiento:

- Se utiliza el concepto de validez para el razonamiento deductivo y, para el inductivo, el concepto de probabilidad.
- Un razonamiento es deductivo si la conclusión se sigue necesariamente de las premisas. Cuando se deriva necesariamente de las premisas es válido y, si es válido, significa que, siendo las premisas verdaderas, las conclusiones, también lo serán.
- El razonamiento deductivo es proposicional, de tipo silogístico, de relaciones... De este tipo de razonamiento, se pueden obtener razonamientos válidos e inválidos. Son válidos si, cuando son las premisas verdaderas, las conclusiones también lo son. De lo contrario, los razonamientos serían inválidos.
- Un argumento es válido cuando es imposible que su conclusión sea falsa, siendo sus premisas verdaderas.

Empleamos razonamientos permanentemente y sin darnos cuenta de su estructura lógica e incluso sin percatarnos de los pasos que estamos utilizando a la hora de argumentarlos. Una clasificación útil del razonamiento es analizar las dos vías que hay en relación con la explicación más rigurosa de la racionalidad: la explicación científica. Ésta presenta dos opciones o caminos diferentes. Por un lado, el razonamiento inductivo, que consiste en alcanzar una conclusión a partir de la observación, la acumulación de datos y, por último, a partir de éstos extraer una conclusión final. Por el contrario, existe el razonamiento deductivo, el cual es opuesto al inductivo. El razonamiento inductivo parte de cosas particulares y emite una generalización, tiene el propósito de estudiar las pruebas que hacen posible la medición de probabilidad de las reglas para generar argumentos inductivos sólidos, así como la medición de los argumentos mismos y el razonamiento deductivo se fundamenta en una generalidad a partir de la cual se explican los casos concretos de lo que se está analizando.

3.2.3 RAZONAMIENTO ABDUCTIVO Y ARGUMENTATIVO

El razonamiento abductivo se trata de una clase de razonamiento que comienza cuando se describe un fenómeno o un suceso y permite alcanzar una hipótesis que ofrezca una explicación a sus posibles motivos o razones a través de las premisas que se obtienen. Según el lógico, científico y filósofo inglés Charles Sanders Peirce, a quien consideran el fundador y el padre del pragmatismo y de la semiótica moderna, debemos referirnos a los razonamientos abductivos utilizando el término conjeturas. Las conjeturas intentan ser la explicación más probable o acertada a simple vista.

Por su parte, el razonamiento argumentativo está asociado a los argumentos vinculados con la producción del lenguaje. Un argumento, en este sentido, expresa en palabras el resultado de un razonamiento.

3.3 ANALOGÍA

La concepción y papel de la analogía ha variado en la historia de la filosofía. Para pensadores, como Platón o Santo Tomás, la analogía representa un tipo de razonamiento específico e Indispensable, para los empiristas, se limita a afirmar una semejanza bastante débil y sirve para la invención de hipótesis, pero debe ser eliminada en la formulación de los resultados de la investigación científica. [12]

Teóricamente la argumentación constata que el recurso a la analogía constituye una de las características de la comunicación y del razonamiento no formal. En algunos casos la analogía podrá ser eliminada, cuando la conclusión a la cual se llega se resume en una fórmula matemática, pero que muy a menudo se encuentra en el centro de una visión original, sea del universo o de las relaciones entre el hombre y la divinidad.

Para hablar de analogía, se debe distinguir entre Identidad, Igualdad y Semejanza. Hablando en sentido estricto, la Identidad es la unidad en la sustancia; la Igualdad la unidad en la cantidad y la semejanza la unidad en la cualidad.

[12] Perelman, C. (1997). El imperio retórico. Retórica y argumentación. Editorial Norma S.A.

La analogía es una semejanza en sentido estricto, es decir, una semejanza imperfecta, que no llega a la Igualdad, y por eso contiene diferencias.[13]

La analogía es una forma de predicación la cual se puede realizar de tres maneras a varios sujetos de un nombre común:

- Tomando el nombre con la misma significación en todos los casos, es decir, con significaciones perfectamente semejantes o iguales (predicación unívoca)
- Tomando el nombre con una significación completamente diversa en cada caso, con significaciones que son enteramente desemejantes o diferentes (predicación equivoca)
- Tomando el nombre con una significación en parte semejante y en parte desemejante en cada caso, o sea, con significaciones que son simplemente semejantes, pero no iguales, y por eso entrañan también desemejanzas o diferencias (predicación análoga).

La analogía es intermedia entre la univocidad y la equivocidad, y por eso participa en cierto modo de esos dos extremos. Así, conviene con la univocidad en la unidad del nombre y en la semejanza de las significaciones ligadas a ese nombre, pero difiere de ella en que no se trata de una semejanza perfecta (Igualdad), sino de una semejanza imperfecta (con desigualdad). Por otro lado, conviene con la equivocidad en la unidad del nombre y en la desemejanza de las significaciones ligadas a él, pero difiere de la misma en que no se trata de una desemejanza completa, de una diversidad total, sino de una desemejanza parcial, de una desemejanza semejante. [14]

Hasta ahora se entiende que la analogía se centra en la significación de ciertos nombres. Pero la significación de un nombre se puede tomar de dos maneras:

- **Analogía simple,** que se conoce con el nombre de analogía de atribución, se da cuando se compara un término con otro, una forma con otra, es decir se trata de semejanza de formas.
- **Analogía compuesta,** que recibe el nombre de analogía de proporcionalidad, se da cuando se compara una relación entre dos términos o formas con otra relación semejante. Se trata de la semejanza de dos o más relaciones o proporciones. Por consiguiente esta analogía exige al menos cuatro términos comparados dos a dos.

[13] La Analogía en General. Jesús García López. 2007 Servicio de publicaciones de la Universidad de Navarra. 223 paginas.
[14]La Analogía en General. Jesús García López. 2007 Servicio de publicaciones de la Universidad de Navarra. 223 paginas.

3.3.1 EL FUNDAMENTO DE LA ANALOGÍA

En sentido real y lógico. Todo nombre apunta a un concepto, en lo cual consiste su significación, pero el concepto es doble: formal y objetivo.

Formal: La misma representación mental que formamos con nuestro acto de entender.

Objetivo: Es algo real. Una esencia o una forma que existe en la realidad y en tanto que existe en ella, a la cual, le acontece el ser concebida o ser representada en un concepto formal.

3.3.2 ESTRUCTURA DE LA ANALOGÍA

Las partes que constituyen la analogía son el análogo, la trama o relación analógica y el tópico. El análogo es núcleo central de la analogía que representa el mensaje, el conocimiento ya conocido. La trama o relación analógica es el conjunto de relaciones que se establecen para comparar características semejantes de determinadas partes del análogo y del tópico, y el tópico está formado por los contenidos conceptuales, procedimentales o actitudinales desconocidos que se pretenden enseñar. [15]

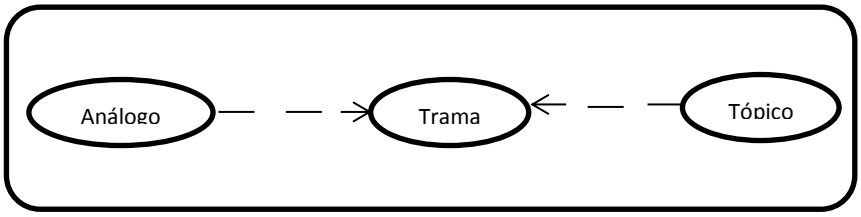


FIGURA 1 ESTRUCTURA DE UNA ANALOGÍA

La analogía consiste en establecer un conjunto de relaciones para comparar características semejantes de determinadas partes de la representación de las

[15] Estructura de las Analogías y su uso didáctico. Nicolás Elortegui, E. Moreno Jiménez, T. González González, B.M. Grupo Blas Cabrera Felipe –GITEP.

estructuras del análogo y el tópico, permitiendo mediante dicha comparación la comprensión del tópico. [16]

3.4 FALACIA

Las falacias se definen como errores de razonamiento. Se clasifican:

- Puramente lógicas o fórmales
- Semilógicas o verbales
- Materiales.

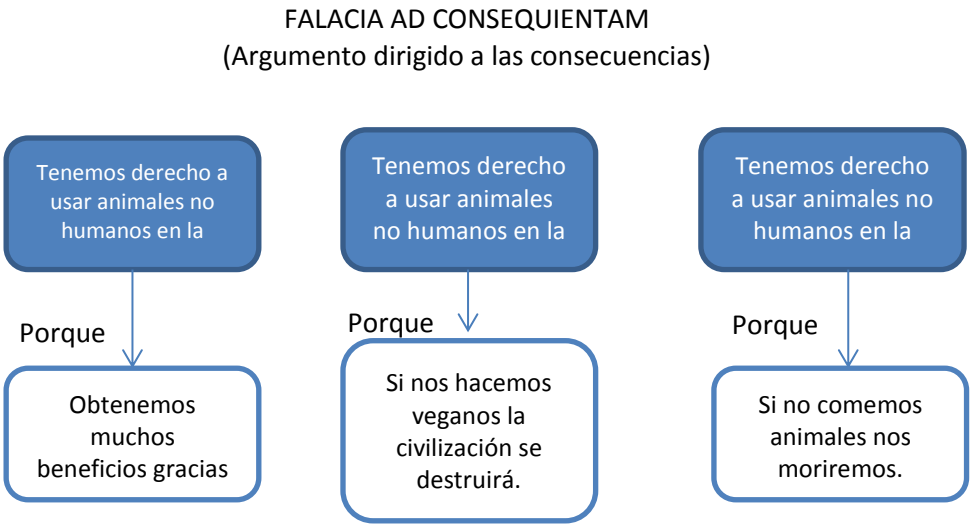


FIGURA 2 FALACIA FUENTE: FILOSOFÍA VEGANA

3.4.1 FALACIAS FORMALES

Razonamientos que no se ajustan a los tipos de inferencia valida, en relación con los diversos test o reglas que diferencian las formas validas de razonamiento de las que no lo son. Cuando se les da forma hipotética, todas las falacias resultan razonamientos que parecen de la afirmación del consecuente o de la negación del antecedente o bien que afirman una implicación o conexión lógica donde no la hay. Ejemplo de esto último es el silogismo en el que el término medio no está

[16] Las Analogías como modelo y como recurso en la enseñanza de las ciencias. Fernández González, J. Moreno Jiménez, T. González González, B.M. Grupo Blas Cabrera Felipe.

distribuido. Esto lo reduce a un razonamiento con cuatro términos cuyas premisas no brindan ninguna base o prueba para afirmar la conclusión.

3.4.2 FALACIAS SEMILÓGIAS O VERBALES

Formas aparentes validas de inferencia, que al examinarla se advierte que no lo son. Tal apariencia de validez obedece a una ambigüedad, esto es, al uso de la misma palabra o signo verbal para representar dos conceptos diferentes.

El razonamiento parece ser de la forma: *A es B*, y *B es C*, por lo tanto, *A es C*; pero en realidad, es de la forma siguiente: *A es B*, y *D es C*, por lo tanto *A es C*.

3.4.3 FALACIAS MATERIALES

Se dice que un argumento es falaz o que "contiene una falacia en alguna parte" si conduce a conclusión falsa. Aceptando que la lógica no se Identifica con todo conocimiento y no puede garantizar la verdad de todas las conclusiones. Afirmar que solo ella sea capaz de indicar qué conclusiones son, de hecho, falsas, resulta inadmisible. Una conclusión puede ser de hecho falsa, y sin embargo ser absolutamente correcto el razonamiento por el cual se la dedujo (a partir de una premisa falsa). Pero cuando deducimos una proposición de otra proposición falsa no conseguimos probar, su verdad material. Podemos designar como falacias materiales a las pruebas falsas o ilusorias. [17]

3.5 SÍMIL

Símil es un término con origen en el vocablo latino *simïlis* que hace referencia a lo semejante[18]. El concepto se utiliza para establecer una comparación entre dos cosas.

Símil es una figura retórica que consiste en la comparación expresa entre una cosa y otra, para dar una Idea eficaz de una de ellas. Al establecer la comparación por semejanza, se trasladan las características simbólicas o físicas de uno a otro. Los símiles apelan a elementos de relación como *"que", "como"* o *"cual".*

[17] Nagel, M. C. (s.f.). Introducción a la lógica y al método científico. I Lógica formal. Buenos Aires: Amorrortu editores.
[18] Negroponte, N (1995) El Mundo digital. Barcelona: Ediciones B.

FIGURA 3 SIMIL: OJOS AZULES COMO EL MAR

3.5.1 USO DE LOS SIMILES

Figura común en la literatura, en la poesía o el teatro es directa y expresa el vínculo simbólico entre las dos realidades comparadas. Algo que hace que se destaque de la analogía.

Es un recurso preferido por el lenguaje poético, muy utilizada por burla o ironía, también se puede avanzar en la discusión y dar a ver la realidad difícil de definir sin imágenes.

Pertenece a la clase de "figuras de la semejanza". Es de suma Importancia en el lenguaje, tanto en su frecuencia por su función, opera una reconciliación inesperada e innecesaria entre dos realidades, a priori desconocidos entre sí, pero con una relación de semejanza y contigüidad semántica. "Enfatizar las comparaciones y semejanzas entre las cosas, pero no cambian el significado de las palabras". Siendo así una unidad fuertemente incrustado en su sintaxis y el contexto del discurso.

En tanto, los símiles pueden ser:

- Reversibles (se disponen los dos términos sucesivamente en diferente orden).
- Graduados (inferioridad, Igualdad o superioridad). [19]

3.6 HERRAMIENTAS SOFTWARE

Algunas de las herramientas tecnológicas que utilizamos para el desarrollo de nuestro proyecto las describiremos a continuación.

[19] Florencia. (s.f.). Definición de símil. Obtenido de Definición ABC. Tu diccionario hecho fácil. Para más información invitamos a hacer parte de www.definicionabc.com

3.6.1 ENTORNOS DE DESARROLLO

3.6.1.1 NETBEANS (v.8.0.1)

Netbeans es un entorno integrado de desarrollo o IDE (Integrated Development Enviroment), en el cual podemos realizar todas las tareas asociadas a la programación tales como editar código, compilarlo, ejecutarlo y depurarlo. Netbeans es un producto libre y gratuito sin restricciones de uso.

Netbeans nos simplifica algunas tareas que en proyectos grandes pueden ser pesadas o aburridas, nos asiste en la escritura de código, aunque no nos libera de aprender el lenguaje de programación, nos ayuda en la navegación de las clases predefinidas en la plataforma.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

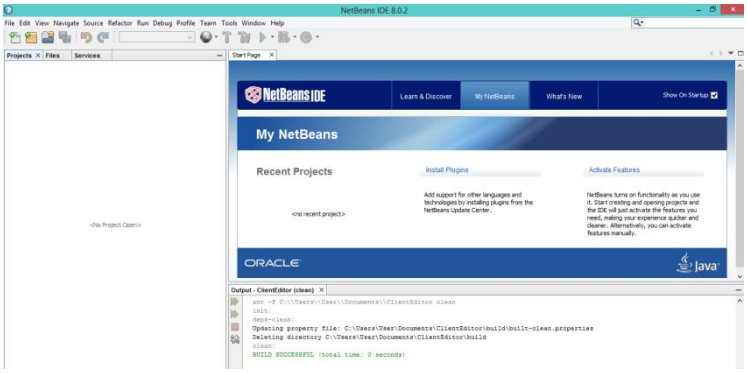


FIGURA 4 NETBEANS. TOMA CAPTURA PANTALLA ENTORNO DE DESARROLLO.

3.6.1.2 JAVA ENTERPRISE EDITION

Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. Si se necesita trabajar en una aplicación web sencilla o una transaccional, segura, interoperable y aplicación distribuida, no se desarrollaran todas las API de bajo nivel: se usa para esto la versión Enterprise Edition de Java. Al igual que Java Standard Edition (Java SE) proporciona una API para manejar colecciones, Java EE proporciona una forma estándar para manejar transacciones con Java Transaction API (JTA), mensajería con Java Message Service (JMS), o la persistencia con la API Java Persistence (JPA). Java EE es un conjunto de especificaciones destinadas a aplicaciones empresariales. Puede verse como una extensión de Java SE para facilitar el desarrollo de aplicaciones distribuidas, robustas, potentes y de alta disponibilidad.

3.6.1.2.1 ARQUITECTURA JAVA EE

Java EE es un conjunto de especificaciones implementadas por diferentes contenedores. Los contenedores son entornos de ejecución Java EE que prestan determinados servicios a los componentes que albergan tales como la gestión del ciclo de vida, la inyección de dependencia, concurrencia, y así sucesivamente. Estos componentes utilizan los contratos bien definidos para comunicarse con la infraestructura de Java EE y con los otros componentes. Necesitan ser empaquetado en una forma estándar (siguiendo una estructura de directorios definida que se puede comprimir en archivos comprimidos) antes de ser desplegado. Java EE es un súper conjunto de la plataforma Java SE, lo que significa que las API de Java SE pueden ser utilizados por cualquiera de los componentes de Java EE.

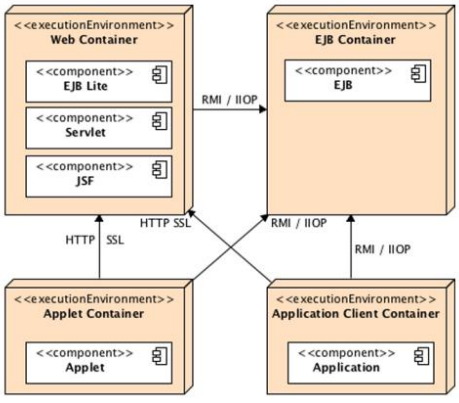


FIGURA 5 CONTENEDORES ESTANDAR JAVA EE

Muestra Las Relaciones Lógicas Entre Los Contenedores. Las Flechas Representan Los Protocolos Utilizados Por Un Contenedor Para Acceder A Otro. Por Ejemplo, Los Servlets Anfitriones Contenedor Web, Que Pueden Acceder A Los Ejb A Través De Rmi-Iiop. Tomado De Goncalves, Antonio, 2013, Beginning Java Ee, Apress, Chapter 1 Java Ee 7 At A Glance Pag 2.

3.6.1.3 JAVA PERSISTENCE API (v.2.2)

Las aplicaciones se componen de la lógica de negocio, la interacción con otros sistemas, interfaces de usuario y datos. La mayoría de los datos que manipulan nuestras aplicaciones tienen que ser almacenados en bases de datos, recuperados y analizados. Las bases de datos son importantes: almacenan los datos de negocio, actúan como un punto central entre las aplicaciones y los datos de proceso a través de disparadores o procedimientos almacenados. Datos persistentes están en todas partes, y la mayoría de las veces se utilizan las bases de datos relacionales como el motor de persistencia subyacente. Las bases de datos relacionales almacenan datos en tablas hechas de filas y columnas. Los datos se identifican mediante claves primarias, que son columnas especiales con restricciones únicas y, a veces, índices. Las relaciones entre las tablas utilizan claves ajenas y se unen a las tablas con restricciones de integridad.

Todo este vocabulario es completamente desconocido en un lenguaje orientado a objetos tales como Java. En Java, manipulamos objetos que son instancias de clases. Objetos heredan de otros, tienen referencias a las colecciones de otros objetos, y a veces apuntan a sí mismos de una manera recursiva. Tenemos clases concretas, clases abstractas, interfaces, enumeraciones, anotaciones, métodos, atributos, y así sucesivamente. Objetos encapsulan un estado y el comportamiento de una manera agradable, pero este estado sólo es accesible cuando la Máquina Virtual Java (JVM) está funcionando: si la JVM detiene o el recolector de basura limpia su contenido de memoria, los objetos desaparecen, así como su estado.

Algunos objetos tienen que ser persistentes. Por datos persistentes, ser refiere a los datos que se almacenan deliberadamente en forma permanente en medios magnéticos, memoria flash, y así sucesivamente. Un objeto que puede almacenar su estado para ser reutilizada posteriormente se dice que es persistente.

El principio de mapeo objeto-relacional (ORM) es acercar el mundo de la base de datos y objetos juntos. Se trata de delegar el acceso a bases de datos relacionales a herramientas o marcos externos, que a su vez dan una visión orientada a objetos de datos relacionales, y viceversa. Herramientas de mapeo tienen una correspondencia bidireccional entre la base de datos y objetos. Varios marcos logran esto, como Hibernate, TopLink, y Java Data Objects (JDO), pero Java Persistence API (JPA) es la tecnología preferida y es parte de Java EE 7.

3.6.1.4 ENTERPRISE JAVABEANS (v.3.1)

Para separar la capa de persistencia de la capa de presentación, para implementar la lógica de negocio, para añadir la gestión de transacciones y la seguridad, las aplicaciones necesitan una capa de negocio. En Java EE, implementamos esta capa utilizando Enterprise JavaBeans (EJB).

zEJBs son componentes del lado del servidor que encapsulan lógica de negocio y se ocupan de las operaciones y la seguridad. También tienen una pila integrada para mensajería, programación, acceso remoto, puntos finales de servicios web (SOAP y REST), inyección de dependencias, el ciclo de vida de los componentes, AOP (programación orientada a aspectos) con interceptores, y así sucesivamente. Además, los EJB se integran perfectamente con otras tecnologías Java SE y Java EE, tales como JDBC, JavaMail, JPA, Java Transaction API (JTA), Java Messaging Service (JMS), Java Authentication and Authorization Service (JAAS), Java Naming and Directory Interface (JNDI) y Remote Method Invocation (RMI). Es por eso que se utilizan para construir la capa de lógica de negocio (véase la Figura 11), se sientan en la parte superior de la base de datos, y orquesta la capa de modelo de negocio. EJBs actúan como un punto de entrada para las tecnologías de la capa de presentación como JavaServer Faces (JSF), sino también para todos los servicios externos (JMS o servicios web).

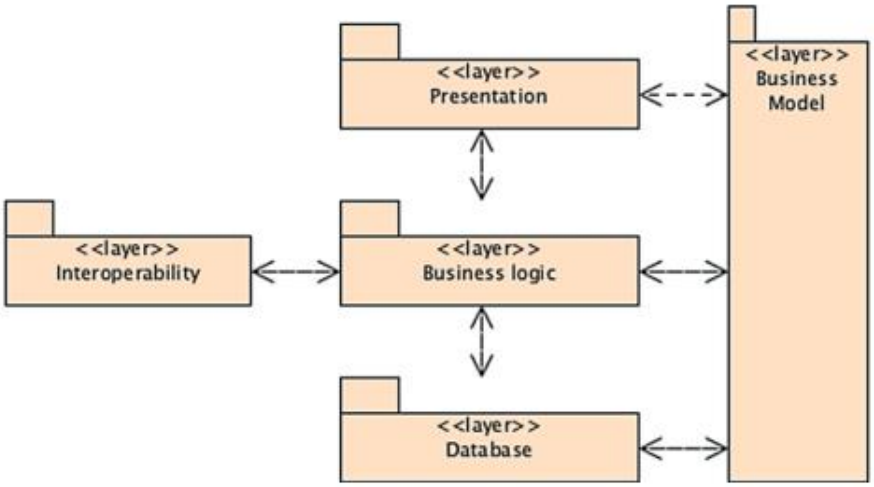


FIGURA 6 ARQUITECTURA BASADA EN CAPAS.

TOMADO DE GONCALVES, ANTONIO, 2013, BEGINNING JAVA EE, APRESS, CHAPTER 7 ENTERPRISE JAVA BEANS PAG 228.

EJB utiliza un modelo de programación muy potente que combina facilidad de uso y robustez. Hoy EJB son un muy simple modelo de desarrollo de servidor Java, reduciendo la complejidad mientras que trae la reutilización y la escalabilidad de las aplicaciones empresariales de misión crítica. Todo esto viene de anotar un solo POJO (Plain Old Java Object*)* que se desplegará en un recipiente. Un contenedor EJB es un entorno de ejecución que proporciona servicios, tales como la gestión de transacciones, control de concurrencia, puesta en común, y la autorización de seguridad. Históricamente, los servidores de aplicaciones han añadido otras características tales como clustering, balanceo de carga y conmutación por error. Desarrolladores de EJB pueden entonces concentrarse en la implementación de la lógica de negocio.

Hoy más que nunca, con la versión 3.1, EJB se puede escribir una vez y desplegar en cualquier recipiente que soporta la especificación. APIs estándar, nombres JNDI portátiles, componentes ligeros, integración CDI y la configuración de excepción permiten un fácil despliegue de EJB en código abierto, así como implementaciones comerciales. La tecnología subyacente fue creado hace más de 12 años, lo que resulta en aplicaciones EJB que se benefician de conceptos probados.

3.6.1.5 JAVASERVER FACES (v.2.2)

Si se desea mostrar gráficamente la información procedente de lo visto anteriormente, hay que crear una interfaz de usuario. Puede ser de varios tipos: las aplicaciones de escritorio, aplicaciones web que se ejecutan en un navegador, o aplicaciones móviles que se ejecutan en un dispositivo portátil que muestra una interfaz gráfica e interactúa con el usuario final.

Hoy vivimos en un mundo en Internet dependiente. Con el procesamiento de nuestra parte final transaccional miles de solicitudes, y la comunicación con sistemas heterogéneos a través de servicios web, necesitamos una capa de presentación para interactuar con los usuarios finales, de preferencia uno que se ejecuta en un navegador. Los navegadores están en todas partes, las interfaces de usuario son más ricas, más dinámicas y más fáciles de usar que antes. Aplicaciones dinámicas de Internet están ganando en popularidad a medida que los usuarios esperan más de su experiencia de navegación. Ellos necesitan consultar los catálogos de libros y CDs en línea, pero también quieren tener acceso a correo electrónico y documentos, recibir un correo electrónico, o tiene partes de su página del navegador actualizan de forma selectiva cuando se produce un evento de servidor. Añadir a que la Web 2.0 filosofía del ser humano pueden compartir todo tipo de información con grupos de amigos e interactuar con los demás, y el resultado es interfaces web que se están volviendo más y más complejas para desarrollar. JavaServer Faces (JSF, o simplemente Faces) fue creada para facilitar la creación de interfaces gráficas.

3.6.1.6 FRAMEWORK PRIMEFACES

PrimeFaces es un conjunto de componentes JSF de código abierto con varias extensiones que facilitan la creación de las aplicaciones web.

- Rico conjunto de componentes (Editor HTML, Dialogo, Autocompletar, Cuadros y muchos más).
- Construido en Ajax basado en APIs estándar Ajax JSF 2.0.
- Ligero, cero configuración y sin dependencias requeridas.
- Soporte de empuje a través Atmosphere Framework.
- Kit de interfaz de usuario móvil para crear aplicaciones web para móviles.
- Amplia documentación.
- Grande, vibrante y activa comunidad de usuarios.

- Desarrollado con “pasión” de desarrolladores de aplicaciones para desarrolladores de aplicaciones.
- Componentes en PrimeFaces se desarrollan con un principio de diseño que establece que "Un buen componente de interfaz de usuario debe ocultar la complejidad, pero mantener la flexibilidad" mientras lo hace.

3.6.1.7 BOOTSTRAP

Bootstrap es un framework front-end libre para que el desarrollo web sea más fácil y rápido, bootstrap incluye plantillas HTML y CSS de diseño basado en la tipografía, formas, botones, tablas, navegación, modelos, imágenes en carrusel y muchas otras, así como plugins de Java Script opcionales. Bootstrap también da la posibilidad de crear fácilmente responsive designs.

Ventajas de usar Bootstrap:

- Fácil de usar: Cualquier persona con conocimientos básicos de HTML y CSS puede empezar a utilizar bootstrap.
- Características Responsive: Bootstrap’s responsive CSS se ajusta a teléfonos, tabletas y ordenadores.
- Compatibilidad del navegador: Bootstrap es compatible con todos los navegadores modernos (Chrome, Firefox, Internet Explorer, Safari y Opera).

3.7 ARQUITECTURA MVC (MODELO VISTA CONTROLADOR).

Si alguna vez se ha desarrollado algún sitio web mediante PHP sin utilizar ningún tipo de framework, seguro se tendría un archivo PHP por cada página HTML del sitio. Además todos esos archivos PHP contendrían la misma estructura: inicialización y configuración global, lógica de negocio relacionada con la página solicitada, obtención de registros de la base de datos y por último el código PHP que se emplea para generar la página.

También es factible usar un modelo de plantillas para separar código PHP y el HTML. Se puede utilizar capaz de abstracción de base de datos para separar la lógica de negocio y la interacción con el modelo de datos. A pesar de estas mejoras, siempre se va a encontrar con una gran cantidad de código que va a ser muy difícil de comprender. Programar una aplicación de estas maneras seguro es de un costo de tiempo más corto, pero modificarla y añadirle nuevas

características se convierte en una pesadilla sobre todo porque nadie más sabe cómo está construida y como funciona.

La solución más utilizable actualmente para organizar el código es el patrón de diseño MVC. El patrón de diseño MVC organiza el código en base a su función. Este patrón separa el código en tres capas.

- La capa modelo define la lógica de negocio (la base de datos pertenece a esta capa) es la representación de la información con la cual el sistema opera, por lo tanto gestiona los accesos a dicha información, consultas, actualizaciones, implementa también los privilegios de acceso que se hayan descrito anteriormente en la aplicación.
- La vista es lo que utilizan los usuarios para poder interactuar de manera visual con la aplicación (los gestores de plantilla están en esta capa).
- El controlador es un bloque de código el cual responde a eventos y tiene como función hacer llamadas a la capa modelo y luego se los pasa a la vista para mostrarlos al usuario final.

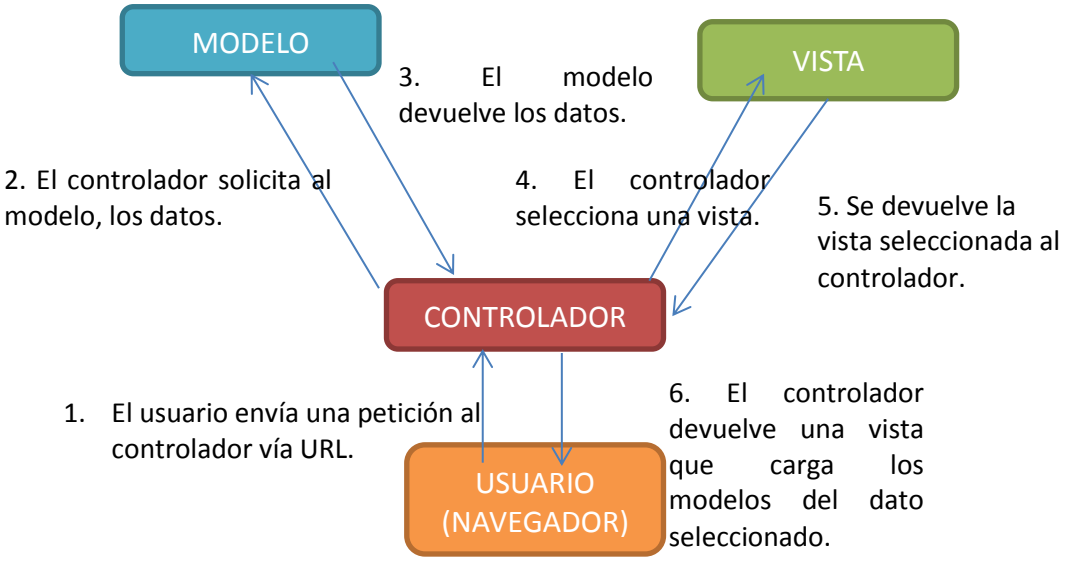


FIGURA 7 MODELO VISTA CONTROLADOR

3.8 SISTEMA DE CONTROL DE VERSIONES –GIT-

Git es un sistema de versiones distribuido gratis y de código abierto[20], diseñado para manejar proyectos grandes y pequeños con velocidad y eficiencia. Algunas de las ventajas de usar Git son: Fusiona archivos no borrara otros archivos subidos por alguien anteriormente, es distribuido permite que muchas personas puedan contribuir al mismo tiempo en algún proyecto en común, guarda una línea de tiempo, es un sistema distribuido, es open source y es capaz de manejar proyectos de gran tamaño.

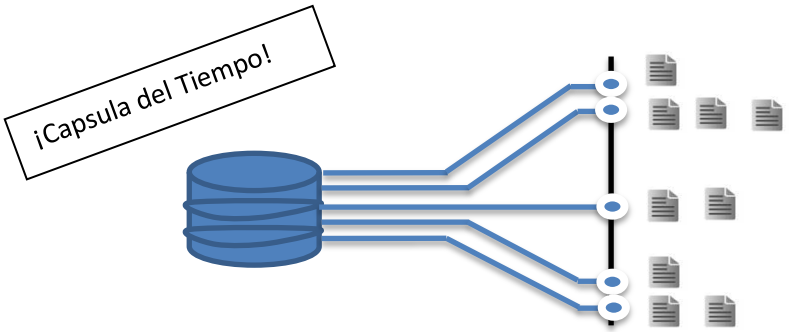


FIGURA 8 COMO FUNCIONA GIT

FIGURA 8. COMO FUNCIONA GIT.

Git funci (8)) que representa el repositorio donde se encuentran nuestros archivos en la nube, tenemos un servidor remoto al que subimos los archivos y ese servidor va guardando una línea del tiempo de todos los comits o modificaciones que le hacemos al proyecto luego podemos ir de atrás hacia adelante en nuestra capsula del tiempo para ir manipulando las modificaciones diferentes que se le han hecho al programa.

3.8.1 GITHUB

Github es una red social donde podemos almacenar nuestros proyectos en línea es un servicio de hosting que tiene tanto versiones pagas como gratuitas en donde enorme cantidad de empresas importantes hacen uso de él.

[20] ProGit, Everything you need to know about git, The Expert's Voice, Second Edition, Scott Chacon and Ben Straud.

La diferencia entre Git y Github es que mientras Git es el sistema de control de versiones, Github es el aparte donde se almacenan nuestros proyectos y estos proyectos almacenados podemos modificarlos a través de Git.

GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome). Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.[21]

[21] GitHub. Definición tomada de Wikipedia la enciclopedia libre. Para conocer más ingresar a: https://es.wikipedia.org/wiki/GitHub

4. ESTADO DEL ARTE

Para tener en cuenta en el estado del arte hemos hecho la búsqueda de diferentes softwares que nos muestran sus diferentes funciones con respecto a la deducción lógica, los cuales nos dan la posibilidad de evaluar predicados lógicos, deducción natural de predicados lógicos y proposiciones, revisar tablas de verdad, enseñarnos lógica deductiva, etc.

Se han definido unos campos de indagación tales como software de enseñanza de razonamiento lógico, que tan importantes son estos programas a la hora de enseñar. Buscamos encontrar en estos programas datos relevantes, ampliar el conocimiento sobre lo estudiado con referencia a figuras retoricas, estudiar la evolución de lo que estamos investigando, generar nuevas ideas y posturas sobre lo que estamos investigando, queremos identificar vacíos o necesidades que hagan falta a la hora del desarrollo de un software de educación.

Algunas de las herramientas que encontramos para el aprendizaje de lógica en primer orden y figuras retoricas se encuentran:

Silogística Virtual.[22] Un programa desarrollado por medio de JavaScript para la realización de ejercicios relacionados con las figuras y los modos del silogismo el cual revisa las 256 formas diferentes de silogismos.

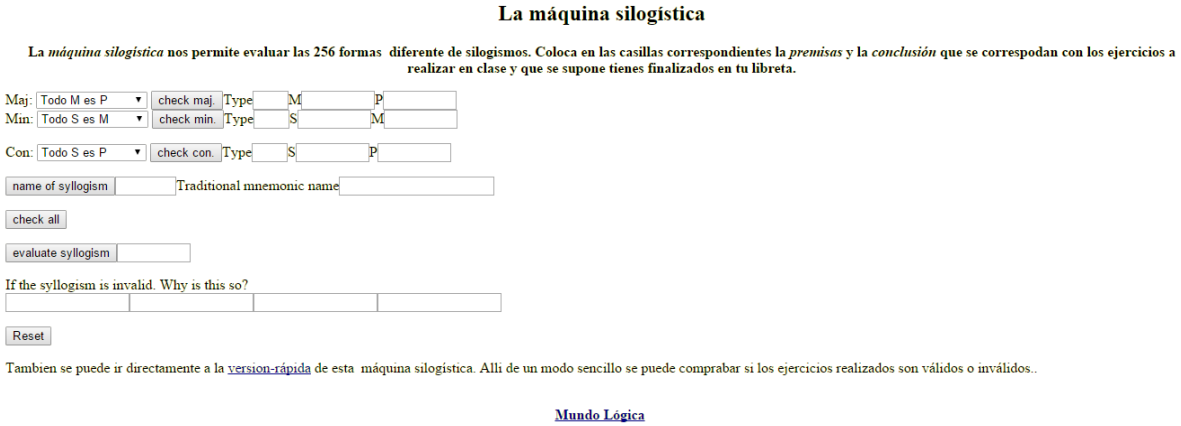


FIGURA 9 SILOGÍSTICA VIRTUAL.

[22]La máquina para revisar los silogismos se puede encontrar en http://www.paginasobrefilosofia.com/html/maquisil.html#Maquina

APROS.[23] El objetivo de este software es ayudar a los estudiantes a descubrir la estructura de los argumentos lógicos, la posibilidad de practicar deducciones naturales y predicados lógicos.

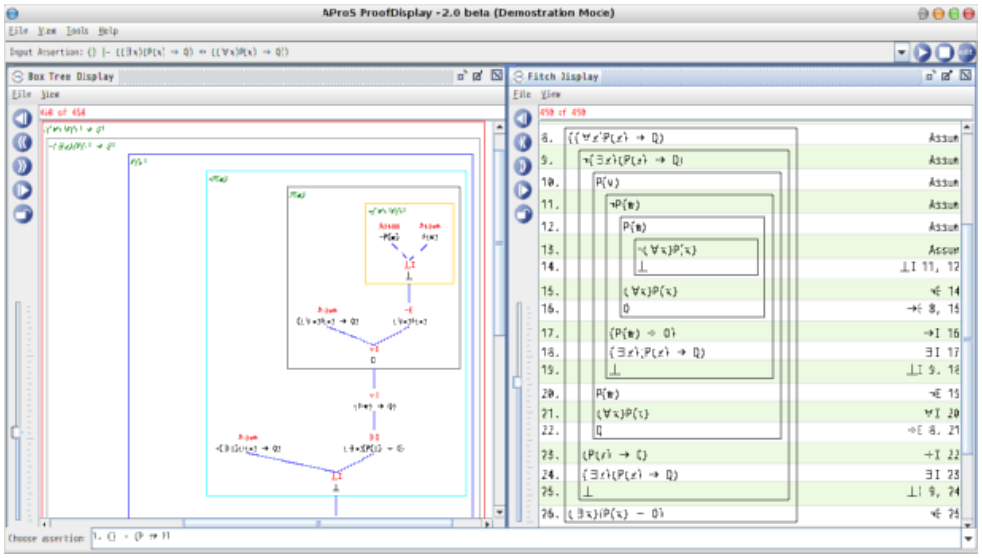


FIGURA 10 APROS

Asistente para la Deducción Natural (ADN).[24] Es una herramienta didáctica que asiste al alumno que se inicia en la técnica de Deducción Natural a elaborar sus propias deducciones.

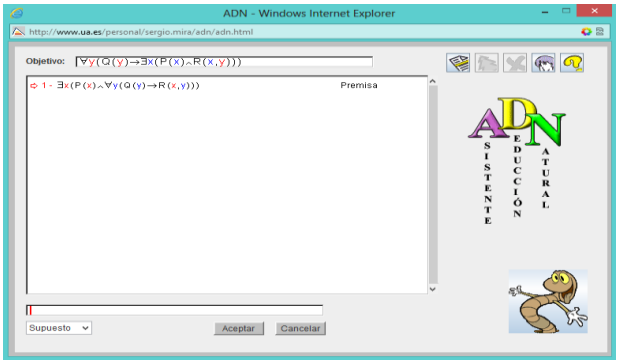


FIGURA 11 ADN

[23] Se puede encontrar en http://www.phil.cmu.edu/projects/apros/

[24] Se encuentra disponible en http://www.dccia.ua.es/logica/ADN/index.htm

Proyecto A.F.R.I (Aprendiendo a Filosofar con Recursos Informáticos).[25] Es un recurso informático que se centra en la enseñanza de la filosofía y se centra en ocho filósofos: Platón, Tomás de Aquino, Descartes, Kant, Marx, Nietzsche, Ortega y M. Zambrano. Los contenidos sobre cada autor se van presentando a través de diferentes secciones: Menú del autor, Cuestión, Vida, Contexto, Esquemas, Desarrollo, Textos, Ejercicios, Detalle y Complementos.

FIGURA 12 PROYECTO A.F.R.I

LogicforFun.[26] Con esta plataforma web se busca enseñar lógica a través de rompecabezas y acertijos.

Los principales problemas a resolver que se han notado son el hecho de que son aplicaciones software muy básicas algunas difíciles de manejar, siendo herramientas con un nivel muy básico de estética y orden del cual a veces no se encuentra sentido alguno de orientación. Algunas de estos software son desarrollados en otro idioma (ingles) lo cual puede hacer más complicado su comprensión, también se nota que en algunos de estos nos obligan a bajar el programa para poder correrlo en nuestros computadores, hasta encontramos una que no es de gratis acceso. Lo que se busca principalmente es hacer una versión mejorada una plataforma de trabajo en la web con fácil acceso público con una presentación ordenada de los temas y sin la necesidad de tener que obligar al usuario a instalar alguna herramienta para poder hacer uso de este.

[25] Se puede encontrar en la dirección web http://www.proyectoafri.es/
[26] Se puede encontrar en http://logic4fun.rsise.anu.edu.au/

5. ESTRUCTURA LOGICA

5.1 REQUISITOS[27]

5.1.1 DEFINICIONES Y ACRÓNIMOS

- Definiciones

Módulo	Se refiere a la ventana de la figura retórica que se esté trabajando puede ser silogismo, falacia, analogía, etc. Y compone toda su teoría, ejemplos y ejercicios.
Rol, Tipo de Usuario	Es el tipo de usuario que se encuentra en nuestro sistema, puede ser profesor, estudiante o administrador y cada uno tiene una función diferente en el software.

- Acrónimos

AAFR	Aplicación web para el aprendiza de figuras retoricas
ERS	Especificación de requisitos software
RFXX	El estándar seguido para la especificación del identificador de cada requisito funcional será de la siguiente manera: • R = Requisito • F = Funcional • XX = Secuencia de tres dígitos que servirá para la enumeración de cada requisito.
RNFXX	El estándar seguido para la especificación del identificador de cada requisito no funcional será de la siguiente manera: • R = Requisito • NF = No Funcional

[27] IEEE Recommended Practices for Software Requierements especification ANSI/IEEE 830 1998.

	• XX = Secuencia de tres dígitos que servirá para la enumeración de cada requisito.

En este apartado se presentan los requisitos funcionales que deberán ser satisfechos por el sistema. Todos los requisitos aquí expuestos son esenciales, es decir, no sería aceptable un sistema que no satisfaga alguno de los requisitos expuestos. Los requisitos se han especificado de manera que sea fácil comprobar si el sistema los ofrece o no y si los ofrece de manera adecuada.

La definición de requerimientos enfocado en la construcción de la herramienta de software AAFR es una de las principales tareas desarrolladas en este proyecto ya que se a definido una condición o capacidad que debe exhibir o poseer este sistema para satisfacer la necesidades del grupo de investigación filosofía y enseñanza de filosofía. Estos requerimientos estas separados en funcionales (qué debe hacer un sistema) y no funcionales (cómo debe ser el sistema.)

La metodología utilizada para esta identificación fue: entrevistas virtuales con usuarios finales, análisis de software en cuanto a diseño y funcionalidades existentes.

5.1.2 REQUERIMIENTOS FUNCIONALES

Los requisitos funcionales definidos para la realización de la herramienta de software se definieron según los módulos requeridos por el grupo de investigación:

ID	**RF01**	**Fuente**	Usuarios		
Nombre		Inicio de Sesión			
Complejidad		Alta		**Prioridad**	Alta
Usuarios		Administrador, estudiante y profesor			
Descripción					
La aplicación Web debe permitir el inicio de sesión únicamente con usuario y contraseña, esta debe ser capaz de identificar el rol que contiene.					

TABLA 4 REQUERIMIENTO FUNCIONAL 01

ID	**RF02**	**Fuente**			
Nombre		Módulos			
Complejidad		Baja		**Prioridad**	Alta
Usuarios		Administrador, estudiante y profesor			

Descripción
La aplicación Web deberá estar conformada por módulos de teoría, ejemplos y ejercicios para cada una de las figuras retoricas.

TABLA 5 REQUERIMIENTO FUNCIONAL 02

ID	RF03	Fuente	Usuarios		
Nombre	Generador de premisas				
Complejidad	Alta		**Prioridad**	Alta	
Usuarios	Profesor, estudiante				
Descripción					
La aplicación Web deberá contener un generador de premisas, en el cual el usuario será el encargado de diligenciar los términos. Nota: Se da por entendido que el rol profesor será el único encargado de manipular esta función y este conoce a su perfección los términos correctos a incluir.					

TABLA 6 REQUERIMIENTO FUNCIONAL 03

ID	RF04	Fuente	Usuarios		
Nombre	Estructura de un silogismo				
Complejidad	Alta		**Prioridad**	Alta	
Usuarios					
Descripción					
La aplicación web debe ser capaz de reconocer la estructura correcta de un silogismo, al igual que sus distintos modos y clases.					

TABLA 7 REQUERIMIENTO FUNCIONAL 04

ID	RF05	Fuente	Usuarios		
Nombre	Ejercicios silogismo				
Complejidad	Media		**Prioridad**	Media	
Usuarios	Profesor, estudiantes				
Descripción					
El módulo de ejercicio de silogismos estará dado por diferentes niveles de complejidad de ejercicios y el contenido de estos será generado automáticamente.					

TABLA 8 REQUERIMIENTO FUNCIONAL 05

ID	RF06	Fuente	Usuarios		
Nombre	Validar ejercicios				
Complejidad	Media		**Prioridad**	Baja	
Usuarios	Estudiante				
Descripción					
El módulo de ejercicio de silogismos validara la respuesta dada por el rol estudiante sin confirmación del rol profesor.					

TABLA 9 REQUERIMIENTO FUNCIONAL 06

ID	RF07	Fuente	Usuarios
Nombre	Ejercicios símil, analogía, razonamiento y falacia		
Complejidad	Media	**Prioridad**	Media
Usuarios	Profesor		
Descripción			
El módulo de ejercicios de las figuras retoricas: Símil, Analogía, Razonamiento, y Falacia estará dado por ejercicios propuesto por el profesor y este será el encargado de validar la respuesta dada por el rol estudiante.			

TABLA 10 REQUERIMIENTO FUNCIONAL 07

ID	RF08	Fuente	Usuarios
Nombre	Ejemplos símil, analogía, razonamiento y falacia		
Complejidad	Media	**Prioridad**	Media
Usuarios	Profesor		
Descripción			
El módulo de ejemplos de las figuras retoricas: Símil, Analogía, Silogismos, Razonamiento, y Falacia estará dado por ejemplos propuestos por el profesor.			

TABLA 11 REQUERIMIENTO FUNCIONAL 08

ID	RF09	Fuente	Usuarios
Nombre	Teoría símil, analogía, razonamiento y falacia		
Complejidad	Media	**Prioridad**	Media
Usuarios	Profesor		
Descripción			
El módulo de teoría de las figuras retoricas: Símil, Analogía, Silogismos, Razonamiento, y Falacia estará dado por el contenido e información adicionada por el rol profesor.			

TABLA 12 REQUERIMIENTO FUNCIONAL 09

ID	RF10	Fuente	Usuarios
Nombre	Vínculos de Interés		
Complejidad	Baja	**Prioridad**	Baja
Usuarios	Profesor		
Descripción			
El módulo Teoría permitirá adicionar vínculos de interés por el rol Profesor.			

TABLA 13 REQUERIMIENTO FUNCIONAL 10

ID	RF11	Fuente	Usuarios
Nombre	Funcionalidades		
Complejidad	Alta	**Prioridad**	Alta

Usuarios	Profesor, estudiante y administrador
Descripción	
La aplicación Web ofrecerá funcionalidades diferentes a cada uno de los roles especificados (Administrador, Estudiante, Profesor).	

TABLA 14 REQUERIMIENTO FUNCIONAL 11

ID	**RF12**	**Fuente**	Usuarios		
Nombre		Reportes			
Complejidad		Media		**Prioridad**	Baja
Usuarios		Estudiante			
Descripción					
La aplicación Web contara con un módulo de reportes que indicar el progreso de cada uno de los estudiantes por los diferentes módulos navegados.					

TABLA 15 REQUERIMIENTO FUNCIONAL 12

5.1.2 REQUERIMIENTOS NO FUNCIONALES

ID	**RNF01**	**Fuente**			
Nombre		Desarrollo web			
Complejidad		Alta		**Prioridad**	Alta
Usuarios					
Descripción					
La aplicación Web debe ser desarrollada en Java Platform, Enterprise Edition o Java EE.					

TABLA 16 REQUERIMIENTO NO FUNCIONAL 01

ID	**RNF02**	**Fuente**			
Nombre		Aplicación cargada en el servidor grupo de investigación			
Complejidad		Alta		**Prioridad**	Alta
Usuarios					
Descripción					
La aplicación Web debe ser desplegada en el servidor de aplicación dado por el grupo de investigación de filosofía y enseñanza de la filosofía.					

TABLA 17 REQUERIMIENTO NO FUNCIONAL 02

ID	**RNF03**	**Fuente**			
Nombre		Base de datos			
Complejidad		Media		**Prioridad**	Media
Usuarios					
Descripción					

La DB de la aplicación WEB debe ser MYSQL.

TABLA 18 REQUERIMIENTO NO FUNCIONAL 03

En el siguiente apartado se encuentra la explicación lógica de la herramienta de software para el aprendizaje de las figuras retoricas donde se encontrara como se realizó la construcción de cada uno de los módulos (Silogismos, Razonamiento, Símil, Analogía y Falacias), así como también la técnicas que se utilizaron para crear el algoritmo generación de silogismos y así mismo de la creación de conclusiones. Esta estructura se dividirá en 3 fases:

5.2 MODELO

El cual representa la información con la cual el sistema opera. Este gestiona todos los accesos a la información, como consultas, actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio).

5.2.1 MODELO DE PROGRAMACIÓN

5.2.1.1 PACKAGE: COM.FIGURASRETORICAS.ENTITY

Con ayuda del asistente para Clases de Entidad de Base de Datos, se creó una clase de entidad para cada una de las tablas de base de datos con diferentes anotaciones de consultas de los atributos de las tablas representando columnas y relaciones e interpretando claves externas. En esta actividad se utilizó la tecnología JPA que permitió gestionar a la aplicación datos entre los objetos creados en java y la DB relacional que se tiene.

Las clases de entidad que creamos forman una representación, basada en Java, de la base de datos Figuras Retoricas. Mientras cada clase de entidad representa a una tabla de la base de datos, las instancias de estas clases corresponderán a registros que pueden ser salvados (persistentes) en la base de datos.

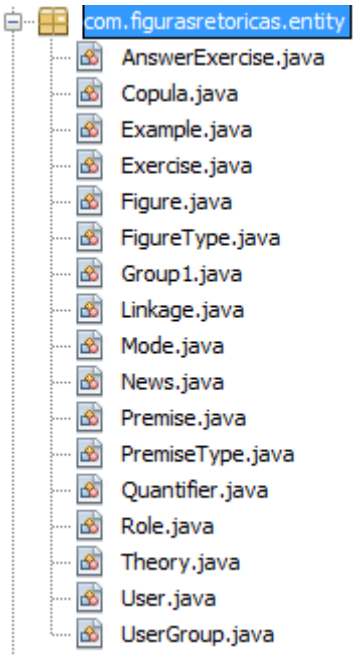


FIGURA 13 PACKAGE: COM.FIGURASRETORICAS

Las consultas estáticas predefinidas por ejemplo findAll obtiene todos los registros de news (así para cada una de las tablas mapeadas) de la base de datos. De esta forma obtenemos una lista de objetos News como un atributo que puede ser referenciado mediante la cadena "News". Además la anotación de @NamedQuery contiene diferentes elementos en los cuales name, y query son los dos requeridos.

```
@Entity
@Table(name = "news", catalog = "figuras_retoricas", schema = "")
@XmlRootElement
@NamedQueries({
  @NamedQuery(name = "News.findAll", query = "SELECT n FROM News n"),
  @NamedQuery(name = "News.findById", query = "SELECT n FROM News n WHERE n.id = :id"),
  @NamedQuery(name = "News.findByPublicationDate", query = "SELECT n FROM News n WHERE n.publicationDate = :publi
  @NamedQuery(name = "News.findByTitle", query = "SELECT n FROM News n WHERE n.title = :title")})
```

FIGURA 14 FIGURA NAMEDQUERY-NEWS

```
@NamedQueries({
  @NamedQuery(name = "Premise.findAll", query = "SELECT p FROM Premise p"),
  @NamedQuery(name = "Premise.findById", query = "SELECT p FROM Premise p WHERE p.id = :id"),
  @NamedQuery(name = "Premise.findBySubject", query = "SELECT p FROM Premise p WHERE p.subject = :subject"),
  @NamedQuery(name = "Premise.findBySubjectPlural", query = "SELECT p FROM Premise p WHERE p.subjectPlural = :subjectPlu
  @NamedQuery(name = "Premise.findByPredicate", query = "SELECT p FROM Premise p WHERE p.predicate = :predicate"),
  @NamedQuery(name = "Premise.findByPredicatePlural", query = "SELECT p FROM Premise p WHERE p.predicatePlural = :predic
  @NamedQuery(name = "Premise.findByArticle", query = "SELECT p FROM Premise p WHERE p.article = :article"),
  @NamedQuery(name = "Premise.findBySingular", query = "SELECT p FROM Premise p WHERE p.singular = :singular"),
  @NamedQuery(name = "Premise.findByQuantity", query = "SELECT p FROM Premise p WHERE p.quantity = :quantity"),
  @NamedQuery(name = "Premise.findByQuality", query = "SELECT p FROM Premise p WHERE p.quality = :quality"),
  @NamedQuery(name = "Premise.findByConclusion", query = "SELECT p FROM Premise p WHERE p.conclusion = :conclusion"),
  @NamedQuery(name = "Premise.findByMediumTerm", query = "SELECT p FROM Premise p WHERE p.mediumTerm = :mediumTerm"),
  @NamedQuery(name = "Premise.findByGender", query = "SELECT p FROM Premise p WHERE p.gender = :gender")})
```

FIGURA 15 FIGURA NAMEDQUERY-PREMISA.

5.2.1.2 PACKAGE COM.FIGURASREOTIRCAS.FACADE

La fachada de sesión es el modelo de diseño que pretende resolver problemas comunes en el entorno de una aplicación modular. Es una arquitectura del lado del servidor que nos permite el desarrollo simple y rápido de aplicaciones distribuidas, transaccionales, seguras y portables.

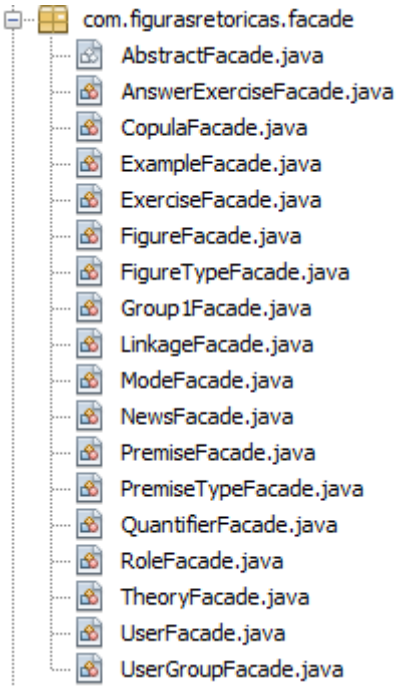


FIGURA 16 PACKAGE COM.FIGURASRETORICAS.

El Asistente de Beans de Sesión para Clases de Entidad crea una fachada EJB de sesión para cada clase de entidad, con métodos básicos de acceso. Se puede visualizar que se crea una clase llamada AbstractFacade esto debido a que el IDE de netbeans, extrae todo el código común de todas las demás clases generadas, y estas a su vez heredan a esta clase generalizada.

Igualmente la clase gestiona las relaciones entre los objetos de negocio, así como el ciclo de vida de éstos, creándolos, localizándolos, modificándolos y borrándolos según lo requiera el flujo de trabajo.

```
public void create(T entity) {
  getEntityManager().persist(entity);
}

public void edit(T entity) {
  getEntityManager().merge(entity);
}

public void remove(T entity) {
  getEntityManager().remove(getEntityManager().merge(entity));
}

public T find(Object id) {
  return getEntityManager().find(entityClass, id);
}
```

FIGURA 17 GESTION RELACIONES ENTRE OBJETOS DE NEGOCIO

Estos beans de sesión de Java Enterprise son invocados por el cliente para llevar a cabo una tarea específica del Negocio. Si abrimos cualquiera de las fachadas de sesión, por ejemplo TheoryFacade, veremos que todas instancian un EntityManager usando la anotación @PersistanceContext.

```
@Stateless
public class TheoryFacade extends AbstractFacade<Theory> {
  @PersistenceContext(unitName = "FigurasRetoricasPU")
  private EntityManager em;

  @Override
  protected EntityManager getEntityManager() {
    return em;
  }

  public TheoryFacade() {
    super(Theory.class);
  }

}
```

FIGURA 18 FACHADA DE SESIÓN, THEORYFACADE

Esta anotación es utilizada para inyectar una EntityManager gestionada por contenedor en una clase. Dicho de otra forma, confiaremos en el contenedor EJB de GlassFish para abrir EntityManager's como y cuando sea necesario. El nombre de la unidad específica la unidad de persistencia FigurarasRetoricas PU que definimos en el fichero persistence.xml.

El EntityManager es un componente integrado de la API de persistencia de Java, que se encarga de implementar acciones de persistencia en la base de datos.

Las implementaciones de la especificación de JPA se llaman proveedores de persistencia. En este caso, la implementación del proveedor de persistencia de referencia es EclipseLink.

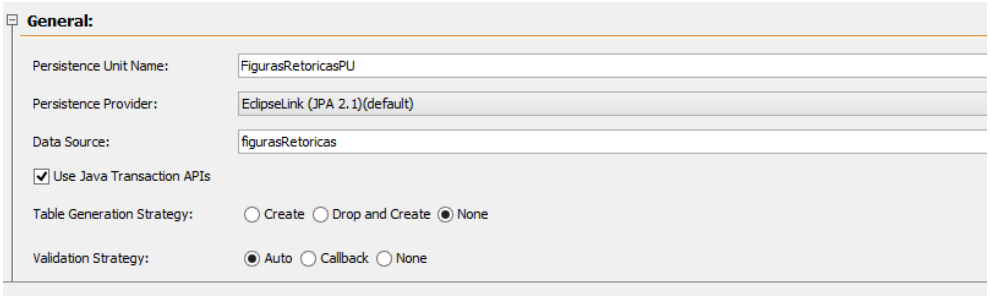


FIGURA 19 IMPLEMENTACIÓN JPA

5.3 MODELO ENTIDAD-RELACIÓN

En este apartado se encuentra el modelo entidad-relación de la base de datos que se utiliza en la aplicación propuesta. Este modelo cuenta con 17 tablas y distinto tipos de relaciones entre ellas.

ANEXO 1. MODELO ENTIDAD-RELACIÓN.

5.4 VISTA

Presenta la interfaz con la que el usuario va interactuar constantemente.

El login es la primer vista la cual el usuario se autentica frente a la aplicación; existen 3 tipos de usuarios el cual la aplicación automáticamente se encarga de

visualizar la interfaz según corresponda el rol. En caso del que el usuario no esté incluido en la aplicación, esta no permitirá ninguna navegación.

FIGURA 20 LOGIN

Existen 3 tipos de páginas principales según el rol de cada uno de los usuarios:

Página Principal Administrador, la cual cuenta con los módulos de administración que se encarga de todo lo relacionado con la gestión de grupos, y roles de estudiantes y profesores. Estos módulos permiten llevar una administración de usuarios y grupos (crear, editar, ver, eliminar) para el uso de la aplicación. Cuando se termine la actividad realizada el usuario con rol administrador puede realizar su logout dirigiéndose al botón superior derechos y así terminar la sesión.

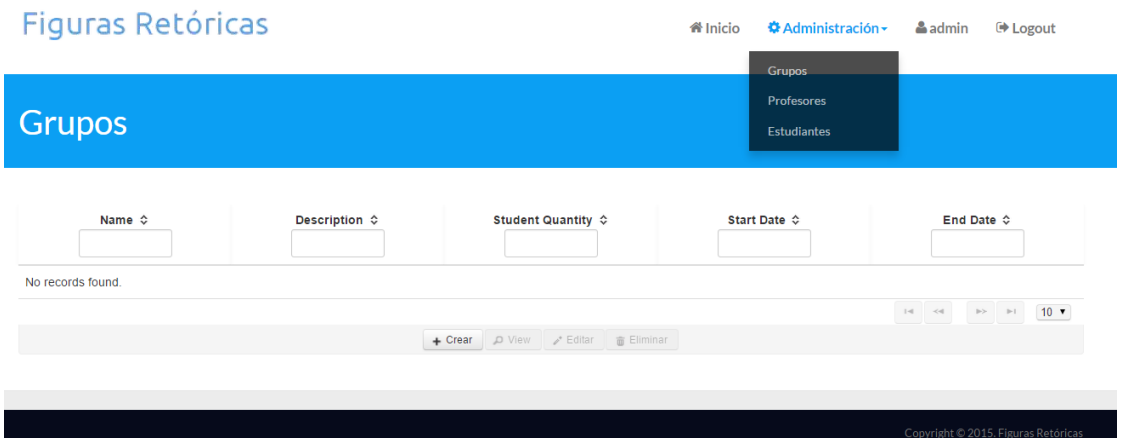


FIGURA 21 VISTA GRUPOS

A continuación se presenta los diferentes formularios con el que el administrador podrá agregar usuarios en los diferentes roles y grupos los cuales se visualizaran al momentos de accionar el botón crear en los diferente módulos de administración.

Nuevo Grupo:

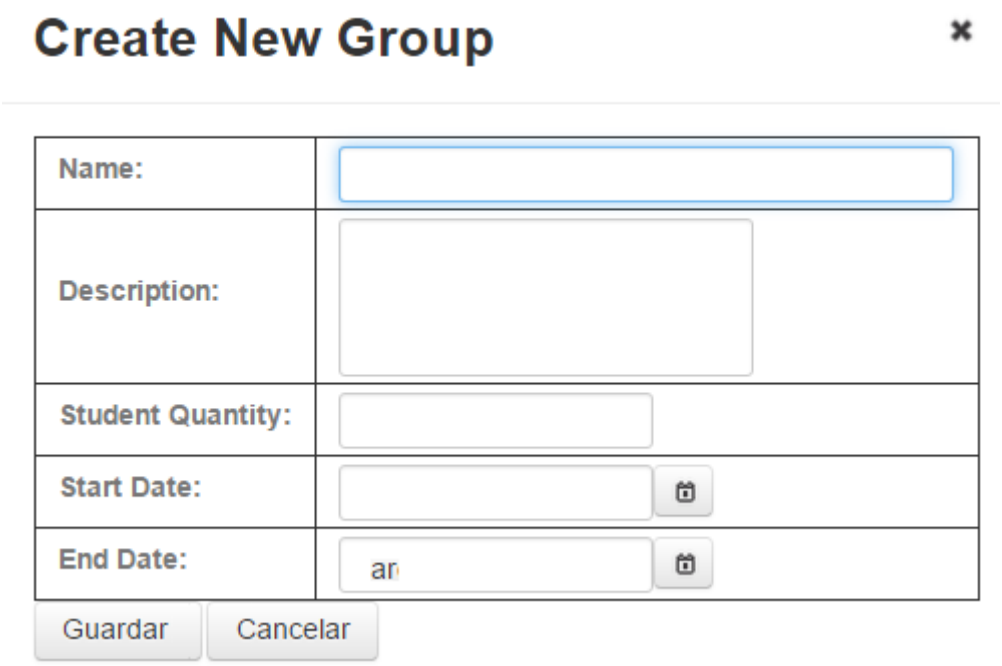


FIGURA 22 VISTA CREAR NUEVO GRUPO

Nuevo Usuario Rol Profesor:

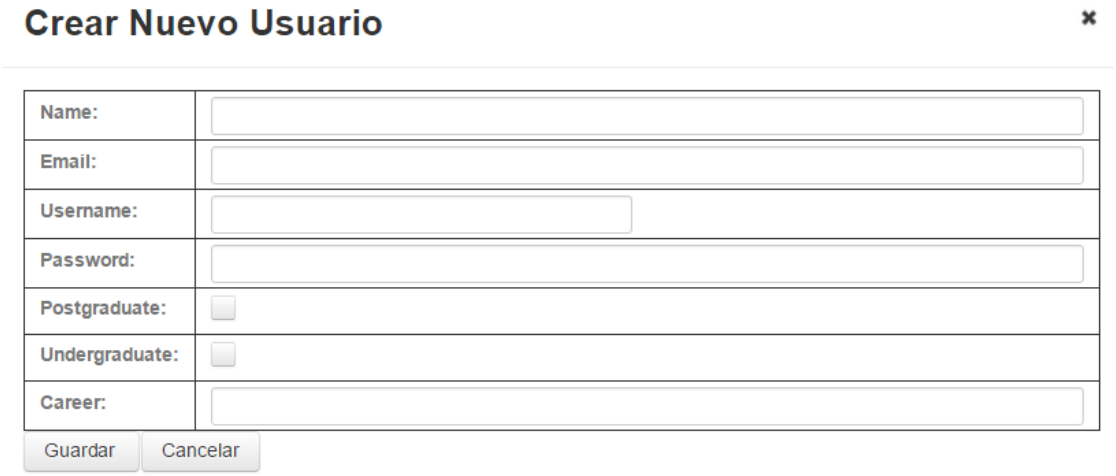


FIGURA 23 NUEVO USUARIO ROL PROFESOR. VISTA

Nuevo Usuario Rol Estudiante (Característica especial asignación de grupo):

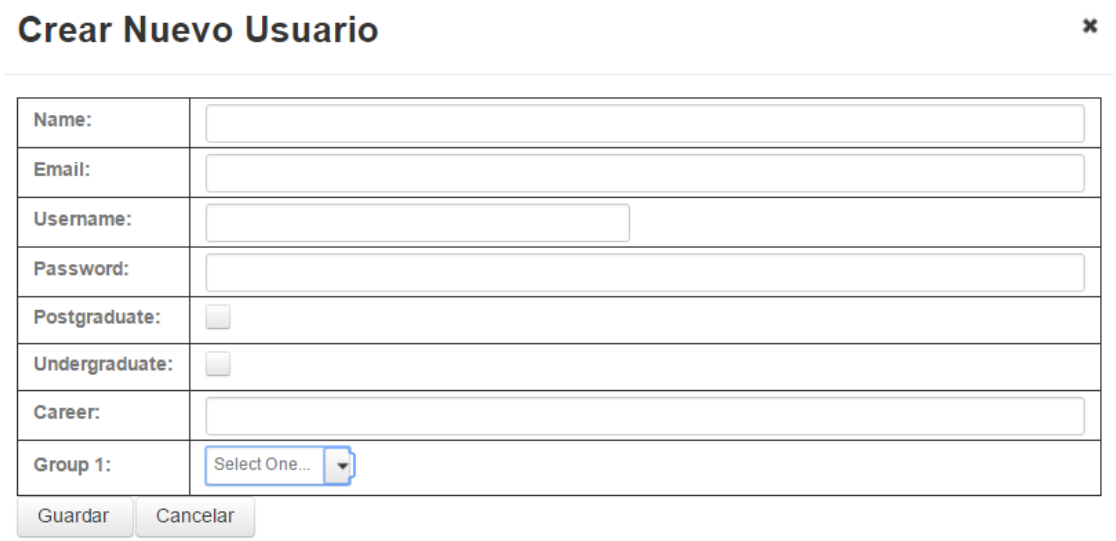


FIGURA 24 NUEVO USUARIO ROL ESTUDIANTE

Página Principal Profesor:

Esta página cuenta con la gestión de cada una de las figuras retoricas en cuantos los ejercicios y ejemplos, en esta vista los usuarios con rol profesor podrán agregar la teoría requeridas, así como links de interés; Esta teoría cuenta con un el formato básico que se le pueda dar y en cualquier momento puede ser editar o borrada por el rol profesor. Además cuenta con el módulo de reporte de estudiantes del grupo y generador de silogismos que nos ayudara a crear premisas y conclusiones de esta figura retórica.

FIGURA 25 PAGINA PRINCIPAL PROFESOR

Creación de ejemplos:

La creación de ejemplos está sujeta al rol profesor y esta podrá crearlos para cualquier figura retórica requerida, además no se tiene límite de caracteres del ejemplos, así como también se podrá incluir un número ilimitado de ejemplos los cuales podrán ser visualizados por el rol estudiante más adelante.

FIGURA 26 VISTA CREACION DE EJEMPLOS

Creación de ejercicios:

Las figuras retoricas se podrán crear diversos tipos de ejercicios relacionados con estas, estas son guardadas en base de datos y podrán ser modificadas en cualquier momento utilizando las diferentes opciones disponibles. Además estos ejercicios pueden ser validados correcta e incorrectamente por parte del rol profesor, y ser visualizados posteriormente por el estudiante.

FIGURA 27 VISTA CREACION DE EJERCICIOS

Generador de silogismos:

El módulo de generador de silogismos está dado por una serie de parámetros (sujeto y predicado) el cual el rol profesor debe diligenciar, y el sistema se encarga de completar la información faltante, igualmente generar los 4 tipos de premisas posibles(A, E, I, O) y ofrece la opción de guardar estas. Así mismo en cualquier momento se podrá generar las conclusiones esto con el fin de generar la premisa faltante, y poder realizar los diferentes niveles de ejercicios validados automáticamente de la figura retórica silogismos.

FIGURA 28 GENERADOR DE SILOGISMOS

Página Principal Estudiante:

Esta página se muestra todos los módulos de figuras retoricas que se encuentras disponibles, cada una ellas cuenta con una serie de opciones que se podrán escoger para realizar el proceso de aprendizaje de cada una de estas figuras.

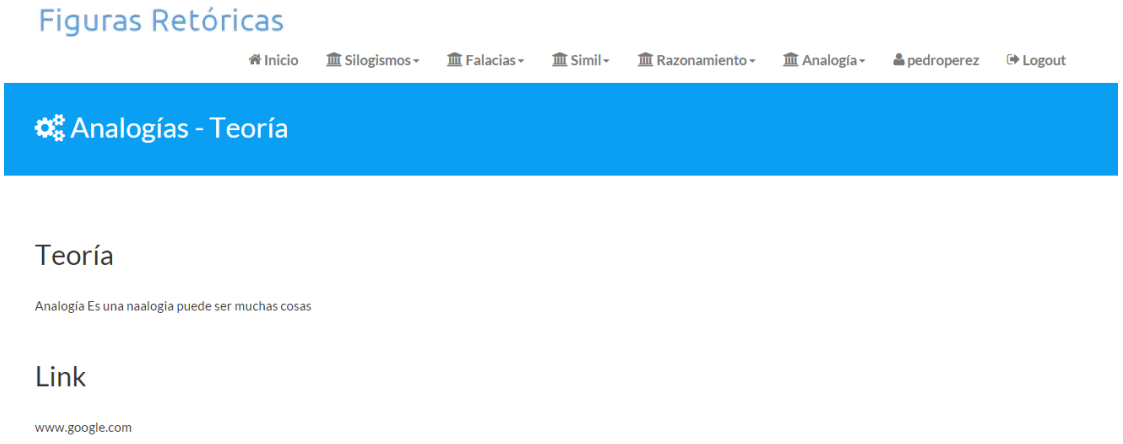


FIGURA 29 PÁGINA PRINCIPAL ESTUDIANTE VISTA.

Se podrá verificar la teoría dispuesta para cada figura y consular los diferentes links de interés que el rol profesor ha referenciado para poder ampliar el conocimiento teórico de una figura retórica especifica.

Así mismos se podrá verificar los ejemplos disponibles para cada una de las figuras retoricas, en caso de no contener ejemplos, el sistema le notificara al rol estudiante de este status. Estos ejemplos podrán ser consultados en cualquier momento y se tiene una variación asegurando que no se repitan al menos que se hayan consultado todos los ejemplos disponibles.

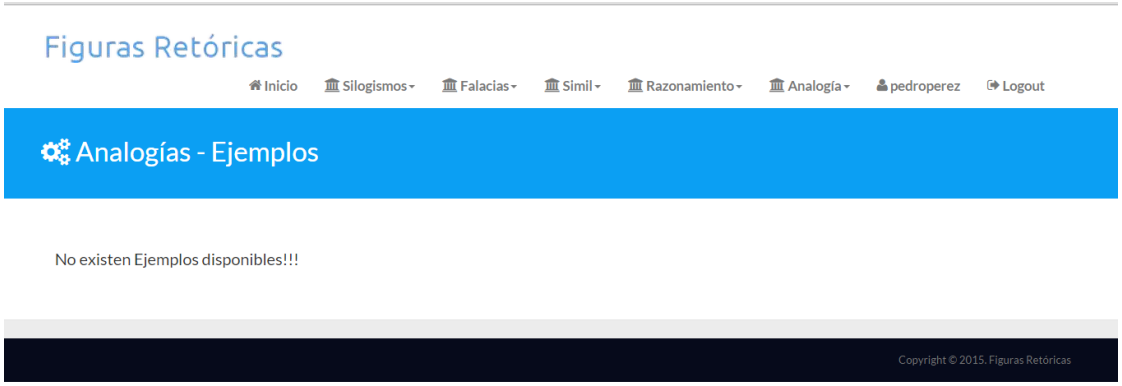


FIGURA 30 VERIFICAR EJEMPLOS VISTA

Los ejercicios de las figuras retoricas están dados por enunciados los cuales el rol estudiante deberá responder y estos son posteriormente validados por el rol profesor el cual ofrece una calificación correcta o incorrecta, y esta es visualizada por el estudiante con su respectivo comentario si aplica. En cualquier momento el estudiante podrá dejar la sesión accionando el botón logout presente en la esquina superior derecha de la aplicación.

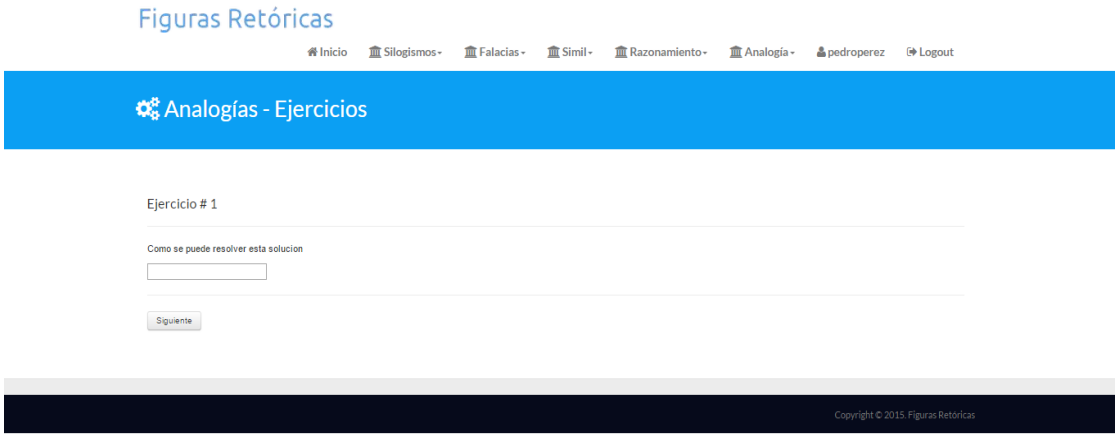


FIGURA 31 VISTA EJERCICIOS

5.4 CONTROLADOR

5.4.1 PACKAGE COM.FIGURASRETORICAS.CONTROLLER:

El cuál es el encargado de responder a eventos e invocar peticiones. Es donde se encuentra la lógica del negocio, este como tal el que contiene toda la comunicación desde la parte del click del usuario (Vista) hasta la operación y manejo de la DB (Modelo) en la realización de alguna operación.

La mayoría de las clases del paquetes de controladores están dadas como entidades desde las base de datos, lo que quiere decir que para cada una de las tablas en nuestro modelo tenemos un controlador, igualmente nuestro IDE crea una clase llamada AbstractController, la cual contiene (al igual que el modelo) los métodos generales utilizados.

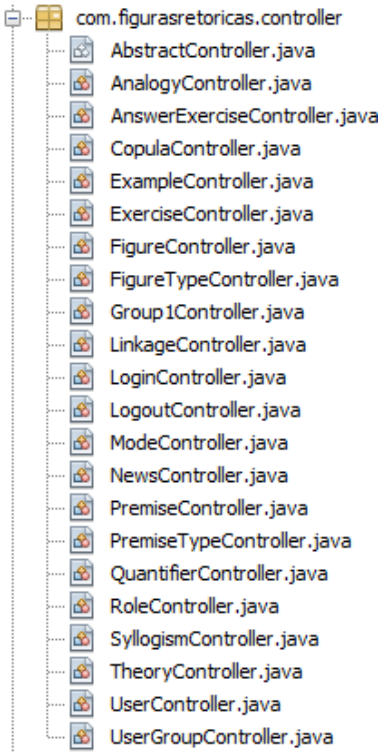


FIGURA 32 PAQUETE CONTROLADORES

Algunas clases fueron creadas independientemente, y no son visualizadas dentro del modelo como:

5.4.1.1 LOGINCONTROLLER.JAVA

Esta clase se encarga de realizar el inicio de sesión de los diferentes roles de usuario, validando el match entre el username y password entregado con los alojados en el modelos datos, y así producir el inicio de sesión, en caso contrario el controller arroja expcecion de usuarios y password incorrecto.

5.4.1.2 LOGOUTCONTROLLER.JAVA

Esta clase función como fin de sesión de los roles de usuarios al dar submit al botón referenciado como “logout” este genera un evento de salida hacia el /index de la aplicación

5.4.1.3 SYLLOGISMCONTROLLER.JAVA

Esta clase contiene toda la lógica de negocios del Generador de silogismos (Generar.xhtml) que a su vez contiene el generador conclusiones el cual es visualizado por el rol profesor dentro de la aplicación. Esta clase contiene diferentes etapas que al ser invocada desde la vista realiza sus respectivas operaciones como:

1. Agregar Plural a los diferentes sujetos y predicados, tomando en cuenta la última letra de estos y agregando “s” o “es” a la palabra digitada en la vista.

2. Identificación referenciado en la vista, dado que la premisa se debe armar con un sentido femenino o masculino, dependiendo de la elección de los usuarios al momento de interactuar con la vista.

3. Generación de los 4 tipos de premisas (‘A, E, I, O’) de los silogismos, con su respectivo cuantificador y copula. El cual por medio de la persistencia de datos y la vista, se construyen y se visualizan al usuario para la validación por parte de este.

4. Almacén de premisas en el modelo de datos posterior submit de botón guardar

5. Generación de conclusiones tomando en cuenta las premisas previamente generadas y guardadas en el modelo de datos, además este método valida las 4 figuras y 19 modos posibles de generación modos validos de silogismos, guardando todo modo correcto, copula, cuantificador, término medio, identificador de figura, nombre de figura en el modelo de datos para su posterior utilización en los niveles de ejercicios de silogismos. Ver: Diagrama de flujo, diagrama de silogismos.

5.5 IMPLEMENTACION WEB

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados.[28]

En este punto es necesario aclarar lo siguiente: mientras que comúnmente se utiliza la palabra servidor para referirnos a una computadora con un software servidor instalado, en estricto rigor un servidor es el software que permite la realización de las funciones descritas.

Para la implementación de la aplicación sobre el servidor web se realiza la generación del archivo .WAR por medio del IDE, y se realiza diferentes casos de pruebas para la comprobación de funcionalidades. **Anexo 2.**

FIGURA 34 IMPLEMENTACION DE LA APLICACIÓN WEB

Esta aplicación es desplegada en un servidor de aplicación glashfish 4.1 integrado con bases de datos MySql sobre el servidor del grupo de investigación de filosofía y enseñanza respondiendo al url: http://filosofiayensenanza.uis.edu.co:8080/figurasretoricas/login

FIGURA 33 PÁGINA PRINCIPAL. FILOSOFIA Y ENSEÑANZA

5.6 DIAGRAMAS DE FLUJOS

5.6.1 DIAGRAMA SILOGISMOS. ANEXO 3.

5.6.2 DIAGRAMA GENERACIÓN DE PREMISAS. ANEXO 4.

5.7.3 DIAGRAMA FLUJO CONVERTIR SUJETO Y PREDICADO SINGULARES EN PLURALES. ANEXO 5.

6. DISEÑO COMPUTACIONAL

En el siguiente apartado mostraremos los actores que guardan una relación con el sistema y que demandan una funcionalidad, los casos de uso que los actores realizan en la aplicación y que nos proporcionaran uno o más escenarios que nos indican como debería interactuar el sistema con el usuario o con otro sistema, la navegación por ventanas de nuestro software y los prototipos de la interfaz del usuario final.

6.1 DEFINICION DE ACTORES

A continuación mostraremos los actores involucrados en nuestra aplicación junto con su descripción, tipo de actor y también los casos de usos en los que estos intervienen.

ACTORE(ES)	
Actor	Administrador
Casos de Uso	Gestionar(Agregar, eliminar, o editar) usuarios Gestionar (Agregar, eliminar, o editar) teoría Gestionar(Agregar, eliminar, o editar) grupos
Tipo	Primario
Descripción	Es el súper usuario de la aplicación, es el encargado de realizar operaciones de creación, edición o eliminación de las diferentes opciones disponibles para el manejo de teoría, grupos, estudiantes, profesores.
Actor	Estudiante
Casos de Uso	Visualizar ejemplos Practicar ejercicios Visualizar teoría
Tipo	Primario
Descripción	Previamente registrado por el administrador en la herramienta web, podrá tener accesos a todo el contenido expuesto en la aplicación al igual que interactuar activamente con este.
Actor	Profesor
Casos de Uso	Gestionar ejemplos de la figura retórica especifica Gestionar ejercicios de la figura retórica especifica. Generar premisas y conclusiones de los silogismos Validar ejercicios

	Generar reportes
Tipo	Primario
Descripción	Previamente registrado por el administrador en la herramienta web, podrá tener acceso a la gestión de ejercicios y ejemplos, además de agregar vínculos ala teoría, realizar reportes y validación de ejercicios realizados por el rol estudiante.

TABLA 19 DEFINICIÓN DE ACTORES

6.2 CASOS DE USO

A continuación se encuentran los casos de uso de la aplicación con su respectiva descripción y su respectiva documentación (Actores participantes, flujo de eventos, precondiciones, sub-flujos y pos-condición).

ID	01		
Nombre	Agregar Usuarios		
Necesidad	Agregar un nuevo usuario para la utilización del software	**ID/ Necesidad**	0001
Actores:	Administrador		
Descripción:	El administrador agregara un usuario nuevo especificando su respectivo rol.		
Disparadores:	Agregar un nuevo usuario al software.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	El usuario especificado es agregado.		
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere agregar un usuario. 2. Se dirige al módulo administración y escoge el rol usuario 3. El administrador marca la opción agregar 4. Se diligencia la información requerida. 5. Se guarda la información diligenciada		
Flujos alternativos:	1. Si el usuario a agregar ya está registrado, el 2. Sistema arrojara un mensaje de “usuario registrado” y no permitirá el doble de registro de		

	estos.
Requerimientos especiales	RF01

TABLA 20 CASO DE USO AGREGAR USUARIOS

ID	02		
Nombre	Editar Usuario		
Necesidad	Editar los datos de un usuario nuevo y guardarlos.	**ID/ Necesidad**	0002
Actores:	Administrador		
Descripción:	El usuario administrador edita un usuario ya guardado modificando los respectivos campos.		
Disparadores:	Editar un usuario al software.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	La información edita del usuario en específico es guardada.		
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere editar un usuario. 2. Se dirige al módulo administración y escoge el rol usuario a editar 3. El administrador marca la opción editar 4. Se escoge el usuario a editar 5. El sistemas arroja la información actual del usuario especificado 6. Se edita la información requerida. 7. Se guarda la información		
Flujos alternativos:	Sistema arrojara un mensaje de error en caso de que la información suministrada no esté completa.		
Requerimientos especiales	RF01		

TABLA 21 CASO DE USO EDITAR USUARIO.

ID	03		
Nombre	Eliminar Usuarios		
Necesidad	Eliminar los datos de un usuario ya existente y guardar.	**ID/ Necesidad**	0003
Actores:	Administrador		
Descripción:	El usuario administrador elimina un usuario específico ya existente.		
Disparadores:	Eliminar un usuario al software.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	El usuario especificado es eliminado		
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere eliminar un usuario. 2. Se dirige al módulo administración y escoge el rol de usuario a eliminar 3. El administrador marca la opción eliminar 4. Se busca el usuario a eliminar 5. Se escoge el usuario y se confirma la opción de eliminación		
Flujos alternativos:	El usuario visualiza mensaje generado por el sistema de la eliminación satisfactoria del usuario		
Requerimientos especiales	RF01		

TABLA 22 CASO DE USO ELIMINAR USUARIO.

ID	04		
Nombre	Agregar Grupos		
Necesidad	Agregar grupos nuevos para uso de estudiantes y profesores.	**ID/ Necesidad**	0004
Actores:	Administrador		
Descripción:	El usuario administrador agrega un nuevo grupo con profesor y estudiantes.		

Disparadores:	Agregar un grupo al software.
Pre-condiciones:	El usuario debe haber iniciado sesión. Deben existir usuarios en rol profesor y estudiante.
Post-condiciones:	El sistema arroja un mensaje de confirmación de usuarios agregado satisfactoriamente.
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere agregar un grupo. 2. Se dirige al módulo administración y selecciona la opción grupo 3. El administrador marca la opción agregar 4. Se diligencia la información requerida. 5. Se guarda la información diligenciada.
Flujos alternativos:	Si el grupo a agregar ya está registrado, el sistema arrojara un mensaje de “grupo registrado” y no permitirá el doble de registro de estos.
Requerimientos especiales	RF01

TABLA 23 CASO DE USO AGREGAR GRUPOS.

ID	05		
Nombre	Editar Grupos		
Necesidad	Modificar un grupo ya existente.	**ID/ Necesidad**	0005
Actores:	Administrador		
Descripción:	El usuario administrador edita un nuevo grupo con profesor y estudiantes.		
Disparadores:	Edita un grupo al software.		
Pre-condiciones:	El usuario debe haber iniciado sesión. Deben existir grupos.		
Post-condiciones:	El sistema arroja un mensaje de confirmación información guardada satisfactoriamente.		
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere editar un grupo.		

	2. Se dirige al módulo administración y selecciona la opción grupo 3. El administrador marca la opción editar 4. Se edita la información y usuarios del grupo 5. Se guarda la información diligenciada.
Requerimientos especiales	RF01

TABLA 24 CASO DE USO EDITAR GRUPOS.

ID	06		
Nombre	Eliminar Grupos		
Necesidad	Eliminar un grupo ya existente.	**ID/ Necesidad**	0006
Actores:	Administrador		
Descripción:	El usuario administrador elimina un grupo existente con profesor y estudiantes.		
Disparadores:	Eliminar un grupo al software.		
Pre-condiciones:	El usuario debe haber iniciado sesión. Deben existir grupos.		
Post-condiciones:	El sistema arroja un mensaje de confirmación de grupo eliminado satisfactoriamente		
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere eliminar un grupo. 2. Se dirige al módulo administración y selecciona la opción grupo 3. El administrador marca la opción eliminar 4. Se selección el grupo requerido. 5. Se selecciona eliminar grupo 6. Se guarda la información diligenciada.		
Requerimientos especiales	RF01		

TABLA 25 CASO DE USO ELIMINAR GRUPOS

ID	07

Nombre	Gestionar Teoría		
Necesidad	Modificar la teoría de un módulo determinado.	**ID/ Necesidad**	0007
Actores:	Administrador		
Descripción:	El usuario administrador gestiona la teoría de un módulo existente cambiando o eliminando.		
Disparadores:	Administrador requiere gestionar la teoría de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	La teoría de la figura retórica específica es gestionada.		
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere gestionar la teoría de las diferentes figuras retoricas. 2. Se dirige al módulo administración y escoge la opción teoría 3. El administrador podrá escoger entre la opción editar, o agregar teoría de la figura retórica especificada 4. El administrador realizara la operación requerida y guardara la información suministrada 5. El sistema guardara la información requerida		
Flujos alternativos:	1. El sistema ofrecerá la opción vista previa de teoría 2. Sistema no permitirá que la teoría sea borrada en su totalidad		
Requerimientos especiales	RF01		

TABLA 26 CASO DE USO GESTIONAR TEORÍA

ID	08		
Nombre	Gestionar Ejemplos de las figuras retoricas		
Necesidad	Modificar los ejemplos de un módulo determinado.	**ID/ Necesidad**	0008
Actores:	Profesor		
Descripción:	El usuario profesor gestiona los ejemplos de un módulo		

	existente cambiando, eliminando o agregando.
Disparadores:	Profesor requiere gestionar los ejemplos de una figura retórica.
Pre-condiciones:	El usuario debe haber iniciado sesión.
Post-condiciones:	EL profesor deberá seleccionar cualquiera de las opciones desplegadas en la opción de gestionar ejemplos.
Flujo Normal:	1. Este caso de uso se inicia cuando el administrador requiere gestionar la teoría de las diferentes figuras retoricas. 2. Se dirige al módulo administración y escoge la opción teoría 3. El administrador podrá escoger entre la opción editar, o agregar teoría de la figura retórica especificada 4. El administrador realizara la operación requerida y guardara la información suministrada 5. El sistema guardara la información requerida
Flujos alternativos:	Si no existe figuras retoricas el sistemas mostrara un mensaje de figura retórica no disponible
Requerimientos especiales	RF01

TABLA 27 CASO DE USO GESTIONAR EJEMPLOS DE LAS FIGURAS RETORICAS

ID	09		
Nombre	Visualizar ejemplos		
Necesidad	Revisar los ejemplos de un módulo determinado.	**ID/ Necesidad**	0009
Actores:	Profesor		
Descripción:	El usuario profesor podrá visualizar los ejemplos de un módulo existente.		
Disparadores:	Profesor requiere visualizar los ejemplos de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión		

	Deben existir ejemplos adicionados
Post-condiciones:	Los ejemplos serán visualizados por el profesor.
Flujo Normal:	1. Este caso de uso se inicia cuando el Profesor requiere visualizar los ejemplos existentes 2. El profesor se dirige a la opción de visualizar ejemplos de cualquiera de las figuras retoricas indicadas en el caso de uso de gestión de ejemplos 3. El profesor podrá visualizar los ejemplos situados en el pool de ejemplos 4. Para finalizar el profesor debe salir de la vista visualización de ejemplos
Flujos alternativos:	1. El profesor podrá editar o eliminar los ejemplos visualizados. 2. El sistema arrojara un mensaje de confirmación al momentos de que un ejemplos sea eliminado o editado 3. El sistemas arrojara mensaje de error en caso en el que el pool de ejemplos este vacío
Requerimientos especiales	RF01

TABLA 28 CASO DE USO VISUALIZAR EJEMPLOS

ID	10		
Nombre	Agregar ejemplos.		
Necesidad	El profesor desea agregar más ejemplos a los módulos.	**ID/ Necesidad**	0010
Actores:	Profesor		
Descripción:	El usuario profesor podrá agregar nuevos ejemplos de un módulo existente.		
Disparadores:	Profesor requiere agregar ejemplos de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión		
Post-condiciones:	El sistema arroja un mensaje de confirmación de adición del ejemplo requerido.		

Flujo Normal:	1. Este caso de uso se inicia cuando el Profesor requiere adicionar ejemplos 2. El profesor se dirige a la opción de adicionar ejemplos de cualquiera de las figuras retoricas indicadas en el caso de uso de gestión de ejemplos 3. El profesor podrá diligenciar el formulario de adición de ejemplos 4. El profesor debe guardar los ejemplos adicionados. 5. Para finalizar el profesor debe salir de la vista adicionar ejemplos
Flujos alternativos:	El sistemas arrojara mensaje de error en caso que el ejemplo no este correctamente diligenciado
Requerimientos especiales	RF01

ID	11		
Nombre	Gestionar Ejercicios de las figuras retoricas		
Necesidad	El profesor desea gestionar los ejercicios de los módulos.	**ID/ Necesidad**	0011
Actores:	Profesor		
Descripción:	El usuario profesor podrá los ejercicios de un módulo existente.		
Disparadores:	Profesor requiere gestionar ejercicios de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión		
Post-condiciones:	El profesor deberá seleccionar cualquiera de las opciones desplegadas en gestión de ejercicios: visualizar, agregar, validar ejercicios y generar premisas.		
Flujo Normal:	1. Este caso de uso se inicia cuando el Profesor requiere gestionar ejercicios de las figuras retoricas 2. Se dirige al módulo de gestión y escoge la figura retórica a gestionar 3. El profesor selecciona la opción ejercicios 4. El caso de uso finaliza cuando el profesor visualice		

	cualquiera de la opciones de la gestión de ejercicios
Flujos alternativos:	Si no existe figuras retoricas el sistemas mostrara un mensaje de figura retoricas no disponible.
Requerimientos especiales	RF01

TABLA 29 CASO DE USO AGREGAR EJEMPLOS Y CASO DE USO GESTIONAR EJERCICIOS DE LAS FIGURAS RETORICAS.

ID	12		
Nombre	Visualizar ejercicios		
Necesidad	Revisar los ejercicios de un módulo determinado.	**ID/ Necesidad**	0012
Actores:	Profesor		
Descripción:	El usuario profesor podrá visualizar los ejercicios de un módulo existente.		
Disparadores:	Profesor requiere visualizar los ejercicios de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión. Deben existir ejercicios agregados por el rol profesor.		
Post-condiciones:	Los ejercicios serán visualizados por el profesor		
Flujo Normal:	1. Este caso de uso se inicia cuando el Profesor requiere visualizar los ejercicios existentes 2. El profesor se dirige a la opción de visualizar ejercicios de cualquiera de las figuras retoricas indicadas en el caso de uso de gestión de ejercicios 3. El profesor podrá visualizar los ejercicios situados en el pool de ejercicios 4. Para finalizar el profesor debe salir de la vista visualización de ejercicios		
Flujos alternativos:	1. El profesor podrá editar o eliminar los ejercicios visualizados. 2. El sistema arrojara un mensaje de confirmación al momentos de que un ejercicio sea eliminado o editado		

	3. El sistemas arrojara mensaje de error en caso en el que el pool de ejercicios este vacío
Requerimientos especiales	RF01

TABLA 30 VISUALIZAR EJERCICIOS.

ID	13		
Nombre	Agregar ejercicios		
Necesidad	Agregar nuevos ejercicios de un módulo determinado.	**ID/ Necesidad**	0013
Actores:	Profesor		
Descripción:	El usuario profesor podrá agregar los ejercicios de un módulo existente.		
Disparadores:	Profesor requiere agregar los ejercicios de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	El sistema arroja un mensaje de confirmación de adición del ejercicio requerido.		
Flujo Normal:	1. Este caso de uso se inicia cuando el Profesor requiere agregar ejercicios 2. El profesor se dirige a la opción de adicionar ejemplos de cualquiera de las figuras retoricas indicadas en el caso de uso de gestión de ejercicios 3. El profesor podrá diligenciar el formulario de adición de ejercicios 4. El profesor debe guardar los ejercicios adicionados. 5. Para finalizar el profesor debe salir de la vista adicionar ejemplos		
Flujos alternativos:	El sistemas arrojara mensaje de error en caso que el ejercicio no este correctamente diligenciado		
Requerimientos especiales	RF01		

TABLA 31 NARRATIVA DE CASOS DE USO AGREGAR EJERCICIOS

ID	14		
Nombre	Validar ejercicios		
Necesidad	Validar nuevos ejercicios de un módulo determinado agregados anteriormente.	**ID/ Necesidad**	0014
Actores:	Profesor		
Descripción:	El usuario profesor podrá validar los ejercicios de un módulo existente enviados por estudiantes.		
Disparadores:	Profesor requiere validar los ejercicios de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión. Ejercicios disponibles a validar.		
Post-condiciones:	El sistema arroja un mensaje de confirmación de validación correcta de ejercicios.		
Flujo Normal:	1. Este caso de uso se inicia cuando el Profesor requiere validar ejercicios 2. El profesor se dirige a la opción de validar ejercicios de cualquiera de las figuras retoricas indicadas en el caso de uso de gestión de ejercicios 3. El profesor podrá visualizar los ejercicios pendientes a validar 4. El profesor selecciona el ejercicios a validar y lo califica según la respuesta del estudiante(correcto o incorrecto) 5. Para finalizar el profesor debe salir de la vista Validar ejercicios		
Flujos alternativos:	El sistema arrojara mensaje en caso que no haya ejercicios a validar.		
Requerimientos especiales	RF01		

TABLA 32 CASO DE USO VALIDAR EJERCICIOS.

ID	15		
Nombre	Generar premisas		
Necesidad	El profesor requiere generar premisas dentro de la gestión de ejercicios.	**ID/ Necesidad**	0015
Actores:	Profesor		
Descripción:	El usuario profesor podrá generar premisas llenando el formulario requerido.		
Disparadores:	Profesor requiere generar premisas de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	El sistema arroja un mensaje de confirmación de generación de premisas y conclusiones.		
Flujo Normal:	1. Este caso de uso se inicia cuando el Profesor requiere generar premisas silogísticas dentro la gestión ejercicios 2. El profesor se dirige a la opción de generar premisas silogísticas 3. El profesor diligencia el formulario requerido 4. El profesor selecciona la opción generar premisas 5. El profesor escoge la opción guardar premisas. 6. El caso de uso finaliza cuando el profesor le indica salir de generador de premisas		
Flujos alternativos:	1. El sistema genera las conclusiones automáticamente después de la generación de premisas. 2. El sistemas arrojara error en caso de que se quiera agregar premisas repetidas		
Requerimientos especiales	RF01		

TABLA 33 CASO DE USO GENERAR PREMISAS.

ID	16		
Nombre	Generar reportes		
Necesidad	El profesor requiere generar reportes	**ID/**	0016

	de cómo está siendo el uso de la herramienta.	**Necesidad**	
Actores:	Profesor		
Descripción:	El usuario profesor podrá generar reportes de resultados del trabajo de los estudiantes.		
Disparadores:	Profesor requiere generar reportes de la herramienta.		
Pre-condiciones:	El usuario debe haber iniciado sesión. El rol estudiante debe realizar ejemplos y ejercicios.		
Post-condiciones:	El sistema mostrar el reporte generado.		
Flujo Normal:	1. Este caso de uso se inicia cuando el profesor requiere generar reportes de uso de la herramienta 2. Se dirige al módulo de generar reportes 3. El profesor seleccionara el reporte a generar 4. Finaliza cuando el profesor visualice el reporte requerido		
Flujos alternativos:	En caso de no hayan reporten disponible el sistema arrojara un mensaje alertando dicha situación		
Requerimientos especiales	RF01		

TABLA 34 CASO DE USO GENERAR REPORTES.

ID	17		
Nombre	Visualizar Teoría		
Necesidad	El estudiante requiere visualizar la teoría de un módulo en específico.	**ID/ Necesidad**	0017
Actores:	Estudiante		
Descripción:	El usuario estudiante podrá visualizar la teoría de la figura retórica que desee.		
Disparadores:	Estudiante requiere visualizar teoría de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	El sistema mostrara al usuario la teoría disponible para la		

	figura retórica seleccionada.
Flujo Normal:	1. Este caso de uso se inicia cuando el estudiante requiere visualizar teoría 2. Se dirige a la figura retórica deseada y selecciona la opción teoría 3. Finaliza cuando el estudiante visualiza la teoría disponible de la figura retórica seleccionada.
Requerimientos especiales	RF01

TABLA 35 CASO DE USO VISUALIZAR TEORÍA.

ID	18		
Nombre	Visualizar ejemplos		
Necesidad	El Estudiante podrá visualizar los ejemplos disponibles de cada módulo.	**ID/ Necesidad**	0018
Actores:	Estudiante		
Descripción:	El usuario estudiante podrá visualizar los ejemplos propuestos de la figura retórica que desee.		
Disparadores:	Estudiante requiere visualizar los ejemplos de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión.		
Post-condiciones:	El sistema mostrara al usuario el ejemplo disponible para la figura retórica seleccionada.		
Flujo Normal:	1. Este caso de uso se inicia cuando el estudiante requiere visualizar ejemplos de las figuras retoricas 2. Se dirige a la figura retórica deseada y selecciona la opción ejemplos 3. Finaliza cuando el estudiante visualiza el ejemplo disponible de la figura retórica seleccionada		
Flujos alternativos:	En caso de no hayan ejemplos disponibles para la figura retórica seleccionada el sistema arrojara un mensaje alertando dicha situación.		

Requerimientos especiales	RF01

TABLA 36 CASO DE USO VISUALIZAR EJEMPLOS.

ID	19		
Nombre	Practicar ejercicios		
Necesidad	El Estudiante necesita practicar en ejercicios lo visto en teoría y ejemplos.	**ID/ Necesidad**	0019
Actores:	Estudiante		
Descripción:	El usuario estudiante podrá practicar realizando los ejercicios propuestos de la figura retórica que desee.		
Disparadores:	Estudiante requiere realizar los ejercicios de una figura retórica.		
Pre-condiciones:	El usuario debe haber iniciado sesión. Deben existir ejercicios disponibles para la figura retórica seleccionada.		
Post-condiciones:	El sistema mostrara al usuario el ejercicio disponible para la figura retórica seleccionada.		
Flujo Normal:	1. Este caso de uso se inicia cuando el estudiante requiere Practicar de las figuras retoricas 2. Se dirige a la figura retórica deseada y selecciona la opción ejercicios 3. Finaliza cuando el estudiante visualiza el ejercicio disponible de la figura retórica seleccionada		
Flujos alternativos:	1. En caso de no hayan ejercicios disponibles para la figura retórica seleccionada el sistema arrojara un mensaje alertando dicha situación 2. El estudiante podrá realizar el ejercicio y enviara la respuesta para la validación del rol profesor 3. El sistemas validara el ejercicio de la figura retórica silogismos como correcto o incorrecto		
Requerimientos especiales	RF01		

TABLA 37 CASO DE USO PRACTICAR EJEMPLOS.

7. CONCLUSIONES

- Se realiza la construcción de la estructura de datos de las figuras retoricas, permitiendo los diferentes tipos de operaciones sobre esta, además se logra realizar generaciones, validaciones y creación de los modos silogísticos existentes, dando como resultado la automatización de estas operaciones filosóficas por lo cual se da por cumplido el objetivo estipulado.

- Se implementa el modelo pedagógico para el aprendizaje de las figuras retoricas conformado por el consolidado de teoría en el cual el estudiante podrá informarse y conocer diferentes tipos de conceptos, además de realizar una práctica con los diferentes ejemplos propuestos por su profesor y realizar una autoevaluación del conocimiento adquirido ejecutando y resolviendo los ejercicios propuestos en la herramienta web.

- Se determinaron los requerimientos de la herramienta web siendo estos los estándares mininos con los cuales se construyó la aplicación cumpliendo así el objetivo pactado.

- Se diseñó una interfaz siguiendo el estándar Java EE y el MVC, brindando con estas tecnologías un agradable y amigable entorno web en busca del correcto uso de la herramienta por parte de los usuarios. Además se diseñó un módulo de gestión de figuras retoricas para cada una de estas con el fin de cumplir los objetivos propuestos en el proyecto.

- Como parte del objetivo del desarrollo de módulos de la herramienta web, se realiza la generación del módulo de reportes el cual permite visualizar resultados tomando como referencia los ejercicios correctos e incorrectos respondidos por los estudiantes.

- Se implementa la aplicación en un servidor web, el cual permitirá el fácil acceso por parte de los miembros del grupo a la herramienta. Además se realiza copias de seguridad a la base de datos por medio del mysqldump asegurando la estructura de datos existente.

- El desarrollo de este proyecto fue útil a sus autores en la formación como ingenieros de sistemas, ya que permitió poner en práctica las herramientas aprendidas y conocimientos adquiridos durante la carrera en las diferentes áreas como ingeniería del software, bases de datos, y programación. Además de contribuir al desarrollo multidisciplinario.

8. RECOMENDACIONES

- Se recomienda implementar algoritmos que ayuden a la generación automática de ejercicios de las figuras retoricas símil, analogía y falacia basándose en el controlador desarrollado en este proyecto de grado(SyllogismController), el cual realiza la construcción ,validación y generación de silogismos validos

- Se recomienda alimentar al sistema con nuevas teoría, ejemplos y ejercicios para llegar a tener una base de conocimientos sólida y en la cual los estudiantes puedan aprender de cada una de las figuras retoricas y los profesores se puedan apoyar en esta herramienta web en su proceso diario de enseñanza.

BIBLIOGRAFIA

- “Bootstrap Get Started, W3Schools.com”, Internet: (http://www.w3schools.com/bootstrap/bootstrap_get_started.asp>)

- CARNERO, Silvia. “El Silogismo Historia y Desarrollo”. A Parte Rei. Revista de Filosofía. No.39, 2005.

- ÇIVICI. Çağatay, “Prime Faces User Guide 5.0”, Prime First Edition. Internet: (http://www.primefaces.org/docs/guide/primefaces_user_guide_5_0.pdf)

- FERNÁNDEZ GONZÁLEZ, José. GONZÁLEZ GONZÁLEZ, Benigno Martín. MORENO JIMÉNEZ, Teodomiro. B.M. Grupo Blas Cabrera Felipe –GITEP. Departamento de Didácticas Especiales. “Las Analogías como modelo y como recurso en la enseñanza de las ciencias”. Centro Superior de Educación. Universidad de la Laguna, 2003.

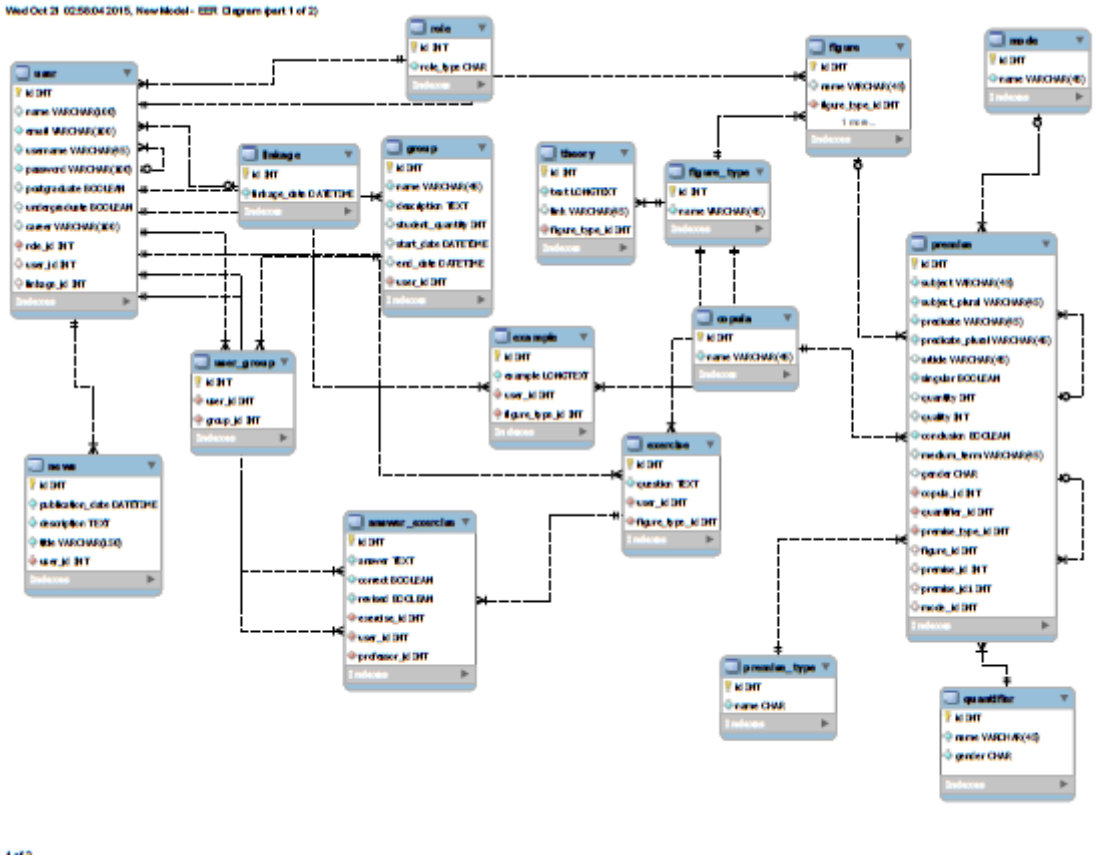
- GARCÍA BARRIENTOS, José Luis. “Las Figuras Retóricas:. El lenguaje literario 2”. Cuaderno de lengua española 56 Arco/Libros, S.L. 1º Edición, 1998.

- GIMENO, Juan Manuel. GONZÁLEZ, José Luis. “Introducción a Netbeans. Programación 2 – Curso 2010/2011”. Internet:(http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/1-introduccioi81n-a-netbeans.pdf>)

- GONCALVES, Antonio. “Beginning Java EE7, Ed. Apress, 2013.

ANEXOS

ANEXO1



Anexo 2

ANEXO 2 CASOS PRUEBA

1. Caso de prueba: 01
   Título: Existencia de interfaz de usuario
   Fecha: 19Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la url: http://filosofiayensenanza.uis.edu.co:8080/figurasretoricas/index	Visualización de página de inicio aplicación figuras retoricas	Se obtuvo el valor esperado

2. Caso de prueba: 02
   Título: inicio de sesión
   Fecha: 19Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar al módulo login e iniciar sesión con el usuario y password requerido	Según rol de usuario ingresa a los diferentes módulos de gestión de contenido, usuarios , o la práctica de la figura retórica esperada	Se obtuvo el valor esperado

3. Caso de prueba: 03
   Título: Agregar usuario
   Fecha: 19Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario Admin luego en el módulo de administración escoger el usuario y agregarlo.	Según rol de usuario que se requiera agregar el sistema visualizara los parámetros especificado para cada usuario(nombre ,email ,username y password ,carrer, group) inmediatamente después de diligenciada la información , y accionar el botón guardar se debe visualizar el usuario en la pantalla de gestión de usuarios	Se obtuvo el valor esperado

4. Caso de prueba: 04
   Título: Agregar grupo
   Fecha: 19Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario Admin , módulo de administración escoger grupo y agregar	Se desplegará una nueva vista la cual pedirá atributos del grupo , así como el profesor dueño del grupo, luego se accionara el botón guardar y este quedara guardado dentro de la ventana de administración de grupos	Se obtuvo el valor esperado

5. Caso de prueba: 05
   Título: Agregar teoría
   Fecha: 18Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario profesor ,módulo gestión, figuras retórica requerida y agregar teoría	Se visualizara la pantalla la teoría especificada de la figura retórica y esta permitirá la edición, el borrado o el guardado de la actual teoría mostrada al rol estudiante	Se obtuvo el valor esperado

6. Caso de prueba: 06
   Título: Agregar ejemplos
   Fecha:17Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario profesor ,módulo gestión, figuras retorica requerida , pestaña ejemplos	Se visualizara la pantalla los ejemplos agregados y este tendrá opciones de crear, editar y eliminar estos.	Se obtuvo el valor esperado

7. Caso de prueba: 07
   Título: Agregar ejercicios
   Fecha: 18Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario profesor ,módulo gestión, figuras retórica requerida , pestaña ejercicios	Se visualizara la pantalla los ejercicios agregados y este tendra opciones de crear, editar y eliminar estos.	Se obtuvo el valor esperado

8. Caso de prueba: 08
   Título: Agregar ejercicios
   Fecha: 14Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario profesor ,módulo gestión, figuras retórica requerida , pestaña ejercicios	Se visualizara la pantalla los ejercicios agregados y este tendra opciones de crear, editar y eliminar estos.	Se obtuvo el valor esperado

9. Caso de prueba: 09
   Título: Generar premisas
   Fecha: 15Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario profesor, módulo generador.	Se visualizara la pantalla cuadro de texto donde se podrá diligenciar sujeto y predica, además visualizarse el botono generar premisas que al accionarlo me creara las 4 premisas posibles las cuales se podrán guardar	Se obtuvo el valor esperado

10. Caso de prueba: 10
    Título: Generar conclusiones
    Fecha: 01Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario profesor ,módulo generador	Se visualizara un botón generar conclusiones.	Se obtuvo el valor esperado

11. Caso de prueba: 11
    Título: Consultar Teoría
    Fecha: 09Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario rol estudiante ,figura retórica requerida, teoría	Se visualizara la teoría disponible	Se obtuvo el valor esperado

12. Caso de prueba: 12
    Título: Consultar ejemplos
    Fecha: 20Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario rol estudiante ,figura retórica requerida, teoría	Se visualizara la teoría disponible para la figura retórica especificada	Se obtuvo el valor esperado

13. Caso de prueba: 13
    Título: Consultar ejemplos
    Fecha: 20Oct2015

PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario rol estudiante ,figura retórica requerida, ejemplos	Se visualizara los ejemplos disponibles para la figura retórica especificada	Se obtuvo el valor esperado

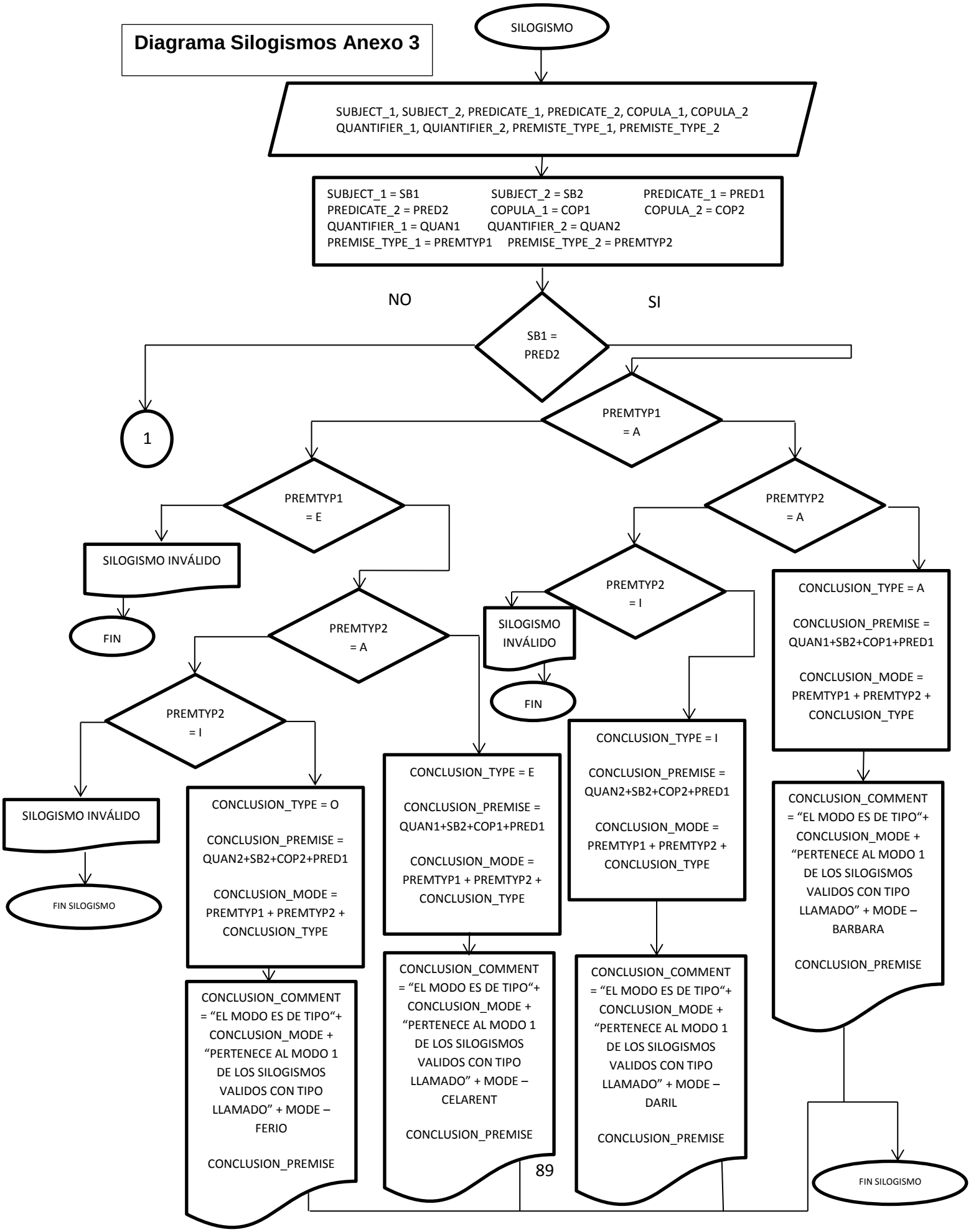
14. Caso de prueba: 14
    Título: Consultar ejercicios
    Fecha: 21Oct2015

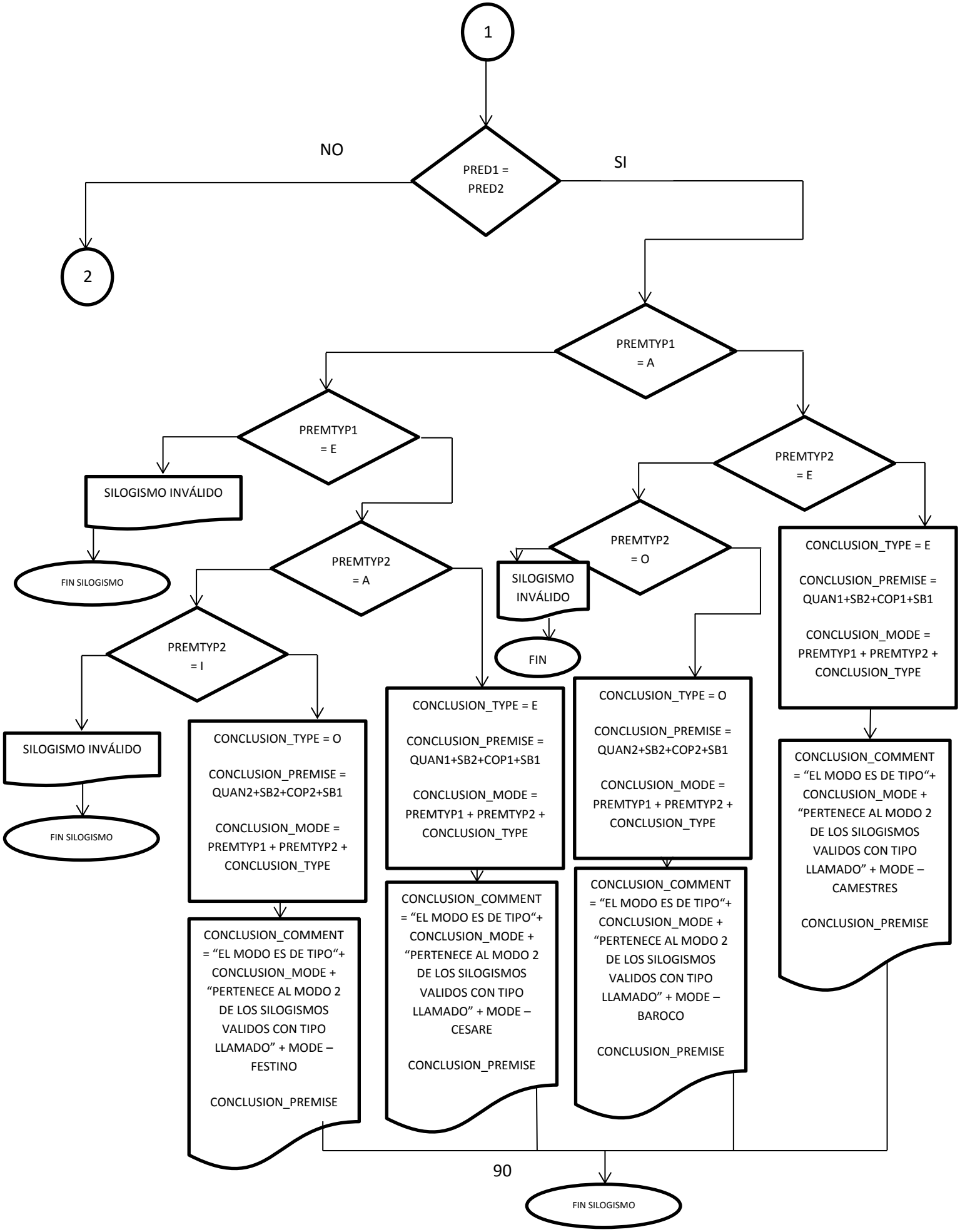
PASO	VALOR ESPERADO	VALOR OBTENIDO
Ingresar a la aplicación con el usuario rol estudiante ,figura retórica requerida, ejercicios	Se visualizara los ejercicios disponibles para la figura retórica especificada	Se obtuvo el valor esperado

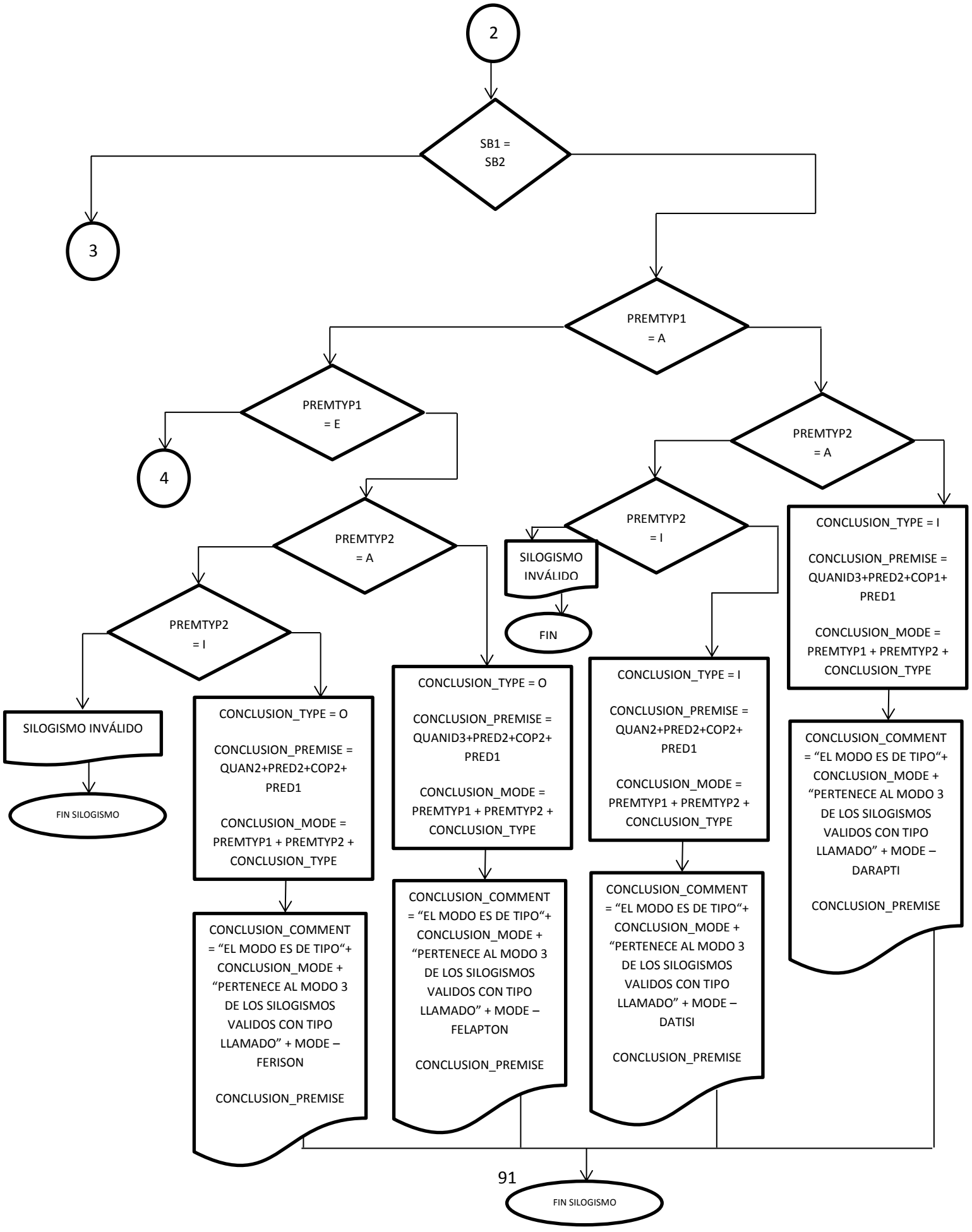
15. Caso de prueba: 15
    Título: cierre de sesión
    Fecha:22Oct2015

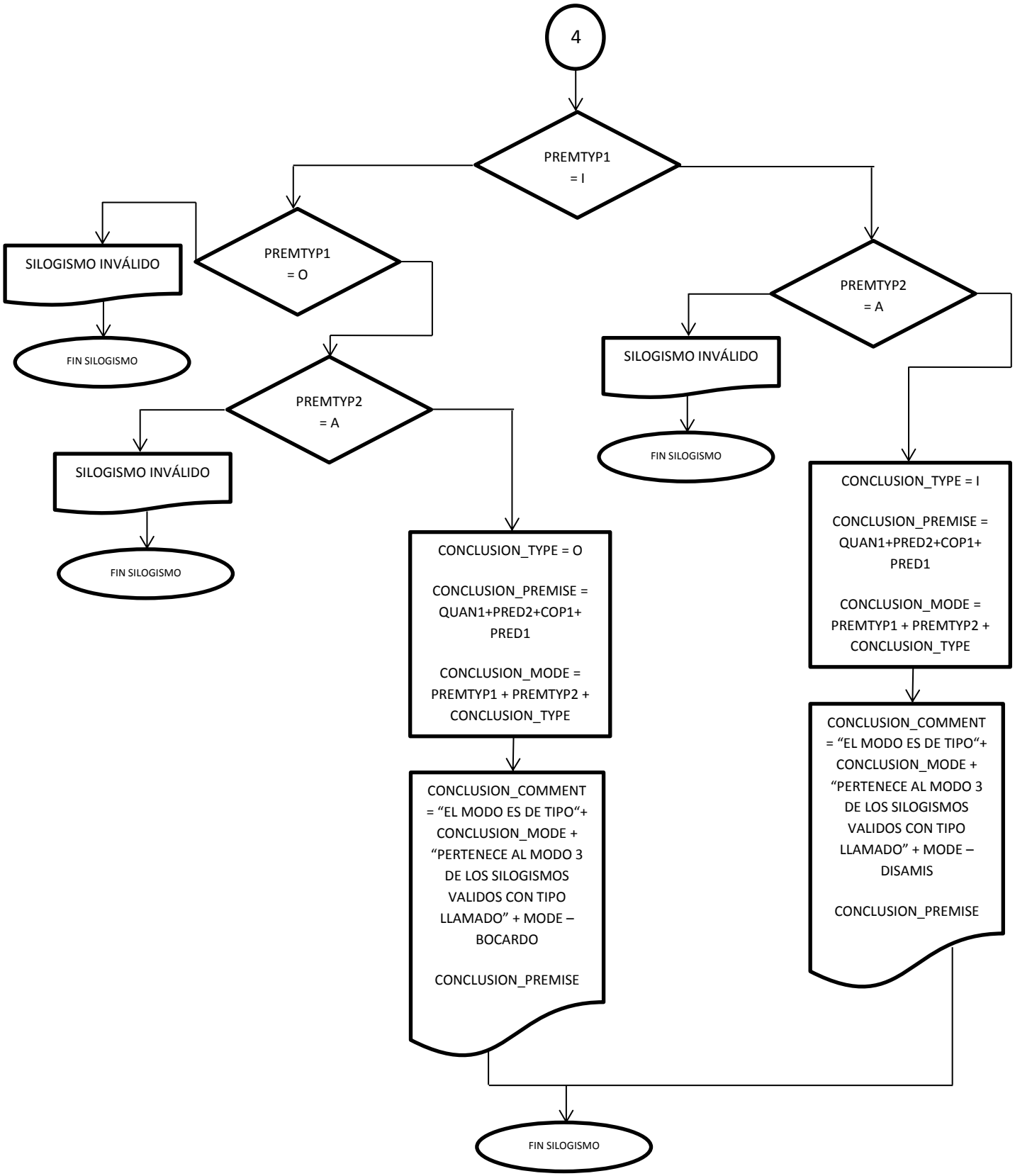
PASO	VALOR ESPERADO	VALOR OBTENIDO
Dentro de la página de inicio o cualquier otro modulo, ir logout	Se mostrara la página inicio de la aplicación	Se obtuvo el valor esperado

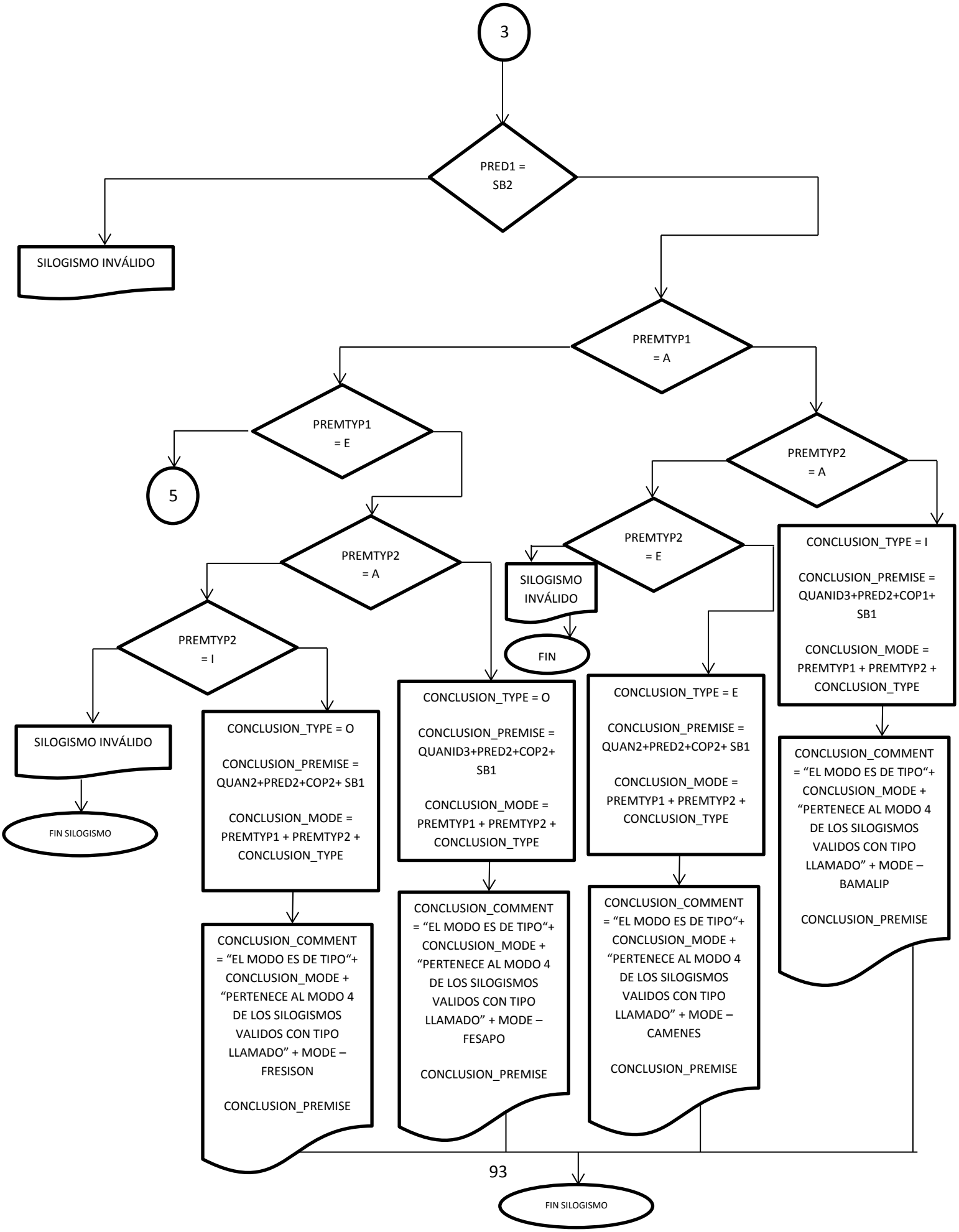
Diagrama Silogismos Anexo 3




1
PRED1 = PRED2
NO
SI
2
PREMTYP1 = A
PREMTYP1 = E
PREMTYP2 = E
SILOGISMO INVÁLIDO
FIN SILOGISMO
PREMTYP2 = A
PREMTYP2 = O
SILOGISMO INVÁLIDO
FIN
CONCLUSION_TYPE = E
CONCLUSION_PREMISE = QUAN1+SB2+COP1+SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 2 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – CAMESTRES
CONCLUSION_PREMISE
PREMTYP2 = I
SILOGISMO INVÁLIDO
FIN SILOGISMO
CONCLUSION_TYPE = O
CONCLUSION_PREMISE = QUAN2+SB2+COP2+SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 2 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – FESTINO
CONCLUSION_PREMISE
CONCLUSION_TYPE = E
CONCLUSION_PREMISE = QUAN1+SB2+COP1+SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 2 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – CESARE
CONCLUSION_PREMISE
CONCLUSION_TYPE = O
CONCLUSION_PREMISE = QUAN2+SB2+COP2+SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 2 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – BAROCO
CONCLUSION_PREMISE
FIN SILOGISMO


2
SB1 = SB2
3
PREMTYP1 = A
PREMTYP1 = E
4
PREMTYP2 = A
PREMTYP2 = I
SILOGISMO INVÁLIDO
FIN SILOGISMO
CONCLUSION_TYPE = O
CONCLUSION_PREMISE = QUAN2+PRED2+COP2+ PRED1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 3 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – FERISON
CONCLUSION_PREMISE
PREMTYP2 = A
PREMTYP2 = I
SILOGISMO INVÁLIDO
FIN
CONCLUSION_TYPE = O
CONCLUSION_PREMISE = QUANID3+PRED2+COP2+ PRED1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 3 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – FELAPTON
CONCLUSION_PREMISE
CONCLUSION_TYPE = I
CONCLUSION_PREMISE = QUAN2+PRED2+COP2+ PRED1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 3 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – DATISI
CONCLUSION_PREMISE
CONCLUSION_TYPE = I
CONCLUSION_PREMISE = QUANID3+PRED2+COP1+ PRED1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 3 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – DARAPTI
CONCLUSION_PREMISE
FIN SILOGISMO


4
PREMTYP1 = I
PREMTYP1 = O
SILOGISMO INVÁLIDO
FIN SILOGISMO
PREMTYP2 = A
SILOGISMO INVÁLIDO
FIN SILOGISMO
PREMTYP2 = A
SILOGISMO INVÁLIDO
FIN SILOGISMO
CONCLUSION_TYPE = O
CONCLUSION_PREMISE = QUAN1+PRED2+COP1+ PRED1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = “EL MODO ES DE TIPO“+ CONCLUSION_MODE + “PERTENECE AL MODO 3 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO” + MODE – BOCARDO
CONCLUSION_PREMISE
CONCLUSION_TYPE = I
CONCLUSION_PREMISE = QUAN1+PRED2+COP1+ PRED1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = “EL MODO ES DE TIPO“+ CONCLUSION_MODE + “PERTENECE AL MODO 3 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO” + MODE – DISAMIS
CONCLUSION_PREMISE
FIN SILOGISMO


3
PRED1 = SB2
SILOGISMO INVÁLIDO
PREMTYP1 = A
PREMTYP1 = E
5
PREMTYP2 = A
PREMTYP2 = E
PREMTYP2 = A
PREMTYP2 = I
SILOGISMO INVÁLIDO
FIN SILOGISMO
SILOGISMO INVÁLIDO
FIN
CONCLUSION_TYPE = O
CONCLUSION_PREMISE = QUAN2+PRED2+COP2+ SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 4 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – FRESISON
CONCLUSION_PREMISE
CONCLUSION_TYPE = O
CONCLUSION_PREMISE = QUANID3+PRED2+COP2+ SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 4 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – FESAPO
CONCLUSION_PREMISE
CONCLUSION_TYPE = E
CONCLUSION_PREMISE = QUAN2+PRED2+COP2+ SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 4 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – CAMENES
CONCLUSION_PREMISE
CONCLUSION_TYPE = I
CONCLUSION_PREMISE = QUANID3+PRED2+COP1+ SB1
CONCLUSION_MODE = PREMTYP1 + PREMTYP2 + CONCLUSION_TYPE
CONCLUSION_COMMENT = "EL MODO ES DE TIPO"+ CONCLUSION_MODE + "PERTENECE AL MODO 4 DE LOS SILOGISMOS VALIDOS CON TIPO LLAMADO" + MODE – BAMALIP
CONCLUSION_PREMISE
FIN SILOGISMO

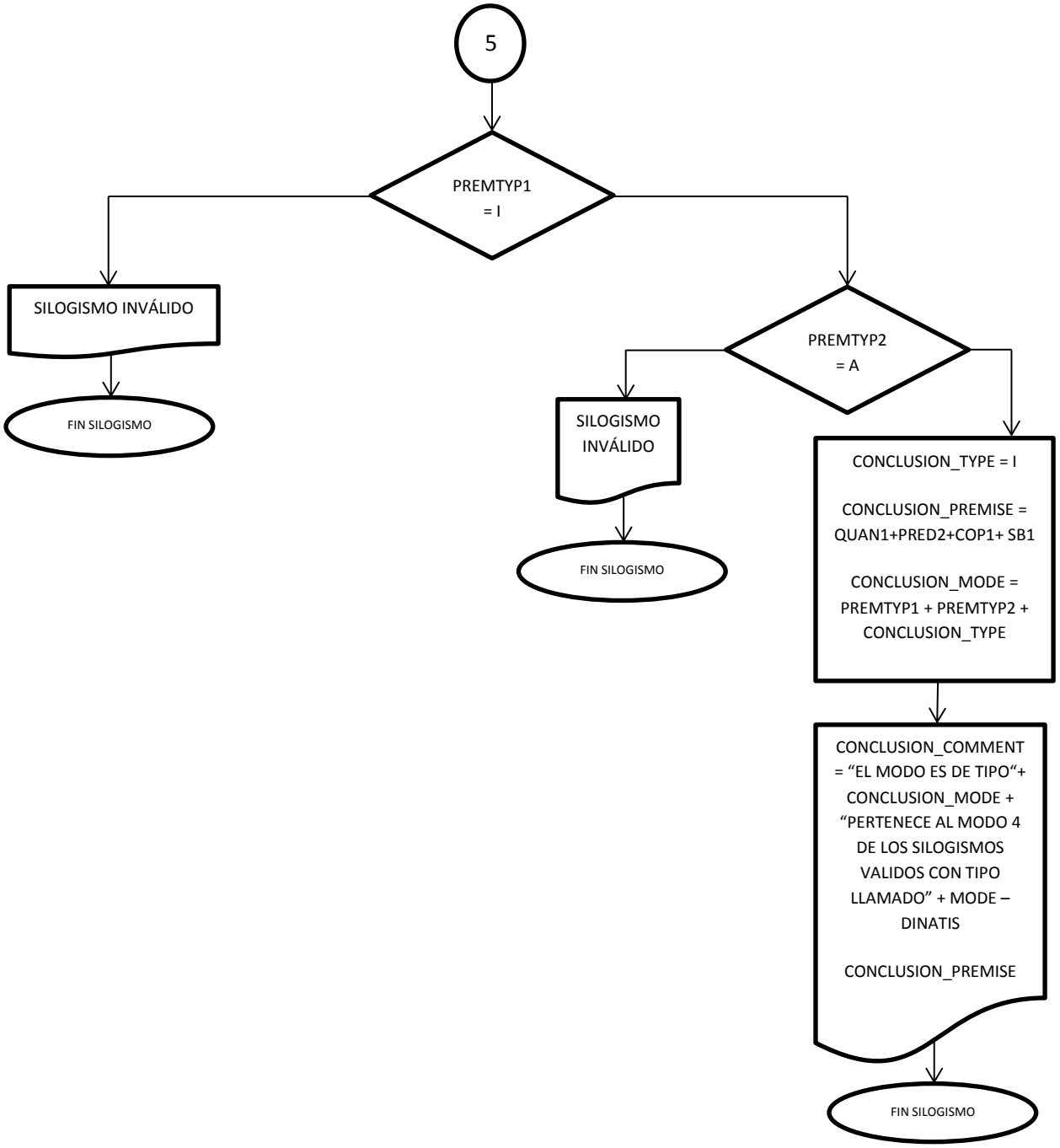
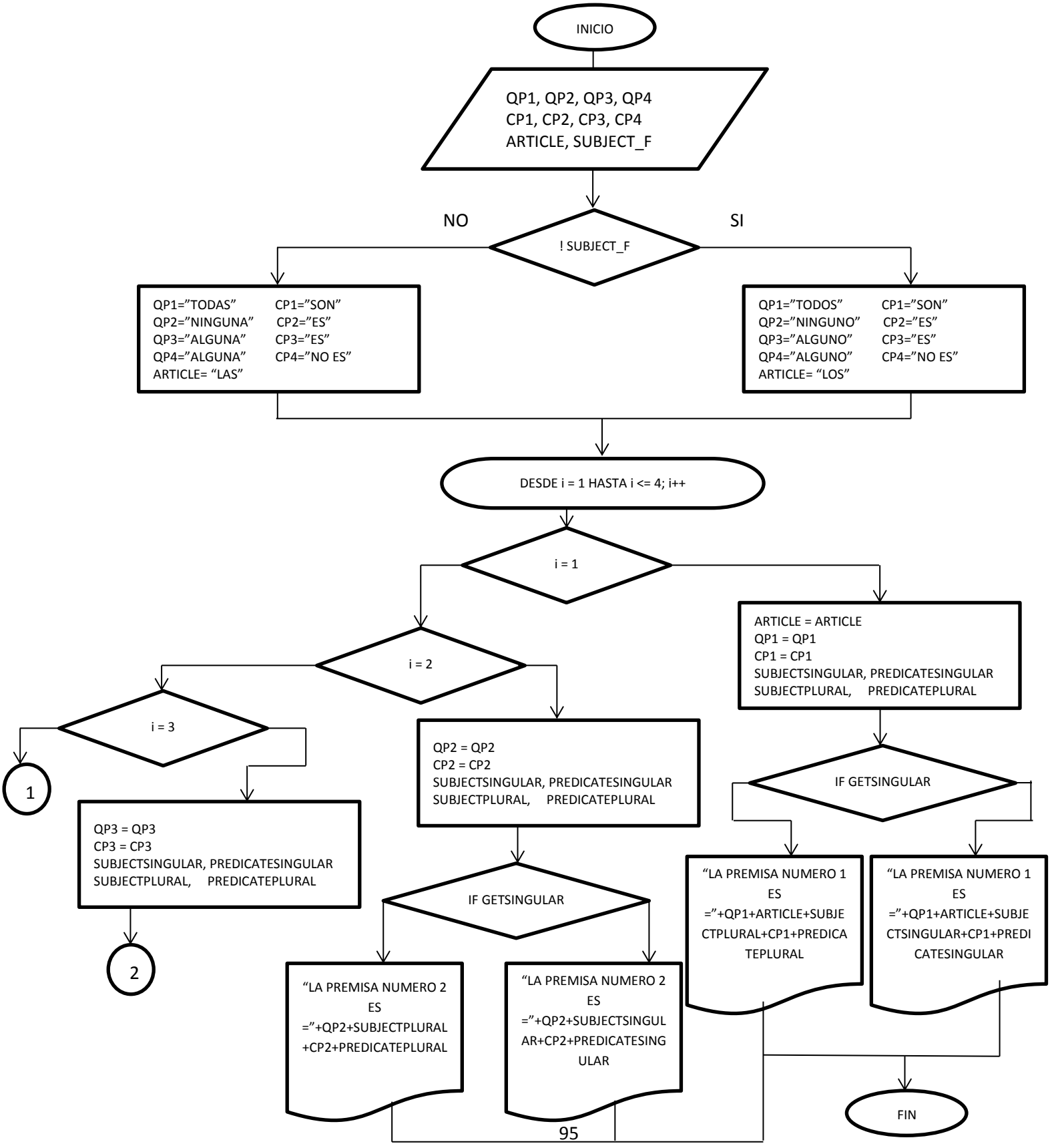

5
PREMTYP1
= I
SILOGISMO INVÁLIDO
FIN SILOGISMO
PREMTYP2
= A
SILOGISMO
INVÁLIDO
FIN SILOGISMO
CONCLUSION_TYPE = I
CONCLUSION_PREMISE =
QUAN1+PRED2+COP1+ SB1
CONCLUSION_MODE =
PREMTYP1 + PREMTYP2 +
CONCLUSION_TYPE
CONCLUSION_COMMENT
= “EL MODO ES DE TIPO“+
CONCLUSION_MODE +
“PERTENECE AL MODO 4
DE LOS SILOGISMOS
VALIDOS CON TIPO
LLAMADO” + MODE –
DINATIS
CONCLUSION_PREMISE
FIN SILOGISMO

DIAGRAMA GENERACIÓN DE PREMISAS (ANEXO 4)



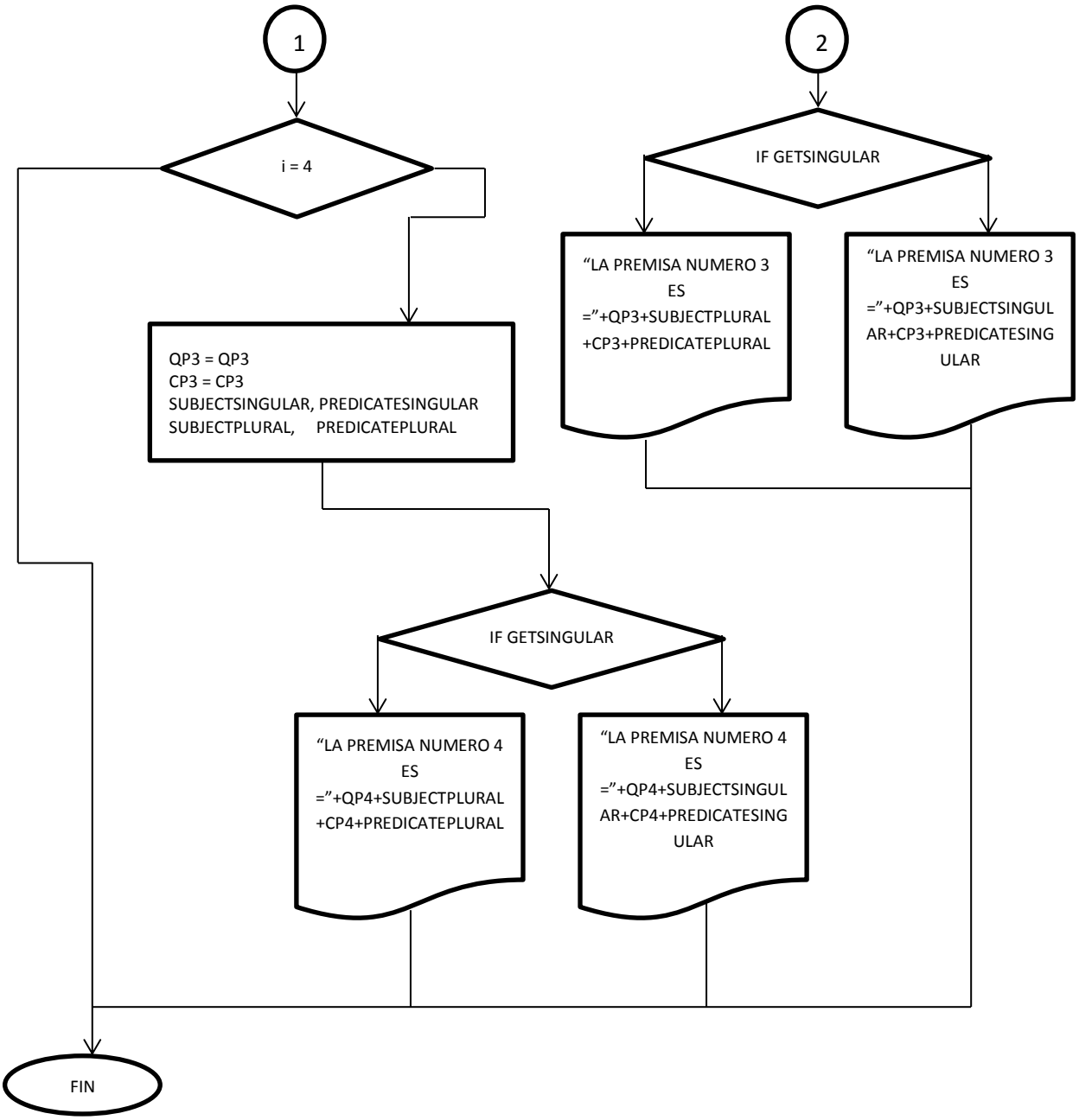

1
2
i = 4
IF GETSINGULAR
QP3 = QP3
CP3 = CP3
SUBJECTSINGULAR, PREDICATESINGULAR
SUBJECTPLURAL, PREDICATEPLURAL
“LA PREMISA NUMERO 3 ES =”+QP3+SUBJECTPLURAL +CP3+PREDICATEPLURAL
“LA PREMISA NUMERO 3 ES =”+QP3+SUBJECTSINGULAR+CP3+PREDICATESINGULAR
IF GETSINGULAR
“LA PREMISA NUMERO 4 ES =”+QP4+SUBJECTPLURAL +CP4+PREDICATEPLURAL
“LA PREMISA NUMERO 4 ES =”+QP4+SUBJECTSINGULAR+CP4+PREDICATESINGULAR
FIN

DIAGRAMA FLUJO CONVERTIR SUJETO Y PREDICADO SINGULARES EN PLURALES (ANEXO 5)