

Universidad de las Ciencias Informáticas

Facultad 6



**Título:**

# **Perfeccionamiento del Lenguaje Descriptor de Estructuras Químicas**

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

**Autores:** Varna Martha Hernández Gutiérrez.  
Yeniel González Dorta.

**Tutores:** Dr. Ramón Carrasco Velar  
Msc. Aurelio Antelo Collado  
Ing. Yania Molina.

Ciudad de La Habana, Junio-2008

“Año 50 de la Revolución.”

*"La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general pueden ser expresadas en un lenguaje comprensible para todos."*

*Albert Einstein.*

## DECLARACIÓN DE AUTORÍA

---

---

### DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los \_\_\_\_ días del mes de \_\_\_\_\_ del año \_\_\_\_\_.

Varna Martha Hernández Gutiérrez

Yeniel González Dorta

\_\_\_\_\_  
Firma

\_\_\_\_\_  
Firma

Dr. Ramón Carrasco Velar

Msc. Aurelio Antelo Collado

Ing. Yania Molina Souto

\_\_\_\_\_  
Firma del Tutor

\_\_\_\_\_  
Firma del Tutor

\_\_\_\_\_  
Firma del Tutor

## DATOS DE CONTACTOS

Dr. Ramón Carrasco Velar.

Centro de Química Farmacéutica, Ciudad de La Habana, Cuba.

Email: [rcarrasco@cqf.sld.cu](mailto:rcarrasco@cqf.sld.cu)

Msc. Aurelio Antelo Collado.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Email: [aantelo@uci.cu](mailto:aantelo@uci.cu)

Ing. Yania Molina Souto.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Email: [ymolinas@uci.cu](mailto:ymolinas@uci.cu)

## AGRADECIMIENTOS

*Varna:*

*Quisiera agradecerles a mis tutores por todo el apoyo y el conocimiento brindado.*

*Gracias a todas las personas que de una forma u otra han contribuido a mi formación profesional.*

*Gracias a la Revolución y a Fidel, por permitirme estudiar y formarme como una persona de bien.*

*Gracias a mi mamá, mi abuela Estrella y mi papá Joseito, por su educación, apoyo, por su amor durante toda mi vida y por ser muy especiales para mí.*

*Gracias a mi tía la Grilla y a mi prima Dailín por ayudarme, aunque estén lejos, siempre están conmigo.*

*Gracias a todos mis tíos y tías por preocuparse por mí, los quiero a todos por igual.*

*Gracias a Máriel por ayudarme, darme apoyo, acompañarme, guiarme, por comprenderme, por ser tan especial conmigo y por darme su amor.*

*Gracias a Yeniel por ser mi compañero de tesis, por soportar mis locuras durante este tiempo y ser muy paciente conmigo.*

*Gracias a todas mis amigas Yeni, Yenyssley, Celia, Daymara, Isais, Yadira, Mariagne e Irais, por estar cuando más me hacían falta, por escucharme, a todas gracias por sus consejos, espero que sepan que pueden contar conmigo.*

*Gracias a todas mi amistades, a todas las compañeras de cuarto que he tenido y a todas las personas que se preocuparon por mí.*

*Gracias a mis compañeros de aula, y a todos con los que he compartido el mismo grupo.*

*Gracias a la gente del proyecto, por ayudarme y compartir su tiempo conmigo.*

*A todos un beso y muchas gracias. Nunca los olvidaré.*

## AGRADECIMIENTOS

---

---

*Yeniel:*

*Quiero agradecer a todo aquel que ayudó a que este sueño se hiciese realidad.*

*A la Revolución cubana por permitirme cursar estudios en esta Universidad y convertirme en el hombre que hoy soy.*

*A mis padres y mi hermana por darme mucho amor y apoyarme en los momentos difíciles, hoy se que los hago muy felices.*

*A mi mujer Evelin por su apoyo y comprensión, ella sabe que la quiero mucho.*

*A mi familia por estar pendientes de mí y preocuparse por ello.*

*A mi mima por ayudarme muchísimo y por quererme tanto, hoy no me convierto en un médico, pero sé que sueña con verme graduado, para ella este es mi regalo.*

*A mis suegros por cuidar de mi hija cuando no estuve presente.*

*Agradecer muchísimo a mis amigos los cuales siempre estarán presentes como hermanos: Franklís, Yasiel, Abbie, Kader, fueron muchos los momentos para ser olvidados.*

*A mi jevita Yaisel por soportar mis pesadeces, ya no te molesto más.*

*A mi hermano negro por estar cuando necesitaba de un buen consejo, gracias por todo.*

*Al piquete que siempre estuvo en cada cuadra para despejar un poco: Jorgito, Ricardo, Pons, Dennis, Anders, Alejo, Victor, Angel, Yoe, Dayron. Voy a extrañar muchísimo esos momentos.*

*A mis colegas del laboratorio por cada momento que compartimos juntos: Yoander, Adonis, Yadira, Leo.*

*A Varna por ser mi dúo y formar parte de este trabajo que tanto soñamos, por preocuparse y dar lo mejor de sí en esta labor.*

*A las pijas y compañeras de estudio: Jenely, Marlien, Idelsis, Rosy, Dayamis, Yarmay, Liyanis, Yamira.*

## AGRADECIMIENTOS

---

---

*A Zoila por su constante preocupación con mi hija, siempre he valorado mucho eso.*

*A mis tutores por su gran apoyo y aporte.*

*Y a todos aquellos que pude haber dejado de mencionar y se preocuparon por el desarrollo de esta tesis, sepan que siempre les estaré muy agradecidos.*

*A todos muchas gracias.*

**DEDICATORIA**

*A mis hermanos Amelia y Juanqui.*

*A mi mamá, mi abuela Estrella y a mi papá  
Joseito.*

*A mi tía la Grilla y a mis primos Dailín,  
Gabi y Jorgito.*

*A mis futuros hijos.*

*A mis padres por ser mis guías.*

*A mi mujer y mi hija por ser mi  
inspiración.*

*A mi hermana por preocuparse tanto.*

*A mis amigos por apoyarme y ayudarme.*

*Varna*

*Yeniel*



### RESUMEN

El trabajo “Perfeccionamiento del Lenguaje Descriptor de Estructuras Químicas”, surge como parte del proyecto bioGRATO, este se desarrolla entre la Universidad de las Ciencias Informáticas (UCI) y el Centro de Química Farmacéutica (CQF). El objetivo principal de este trabajo es perfeccionar el lenguaje descriptor de estructuras químicas, a este se le dieron sus primeras definiciones en el período del 2006 al 2007, con el fin de aumentar la capacidad descriptiva de la actividad biológica<sup>1</sup> en compuestos orgánicos, para facilitar el diseño de nuevos fármacos con propiedades deseadas. El nuevo lenguaje no solo proporciona información topológica<sup>2</sup>, sino que también brinda información sobre propiedades químico-físicas, como la refractividad molecular<sup>3</sup> particionada sobre los átomos de la molécula, que es muy utilizada en estudios de correlación entre las estructuras químicas y la actividad biológica.

### PALABRAS CLAVE

Lenguaje descriptor, Actividad biológica, Fármacos, Refractividad molecular.

---

<sup>1</sup> Actividad Biológica: Actividad que caracteriza el comportamiento biológico en compuestos químicos (Molécula o Fragmento).

<sup>2</sup> Información topológica: es la información sobre como están enlazados los átomos y los tipos de átomos que contiene la molécula.

<sup>3</sup> Refractividad molecular: es el valor de la refractividad de la molécula.

## ÍNDICE

AGRADECIMIENTOS .....	I
DEDICATORIA .....	IV
RESUMEN.....	V
ÍNDICE DE FIGURAS .....	IX
ÍNDICE DE TABLAS .....	XI
INTRODUCCIÓN .....	1
FUNDAMENTACIÓN TEÓRICA .....	6
1.1    Estado del Arte .....	6
1.1.1 SMILES: Simplified Molecular Input Line Entry System. ....	7
1.1.2 SMARTS: SMiles Arbitrary Target Specification. ....	9
1.1.3 InChI: International Chemical Identifier. ....	11
1.1.3.1 InChIKey: International Chemical Identifier Key. ....	12
1.1.4 SSFN: Substructure Superposition Fragmental Notation. ....	13
1.1.5 DCAM: Descriptor Center Adjacency Matriz. ....	15
1.1.6 Nuevo Lenguaje Descriptor de Estructuras Químicas.....	16
1.2 Software de Propósito General. ....	17
1.2.1 Lenguajes de Programación.....	17
1.2.1.1 Lenguaje de programación C++.....	17
1.2.1.2 Lenguaje de programación C#.....	17
1.2.1.3 Lenguaje de programación Java.....	18
1.2.2 Entornos de Desarrollos Integrado (IDE). ....	18
1.2.2.1 Entorno de desarrollo: C++ Builder. ....	18
1.2.2.2 Entorno de desarrollo: Visual Estudio. Net.....	18
1.2.2.3 Entorno de desarrollo: NetBeans. ....	19
1.2.2.4 Entorno de desarrollo: Eclipse. ....	19
1.2.3 Herramientas CASE: Computer Aided Software Engineering. ....	19
1.2.3.1 Herramientas CASE: Rational Rose. ....	20

# ÍNDICE

---

---

1.2.3.2 Herramientas CASE: Visual Paradigm.....	20
1.3 Conclusiones.....	20
MATERIALES Y MÉTODOS.....	21
2.1 Materiales para el desarrollo del Lenguaje Descriptor.....	21
2.1.1 ¿Por qué Software Libre?.....	21
2.1.2 Lenguaje Representativo: UML.....	22
2.1.3 Lenguaje de Programación Java.....	22
2.1.4 Entorno de desarrollo: Eclipse.....	22
2.1.5 Visual Paradigm.....	23
2.2 Lenguaje Descriptor. Métodos y Definiciones para su desarrollo.....	23
2.2.1 Índice del Estado Refractotopológico Total.....	23
2.2.2 Reglas gramaticales del lenguaje.....	25
2.3 Codificación de Centros Descriptores y Caminos de Unión.....	26
2.3.1 Codificación del Centro Descriptor: Ciclo o Anillo.....	28
2.3.2 Codificación del Centro Descriptor: Clúster.....	30
2.2.3 Codificación del Centro Descriptor: Heteroátomos.....	32
2.3.4 Codificación del Metilo (CH <sub>3</sub> ), Metileno (CH <sub>2</sub> ), Metino (CH).....	35
2.3.5 Codificación del Camino de Unión.....	36
2.4 Codificación de un Fragmento.....	37
2.5 Decodificar al fichero .mol.....	39
2.6 Conclusiones.....	40
RESULTADOS Y DISCUSIÓN.....	42
3.1 Modelo conceptual.....	42
3.2 Diseño de clases.....	43
3.3 Descripción de los Algoritmos del Lenguaje.....	44
3.3.1 Búsqueda de CD Anillo.....	44
3.3.2 Búsqueda de CD Clúster.....	45
3.3.3 Búsqueda de CD Heteroátomo.....	45
3.3.4 Búsqueda de los CD metilo, metileno y metino.....	45
3.3.5 Búsqueda de CU.....	45

## ÍNDICE

---

---

3.4 Algoritmos del Lenguaje.....	45
3.5 Fichero de salida del lenguaje descriptor. ....	47
3.6 Pruebas.....	49
3.6.1 Pruebas de Codificación.....	49
3.6.2 Pruebas de Decodificación.....	51
3.7 Estudio Estadístico.....	55
3.7.1 Análisis de la muestra de Anillos de 6 átomos.....	57
3.7.2 Análisis de la muestra de Anillos de 3 a 5 átomos.....	58
3.8 Conclusiones.....	60
CONCLUSIONES .....	61
RECOMENDACIONES .....	62
REFERENCIAS BIBLIOGRÁFICAS.....	63
BIBLIOGRAFÍA.....	65
ANEXOS.....	69
Anexo #1. Algoritmos del lenguaje descriptor.....	69
Anexo # 2. Resultados de Codificación.....	75
GLOSARIO DE TÉRMINOS.....	77

## ÍNDICE DE FIGURAS

Figura # 1.1. Codificación SMILES del Ciclo Hexano.....	9
Figura #1.2. Molécula codificada con SSFN.....	14
Figura # 2.1. Anillo de 6 átomos de carbono.....	28
Figura # 2.2. Cadena de un anillo de 6 átomos.....	29
Figura # 2.3. Clúster de orden 3.....	30
Figura # 2.4. Clúster de orden 4.....	30
Figura # 2.5. Cadena de un clúster.....	31
Figura # 2.6. Cadena de los Heteroátomos.....	34
Figura # 2.7. Metilo, metileno y metino.....	35
Figura # 2.8. Cadena del metilo, metileno y el metino.....	36
Figura # 2.9. Camino de Unión entre dos CD.....	36
Figura # 2.10. Cadena de un Camino de Unión.....	37
Figura # 2.11. Composición de un Fragmento.....	38
Figura # 2.12. Fichero .mol sin coordenadas.....	39
Figura # 2.13. Fichero .mol con coordenadas.....	40
Figura # 3.1. Modelo conceptual.....	42
Figura # 3.2. Diagrama de Clases para codificar Estructuras Químicas.....	43
Figura # 3.3. Diagrama de Clases para decodificar a Ficheros .mol.....	44
Figura # 3.4. Algoritmo para buscar heteroátomos.....	46
Figura # 3.5. Algoritmo para buscar Grupo Funcional.....	46
Figura # 3.6. Molécula del hidroxipiridin-2.tiona.....	47
Figura # 3.7. Estructura de un Fichero del "LDEQODF" de la codificación del hidroxipiridin-2.tiona.....	47
Figura # 3.8. Encabezado del Fichero .Ide.....	48
Figura # 3.9. Codificación y Molécula del benzyl butanoate.....	50
Figura # 3.10. Codificación y Molécula del 1-hydroxypyridine-2-thione.....	51
Figura # 3.11. Fichero .mol antes de codificar.....	52
Figura # 3.12. Fichero .mol después de decodificar.....	53
Figura 3.13. Visualización de un fichero .mol antes de codificar.....	54
Figura 3.14. Visualización de un fichero .mol después de ser decodificado.....	54

## ÍNDICE DE FIGURAS

---

---

Figura # 3.15. Gráfica de la distribución del Índice de anillos de 6, 5, 4 y 3 átomos. ....	56
Figura # 3.16. Gráfica de la distribución del Índice en anillos de 6 átomos según su composición. ....	57
Figura # 3.17. Gráfica de la distribución del Índice en anillos de 5 átomos según su composición. ....	59
Figura # 3.18. Gráfica de la distribución del Índice en anillos de 3 átomos según su composición. ....	59
Figura # A1.1. Algoritmo para buscar CD Anillo. ....	69
Figura # A1.2. Algoritmo para buscar Clústeres primera parte. ....	70
Figura # A1.3. Algoritmo para buscar Clústeres segunda parte. ....	71
Figura # A1.4. Algoritmo para buscar CU.....	72
Figura # A1.5. Algoritmo para buscar el destino de un camino, primera parte.....	73
Figura # A1.6. Algoritmo para buscar el destino de un camino, segunda parte. ....	74
Figura # A2.1. Codificación y Molécula del 1H-pyrazole.....	75
Figura # A2.2. Codificación y Molécula del 1-methylimidazole. ....	75
Figura # A2.3. Codificación y Molécula del 2-amino-3-sulfanyl-propanoic acid. ....	76

## ÍNDICE DE TABLAS

Tabla # 1.1. Moléculas codificadas con el SMILES. ....	8
Tabla # 1.2. Reacciones químicas codificadas con el SMILES. ....	8
Tabla # 1.3. Operadores utilizados en el SMARTS. ....	10
Tabla # 1.4. Codificación de CD en el lenguaje SSFN. ....	14
Tabla # 2.1. Comparación de caracteres.....	27
Tabla # 3.1. Cantidad de anillos según la cantidad de átomos encontrados en la muestra. ....	55
Tabla # 3.2. Cantidad de anillos de 6 según la cantidad de Heteroátomos. ....	57
Tabla # 3.3. Cantidad de anillos de 3 a 5 según la cantidad de Heteroátomos. ....	58

### INTRODUCCIÓN

A lo largo de los años la química farmacéutica ha tenido un creciente desarrollo en cuanto al estudio e investigación de las propiedades y características de las estructuras moleculares. Estas investigaciones estaban relacionadas a fármacos y tóxicos, con el objetivo de conocer en detalle la estructura-actividad de dichos compuestos para su posterior análisis. Dichas investigaciones fueron proporcionadas, en su mayoría, por químicos, biólogos, bioquímicos y áreas afines al tema, que se encontraban esparcidos por todo el mundo. Para ellos, toda la información obtenida del estudio y análisis de las estructuras moleculares era de vital importancia, debido a que le posibilitarían diseñar nuevos compuestos con propiedades deseadas.

Para estos investigadores era necesario obtener una forma para comunicarse y no solo comunicarse, sino que existiera un entendimiento común entre ellos, debido al gran volumen de información que se manipulaba. En principio se utilizó el método gráfico para representar el análisis y estudio de los compuestos, por ser un método muy práctico pero a su vez resultaba poco eficiente cuando se necesitaba manejar mucha información. Otro método utilizado es la fórmula empírica, en este método es posible apreciar información acerca de la composición química de la estructura estudiada, pero presenta como inconveniente que dos o más compuestos que estén representados con la misma fórmula, se tratan por igual, puesto que en esta no se tienen en cuenta la composición de los enlaces. De ahí la necesidad de incursionar en nuevos modelos que permitan mostrar la información más detallada de la molécula.

Con el desarrollo de las ciencias de la computación y la informática se han desarrollado modelos matemáticos y se han diseñado lenguajes descriptores que tiene como base la codificación de las estructuras moleculares. Estos últimos no solo ayudaron a que se pudiera representar la estructura de sustancias de una forma más ampliada para los químicos e investigadores, sino que también condujeron a disminuir el costo económico y el tiempo en que un fármaco estuviera listo, convirtiéndose estos en la forma más estandarizada de comunicación, producto de la gran información que estos aportan para el estudio de las estructuras moleculares.

Ejemplos de estos lenguajes son:

- SMILES (Simplified Molecular Input Line Entry Specification).
- SMARTS (SMiles Arbitrary Target Specification).



- InChI (International Chemical Identifier).

Estos lenguajes descriptores han tenido varios enfoques del diseño molecular, todos basados en la interrelación estructura química-propiedades de las moléculas. La efectividad de estos métodos depende en gran medida de los descriptores moleculares seleccionados para caracterizar la estructura química. Hasta el momento se han utilizado diversos tipos de descriptores en el diseño de fármacos.

Ejemplo de ellos son:

- Químico-Cuánticos.
- Topológicos.
- Físico-Químicos.

Un procedimiento utilizado por los químicos para el trabajo con estructuras moleculares es la fragmentación de las mismas, ayudando a identificar cuál o cuáles de sus partes son responsables de determinadas propiedades que ellas manifiestan. Los lenguajes descriptores antes mencionados son muy útiles para el trabajo en búsqueda de fragmentos de las estructuras en grandes base de datos, sin embargo, tienen como principal problema que no proporcionan la información necesaria para los químicos, que les permita realizar una buena predicción de la actividad biológica. Pues todos poseen como inconveniente el hecho de que no contienen otra información estructural que no sea la topológica. Si se desea adicionar información a la descripción realizada por estos lenguajes, no se puede hacer a partir de la sintaxis de estos.

Un lenguaje que se creó con el fin de erradicar los inconvenientes que tenían los lenguajes anteriores fue el DCAM (Descriptor Centre Adjacency Matriz), como una versión más ampliada del lenguaje SSFN (Substructure Superposition Fragmental Notation). Este último fue creado y diseñado para el sistema experto OREX (Orbital Re-Entry EXperiment), su funcionamiento se basa en la descomposición topológica de las moléculas<sup>4</sup> en fragmentos estructurales y la asociación de estos fragmentos con la actividad biológica.

---

<sup>4</sup> Topología Molecular: Es toda la información (y la única) que puede obtenerse de la conectividad mutua entre todos los pares de átomos en una molécula.

Todos los lenguajes descriptores utilizan varias técnicas para la descripción de las moléculas, entre las más utilizadas está la teoría de grafos y el desarrollo de los índices topológicos y topográficos. Se toma la molécula como un grafo molecular, calculándose de este la matriz de adyacencia entre vértices o las matrices de distancia topológicas.

Recientemente, se han introducido dos índices híbridos, llamados así porque en su definición se ha ponderado la matriz de conectividad entre los vértices del grafo químico, por una propiedad químico-física particionada a nivel atómico como la refractividad molecular [1].

Cuba, a pesar de ser un país bloqueado desarrolla investigaciones con el fin de obtener vacunas y nuevos fármacos. Existen varios centros dedicados a estas funciones, uno de ellos es el Centro de Química Farmacéutica (CQF), institución dedicada al desarrollo de investigaciones científico-técnicas dirigidas a la obtención de sustancias bioactivas<sup>5</sup> para medicamentos de uso humano y veterinario.

En el año 2005 el CQF en conjunto con la Facultad de Bioinformática de la Universidad de las Ciencias Informáticas (UCI), comenzaron un proyecto llamado: Plataforma para la Predicción de Actividad Biológica de Compuestos Orgánicos. En este proyecto surge la necesidad de desarrollar un nuevo lenguaje descriptor de estructuras químicas capaz de ofrecer la información suficiente y necesaria para poder realizar predicciones de actividad biológica con mayor precisión. En este lenguaje se planteó la necesidad de establecer correlaciones entre la presencia de fragmentos ponderados por el Índice del Estado Refractotopológico Total<sup>6</sup> y la actividad biológica.

En el período del 2006 al 2007 se realizó una primera aproximación del lenguaje propuesto, del cual se definieron reglas y se identificaron fragmentos, se definieron caminos de unión y centros descriptores. En su desarrollo fue necesario redefinir este lenguaje para ampliar su capacidad descriptiva de las estructuras moléculas.

---

<sup>5</sup> Sustancias bioactivas: las sustancias bioactivas sólo las generan las plantas asumiendo importantes funciones, como protegerlas de las consecuencias de una radiación solar excesiva, de agentes patógenos y otros enemigos biológicos.

<sup>6</sup> Índice del Estado Refractotopológico Total: se desarrolla a partir de la teoría del grafo químico y de la partición de la refractividad atómica definida por Ghose y Crippen. El índice se basa en la influencia de las fuerzas de dispersión de cada átomo sobre cada uno de los restantes en la molécula, modificado por la topología molecular.

Por lo antes expuesto, se plantea como **problema científico** de la investigación: ¿Cómo mejorar la descripción de las estructuras químicas orientadas al diseño de fármacos para la predicción de la actividad biológica utilizando el Índice del Estado Refractotopológico Total?

Este problema enmarca su **objeto de estudio** en: La codificación de las estructuras químicas.

Abarcando como **campo de acción**: Los lenguajes descriptores de estructuras químicas.

Para dar solución a lo planteado se trazó como **objetivo general**: Perfeccionar el lenguaje descriptor de estructuras químicas en la Plataforma bioGRATO, que incluya información químico-física y esté orientado a la descripción de la actividad biológica.

Para cumplir este objetivo se plantean los siguientes **objetivos específicos**:

- Analizar, rediseñar e implementar algoritmos para el reconocimiento de centros descriptores, caminos de unión y fragmentos en un grafo molecular.
- Redefinir el modelo del lenguaje descriptor.
- Definir nuevas reglas gramaticales que identificarán el lenguaje.
- Implementar algoritmos de codificación y de decodificación para el lenguaje.
- Evaluar estadísticamente el comportamiento del Índice del Estado Refractotopológico Total en diferentes centros descriptores.

Para dar cumplimiento a los objetivos específicos se plantean las siguientes **tareas**:

- Familiarización con el lenguaje Java y la primera definición del lenguaje.
- Definición de la codificación del lenguaje.
- Desarrollo de algoritmos para la lectura y conversión a una matriz de adyacencia a partir de un .mol.
- Desarrollo de algoritmos para el reconocimiento de centros descriptores, caminos de unión y fragmentos a partir del grafo molecular.
- Generación de un fichero de salida para cada molécula que sea codificada con el nuevo lenguaje descriptor.

- Hacer un análisis estadístico sobre el comportamiento del valor del Índice del Estado Refractotopológico Total en centros descriptores.

Como resultado, se pretende brindar a la Plataforma bioGRATO un lenguaje capaz de describir estructuras moleculares a partir del análisis de fragmentos ponderados por el Índice del Estado Refractotopológico Total orientado a la descripción de la actividad biológica.

El documento está estructurado en 3 capítulos:

En el **Capítulo # 1** Fundamentación Teórica, se muestra un estudio detallado sobre los lenguajes utilizados en la actualidad a nivel mundial para la descripción de compuestos químicos.

En el **Capítulo # 2** Materiales y Método para el desarrollo del Lenguaje Descriptor, se realiza una selección de la tecnología actual con el objetivo de realizar un buen desarrollo del lenguaje descriptor. Se justifica porque fueron seleccionadas estas y no otras. También se brinda la redefinición y reglas gramaticales del lenguaje descriptor.

En el **Capítulo # 3** Resultados y Discusión, se muestran los resultados obtenidos a lo largo de la creación del Lenguaje Descriptor de Estructuras Químicas Orientadas al Diseño de Fármacos, y a la vez se discute sobre la importancia y relevancia de los mismos. Se realiza además un análisis estadístico sobre los centros descriptores y el Índice del Estado Refractotopológico Total asociado a estos.

# FUNDAMENTACIÓN TEÓRICA



El lenguaje es un medio que posee el hombre para comunicarse con el mundo que lo rodea, para ello durante el transcurso de su evolución y con la llegada de las computadoras el hombre ha vinculado muchas ramas de la ciencia con la informática, con el fin de hacer más fácil su trabajo y en menos tiempo. La química y específicamente: la química farmacéutica es una de las ciencias que ha encontrado apoyo en la informática, para la cual se han creado varios tipos de lenguajes que se emplean en esta, para la comunicación e intercambio de información estructural así como para describir las estructuras químicas. Representar a las moléculas en una computadora puede parecer algo difícil, sin embargo es una de las cosas que se ha podido lograr con estos lenguajes químicos. En el siguiente capítulo se muestra un estudio detallado sobre los lenguajes utilizados en la actualidad a nivel mundial para la descripción de compuestos químicos.

## 1.1 Estado del Arte

Los lenguajes descriptores son poco conocidos en el mundo, solo tienen conocimiento de su existencia aquellas personas que los utilizan, y no porque su uso esté restringido. Pero el hecho, de que no sean muy conocidos, no les resta importancia.

Los lenguajes descriptores permiten realizar un estudio y reconocimiento más detallado de las estructuras químicas y sus partes más importantes, dando información sobre la composición de las moléculas. Pueden realizar búsquedas de fragmentos según sus gramáticas en base de datos, y brindan la posibilidad de realizar predicciones más exactas sobre los compuestos a la hora del diseño de un fármaco. En los siguientes sub-epígrafes se brinda una descripción de los lenguajes descriptores más utilizados y conocidos en el mundo.

### 1.1.1 SMILES: Simplified Molecular Input Line Entry System.

El lenguaje SMILES (Simplified Molecular Input Line Entry System) o (Sistema de líneas de entrada molecular simplificada), es un lenguaje que fue desarrollado a finales de la década del 80 por David Weininger, basado en el concepto de gráfico molecular como principio para representar las estructuras químicas, tomando como nodo a los átomos y como aristas los enlaces entre ellos. Es un idioma químico en el que las moléculas y reacciones pueden ser especificadas usando caracteres ASCII. La tabla 1.1 y 1.2 muestran ejemplos de moléculas y reacciones químicas codificadas con el lenguaje SMILES.

Código SMILES	Nombre del compuesto
CC	Etano
O=C=O	Dióxido de Carbono
C#N	Cianuro de Hidrógeno
CCN(CC)CC	Tiretil-amina
CC(=O)O	Ácido acético
C1CCCCC1	Ciclohexano
c1ccccc1	Benceno
[OH3+]	Ion Hidronio
[2H]O[2H]	Óxido de deuterio
[235U]	Uranio - 235
F/C=C/F	E-Difluoroeteno
F/C=C\F	Z-Difluoroeteno

<chem>N[C@@H](C)C(=O)O</chem>	L-Alanina
<chem>N[C@H](C)C(=O)O</chem>	D-Alanina

**Tabla # 1.1. Moléculas codificadas con el SMILES.**

Reacción SMILES	Nombre
<chem>[I-].[Na+].C=CCBr&gt;&gt;[Na+].[Br-].C=CCl</chem>	Reacción Desplazamiento
<chem>(C(=O)O).(OCC)&gt;&gt;(C(=O)OCC).(O)</chem>	Intermolecular ó Esterificación

**Tabla # 1.2. Reacciones químicas codificadas con el SMILES.**

La notación del SMILES es una notación lineal, un método tipográfico que usa los caracteres imprimibles, para representar moléculas y reacciones. Es una serie de caracteres en las que no hay espacios, teniendo en cuenta que los átomos de hidrógeno pueden estar omitidos o no [2]. Estas cadenas pueden ser importadas a través de editores gráficos de moléculas, que realizan la conversión de formatos SMILES a modelos moleculares bidimensionales o tridimensionales.

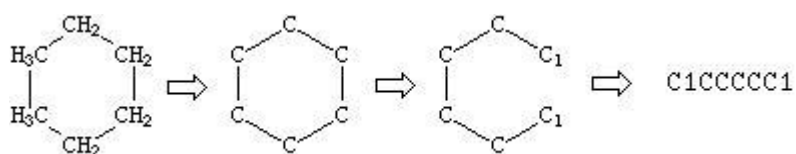
En el SMILES el nombre de una molécula codificada es sinónimo de su estructura. Contiene unas pocas reglas gramaticales y un vocabulario simple. Su gramática incluye información sobre la configuración de dobles enlaces.

Una de las ventajas que brinda este lenguaje es su fácil comprensión. Se emplea principalmente en la descripción de estructuras químicas siguiendo una notación. Puede realizar representaciones de las estructuras utilizando "palabras" en el vocabulario de otros idiomas diseñados para el almacenamiento de información sobre productos químicos.

Existen cinco reglas genéricas de codificación SMILES, las que corresponden a las especificaciones de:

- Los átomos: están representados por su símbolo atómico.
- Los enlaces: ya sean simples, dobles, triples, y aromáticos están representados por los símbolos -, =, #, and: respectivamente.

- Las ramas: se especifican incluyéndolas entre paréntesis y pueden estar anidadas o apiladas.
- Los cierres de anillos: las estructuras cíclicas están representada por el rompimiento de un enlace en cada anillo como se evidencia en la Figura # 1.1.



**Figura # 1.1. Codificación SMILES del Ciclo Hexano.**

- Estructuras desconectadas: estos compuestos son escritos como estructuras individuales separados por un punto "." (punto). El orden en que los iones están incluidos en la lista es arbitraria.

Es muy importante aclarar que pueden existir varios códigos SMILES válidos para una misma estructura. Este lenguaje ayudó mucho a los estudios de la química pero no proporciona mucha información acerca de los compuestos químicos, solo información topológica de las estructuras.

### 1.1.2 SMARTS: SMiles Arbitrary Target Specification.

El SMARTS (SMiles ARbitrary Target Specification) ó (Especificación arbitraria de blancos del SMILES) es un lenguaje que surgió después del SMILES, que identifica fragmentos dentro de las moléculas con determinadas propiedades y características. Describe patrones moleculares de un sub-grafo o fragmento en un grafo molecular.

Las reglas de este lenguaje son simples extensiones del SMILES, por lo que los símbolos, las propiedades y las especificaciones de este último son válidos en el SMARTS [3], aunque las semánticas de ambos sean diferentes, pues las expresiones del SMILES describen moléculas y las expresiones del SMARTS describen fragmentos de moléculas. Esto no ocurre cuando el SMILES trata de interpretar las expresiones del SMARTS.

Existen dos tipos de símbolos al igual que en el SMILES, que son los átomos y los enlaces. Una vez que el lenguaje identifica el gráfico de la molécula le asigna etiquetas al igual que el SMILES a los componentes del gráfico lo que con una diferencia, que las etiquetas de este lenguaje se le adicionaron



operadores lógicos y enlaces especiales lo que le permite ser más general. En la Tabla # 1.3 se muestran los operadores utilizados en el SMARTS.

Símbolo	Expresión	Significado
Exclamación	!e1	No e1(! significa negación)
Ampersand	e1&e2	e1 y e2 (alta prioridad)
Coma	e1,e2	e1 o e2 (ó)
Punto y coma	e1;e2	e1 y e2 (baja prioridad)

**Tabla # 1.3. Operadores utilizados en el SMARTS.**

El SMARTS realiza búsquedas y especificaciones basadas en encontrar un fragmento de importancia, que es tomado como un sub-grafo dentro de la molécula. Es utilizado para describir las pautas y las propiedades moleculares con distintos grados de especificación y generalidad. Puede realizar búsquedas eficientes de subestructuras en grandes base de datos y dar especificaciones en términos que son significativos para los químicos. Describe las propiedades atómicas más allá de los utilizados en el SMILES (símbolo atómico, carga, y especificaciones isotópica). Esto lo realiza gracias a una serie de símbolos primitivos que contiene este lenguaje.

Del lenguaje SMARTS surgieron varias versiones, esto trajo como consecuencia que comenzaran a existir desacuerdos entre los químicos y aparecieran incoherencias en el momento de representar las estructuras moleculares, lo que constituyó la principal desventaja de este lenguaje. Por lo que se tomó como acuerdo tomar todas las versiones del lenguaje que existían, y reunir de cada una de ellas las ventajas y desarrollar uno nuevo, con el objetivo de solucionar las diferencias entre los químicos, que recibió el nombre de InCHI.

A pesar de ser una mejora del SMILES, el SMARTS presenta el mismo problema de este, solo brinda información topológica de las estructuras químicas, y no permite adicionar características de los compuestos en su sintaxis.

### 1.1.3 InChI: International Chemical Identifier.

InChI (International Chemical Identifier) o (Identificador Químico Internacional), es un nuevo identificador químico, que se desarrolló en cooperación del IUPAC (International Union of Pure and Applied Chemistry) o (Unión Internacional de Química Pura y Aplicada) y el NIST (National Institute of Standards and Technology) o (Instituto Nacional de Normas y Tecnología). Fue desarrollado por Dimitri Tchekhovskoi, Steve Stein, Steve Heller y Alan McNaught en el período comprendido entre el 2000 y 2004, la primera versión fue liberada el 30 de agosto del 2006.

Su principal objetivo era proporcionar etiquetas únicas y bien definidas para las sustancias químicas que podrían ser utilizadas en fuentes de datos impresos y electrónicos. Resuelve muchas de las ambigüedades del SMILES y surge para acabar con todas las incoherencias que produjeron las versiones del SMARTS. Este lenguaje genera una cadena de caracteres para cada compuesto químico. Estas etiquetas son generadas por la conversión de un insumo de estructura química, a una única y previsible serie de caracteres ASCII.

La descripción de la estructura química se realiza en varios niveles, según los intereses, por lo que se clasifica como multinivel, identificando varias capas que describen características. Las estructuras químicas se expresan en términos de cinco capas de información: conectividad, tautomérica, isotópica, estereoquímica<sup>7</sup>, y electrónica [4]. Esto proporciona ventajas ya que los compuestos pueden ser comparados en varios niveles.

Este lenguaje es la manera más nueva de escribir las estructuras químicas en texto. Realiza recolección de los datos ya existentes de una forma más sencilla. Una de las principales ventajas de ese lenguaje es que puede ser utilizado por los químicos de todo el mundo sin costo alguno.

El InChI convierte las entradas estructurales en información de identificación, en un proceso de tres pasos:

- normalización (para eliminar información redundante).

---

<sup>7</sup> Estereoquímica: es la parte de la química que toma como base el estudio de la disposición espacial de los átomos que componen las moléculas y el cómo afecta esto a las propiedades de dichas moléculas. Se puede definir como el estudio de los isómeros.

- canonicalización <sup>8</sup> (para generar un conjunto único de átomo de etiquetas).
- serialización <sup>9</sup> (para dar una cadena de caracteres).

El seguimiento de estos procesos genera un identificador diferente para cada compuesto químico, pero siempre le otorga el mismo identificador a un determinado compuesto químico en particular, independientemente de la forma en que se realice la entrada de la estructura. Este procedimiento es aplicable a estructuras conocidas y desconocidas.

El lenguaje plateado resultó ser una solución para el problema que habían ocasionado las tantas versiones del SMARTS, pero tiene las mismas dificultades planteadas en los lenguajes anterior, solo brinda información topológica y no permite adicionar información de los compuestos químicos en su sintaxis.

El InCHI es un lenguaje que ha seguido evolucionando y se ha obtenido una nueva versión llamada InChIKey.

### 1.1.3.1 InChIKey: International Chemical Identifier Key.

En el año 2007 se desarrolló una versión del InCHI a la cual se le dio el nombre de InChIKey (International Chemical Identifier Key) o (Clave del Identificador Químico Internacional). Fue publicado en septiembre del 2007, como parte de la versión beta de InChI 1.02. Se trata de un identificador de longitud fija sobre la base de hash<sup>10</sup>, de una cadena InChI correspondiente. Su principal característica es que las cadenas generadas por este, tienen una longitud fija de 25 caracteres, directamente derivados de InChI.

Otras de las características que presenta esta nueva versión del InChI, es que facilita la búsqueda Web, que anteriormente no se podía realizar porque los motores de búsquedas realizaban rupturas imprevisibles en las cadenas de caracteres InChI. Permite una representación que se almacena en campos de longitud fija. Debido al número ilimitado de posibles moléculas y al limitado tamaño de la InChIKey es inevitable que varias moléculas tengan el mismo InChIKey [5].

---

<sup>8</sup> Canonicalización: significa escoger la mejor codificación para mostrar una molécula.

<sup>9</sup> Serialización: consiste en un proceso de codificación de una molécula.

<sup>10</sup> Hash: es un valor numérico de longitud fija que identifica datos de forma unívoca. Los valores hash se utilizan para comprobar la integridad de los datos.

Los 25 caracteres de la cadena están compuestos por dos partes unidas por un guión. La primera parte es de 14 caracteres, que se basa en la capa de conectividad. La segunda parte, contiene 9 caracteres que se relaciona con todas las demás capas InChI y el último es un caracter de control calculado del resto de la InChIKey. Esta nueva forma de expresión química no es un reemplazo para el código InChI, pero si un identificador para los productos químicos en Internet.

### **1.1.4 SSFN: Substructure Superposition Fragmental Notation.**




El SSFN (Substructure Superposition Fragmental Notation) ó (Notación de Fragmentos de Subestructuras en Superposición) es un lenguaje que fue creado en el año 1974, su algoritmo se basa en el concepto de centros descriptores y los caminos que los unen. Es la forma más sencilla en que se puede representar un compuesto químico. Cuando se codifican las estructuras, se identifican los centros descriptores y se crean listas de cadenas a partir de donde se encuentra un centros descriptores y donde termina con otro, sin pasar por un tercero [6]. Se recomienda este lenguaje durante el análisis de estructura química potencialmente activa para seleccionar los centros descriptores.

Los centros descriptores son fragmentos de moléculas, los cuales tienen ciertas propiedades, algunos de estos son los heteroátomos (N, O, S, P, halógenos, metales), los átomo de carbono e hidrógeno no son centros descriptores. En el lenguaje SSFN son centros descriptores los sistemas aromáticos cíclicos en conjunto y los pares de átomos de carbono conectados por enlaces múltiples (doble o triple, pero no aromáticos) [1]. También se consideran centros descriptores los grupos CH<sub>3</sub>, CH<sub>2</sub>, CH que se encuentran en extremos terminales de las moléculas.

El lenguaje SSFN proporciona las bases para:

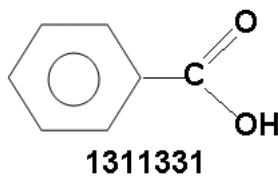
- Representación adecuada de farmacóforos con carácter topológico.
- Diseño de procedimientos eficaces para la identificación de farmacóforos durante el proceso de obtención de la estructura y datos de la actividad para los compuestos químicos.
- Desarrollo de algoritmos fiables para la predicción de actividad biológica de compuestos, basados en grupos farmacofóricos encontrados [6].

El lenguaje SSFN representa o clasifica cada CD con una cadena de caracteres, luego se unen todos estos caracteres para formar el código de la molécula. La Tabla # 1.4 muestra algunas de las cadenas que asigna el lenguaje a los centros descriptores.

Centro Descriptor	Código
	01
	03
<b>-N-H</b>	04
<b>-N-R</b>	05
<b>-N</b>	06
<b>OH</b>	11
<b>-O-R</b>	12
<b>=O</b>	13
	33

*Tabla # 1.4. Codificación de CD en el lenguaje SSFN.*

En la Figura #1.2 se muestra un ejemplo de cómo se codifica una molécula con este lenguaje.



**Figura #1.2. Molécula codificada con SSFN.**

La codificación que se muestra es debido a la cadena que se le asigna a cada centro descriptor definido para este lenguaje anteriormente en la tabla 1.4.

La principal ventaja que este presenta es la sencillez y claridad con que se reflejan las características de las estructuras representadas, muy importante para la manifestación de la actividad biológica. Es muy fácil de aprender para los químicos.

Su principal desventaja es la desintegración de las estructuras, almacenando información de los fragmentos estructurales de las moléculas pero no su interconexión, la cual ha causado la necesidad de crear un nuevo lenguaje que no contenga esta limitante, este recibió el nombre de DCAM.

### **1.1.5 DCAM: Descriptor Center Adjacency Matriz.**

El DCAM (Descriptor Center Adjacency Matrix) o (Matriz de Adyacencia de Centros Descriptores) surgió a partir del lenguaje SSFN, su algoritmo presenta una nueva forma de representar las estructuras. Como el DCAM es una versión más avanzada del SSFN está basado también en el mismo concepto. Su principal función es identificar y codificar centros descriptores. También contiene información sobre los enlaces que unen a los centros descriptores. Las estructuras se describen con una matriz simétrica a partir del grafo de la molécula, donde los nodos son los centros descriptores y las aristas son los caminos que los unen, a este gráfico se le llama centro de conexión de descriptor gráfico [6].

En esta matriz, la diagonal principal representa a los centros descriptores de la estructura y las entradas no diagonales, presentan información para describir las características de los enlaces presentes entre los centros descriptores. Las dimensiones de esta matriz dependen de la cantidad de centros descriptores que forman a la estructura molecular.

A pesar de que el DCAM presenta una sintaxis más detallada de los compuestos y brinda información sobre las propiedades, características y composición química de las estructuras moleculares, todavía no cumple con las necesidades para darle solución a todos los problemas que se plantean los químicos para obtener una mejor descripción de las moléculas, por esto se decidió desarrollar un nuevo lenguaje descriptor, utilizando conceptos del DCAM y nuevas propiedades químico-físicas que brinden mayor información sobre los compuestos químicos.

### 1.1.6 Nuevo Lenguaje Descriptor de Estructuras Químicas.

Es necesario crear un nuevo lenguaje descriptor de estructuras químicas debido a que los descritos anteriormente tienen dos desventajas en común.

- solo proporcionan información topológica
- no permite adicionar características de los compuestos químicos en su sintaxis.

Sería más apropiado un lenguaje que permitiera analizar un compuesto, no solo topológicamente, sino que permita adicionar información y reducir los costos a la hora de realizar investigaciones sobre un fármaco.

Para ello, se desarrolló un nuevo lenguaje descriptor llamado Lenguaje Descriptor de Estructuras Químicas Orientado al Diseño de Fármacos (LDEQODF). El nuevo lenguaje utiliza conceptos del DCAM y los combina con el Índice del Estado Refractotopológico Total, un índice descubierto por el Dr. Ramón Carrasco Velar. Este índice brinda mayor información sobre el comportamiento de las estructuras químicas. Es importante destacar que el nuevo lenguaje descriptor parte de la matriz de conectividad del grafo molecular desprovisto de hidrógeno.

Una de las características que identifica a este lenguaje es que los sub-grafos o fragmentos moleculares, se encuentran ponderados por el Índice del Estado Refractotopológico Total. Este índice permite diferenciar un mismo fragmento dentro de cada molécula según el valor que toma la refractividad molecular en el entorno en el que se le calcula, esto permite diferenciar fragmentos topológicamente idénticos, por la naturaleza de las fuerzas de dispersión de London<sup>11</sup> a él asociadas, dado esto como una función de su entorno molecular. Esto constituye una nueva forma de representación de la estructura química que responde a la necesidad de asociar aspectos estructurales que se diferencien con la actividad biológica [7].

EL nuevo lenguaje identifica centros descriptores de las moléculas, siendo estos los nodos y los caminos de unión que los unen, las aristas. De estos se dará una descripción más detallada en el próximo capítulo,

---

<sup>11</sup> Fuerzas de dispersión ó de London: son un tipo de fuerza inter molecular de Van der Waals. Esta está presente entre moléculas apolares, aunque también se dan en moléculas polares, solo que comparadas con las fuerzas dipolo-dipolo son muy pequeñas.

definiéndose que es un centro descriptores y un camino de unión. Se obtiene una representación de los fragmentos importantes de las moléculas.

Este lenguaje permite almacenar información y realizar estudios más detallados de los compuestos sobre la capacidad para interactuar con otras moléculas. Además, se utiliza una nueva forma de representación de la información que permite incluir los datos que anteriormente eran imposibles de añadir en los lenguajes anteriores por las restricciones y características particulares en las sintaxis [1].

### **1.2 Software de Propósito General.**

En este epígrafe se realizara un estudio de todo el software que se puede utilizar para el desarrollo del lenguaje descriptor. Se analizan sus características, ventajas y desventajas. Luego en el siguiente capítulo se realizara una selección de los que son convenientes para el desarrollo de la aplicación.

#### **1.2.1 Lenguajes de Programación.**

Los lenguajes de programación son herramientas que nos permiten crear programas y software. Está constituido por un grupo de reglas gramaticales y un grupo de símbolos utilizables. Los lenguajes pueden ser de alto o bajo nivel. En el caso que se plantea se utilizan los lenguajes de programación de alto nivel, a continuación se muestran varios lenguajes que se pueden utilizar para escribir el código del lenguaje descriptor.

##### **1.2.1.1 Lenguaje de programación C++.**

C++ es un lenguaje en el que se puede programar orientado a objeto y estructurado, este fue diseñado a mediados de los años 80 del siglo XX. Brinda la posibilidad de redefinir los operadores e identificación de tipos en tiempo de ejecución. Es un lenguaje muy potente debido a que permite trabajar tanto a alto como a bajo nivel.

##### **1.2.1.2 Lenguaje de programación C#.**

El lenguaje de programación C# es orientado a objetos, permite a los desarrolladores crear aplicaciones sólidas y seguras que se ejecutan en la plataforma .NET, en ella existen un conjunto de nuevas tecnologías proporcionadas por Microsoft, con el objetivo de proporcionar una plataforma sencilla y



potente para el desarrollo de aplicaciones y servicios software. Este lenguaje tiene como principal desventaja que no es multiplataforma.

### **1.2.1.3 Lenguaje de programación Java.**

El lenguaje Java es un lenguaje muy potente, puramente orientado a objetos. Esto significa que los programas se construyen a partir de módulos independientes, y que estos módulos se pueden transformar o ampliar fácilmente [8]. Las aplicaciones desarrolladas en este lenguaje pueden funcionar en varios tipos de computadores y sistemas operativos, por lo que es un lenguaje multiplataforma. Tiene un modelo de objetos más simple y elimina herramientas de bajo nivel. Fue liberado bajo la licencia GNU GPL, lo que lo convierte en software libre. Permite realizar aplicaciones para todo tipo de entornos, ya sea para Web, dispositivos móviles, aplicaciones de escritorio, servidor, etc.

### **1.2.2 Entornos de Desarrollos Integrado (IDE).**

Los entornos de desarrollos integrados o IDE es una aplicación que reúne varios programas necesarios para el desarrollador, tales como: editor, compilador y depurador, en los cuales el desarrollador puede escribir, evaluar y compilar el código de la nueva aplicación o software que este creando.

#### **1.2.2.1 Entorno de desarrollo: C++ Builder.**

Builder es un entorno de desarrollo de Borland para la programación bajo Windows por lo que no es una herramienta multiplataforma, basado en el lenguaje C++ y en el entorno Delphi. No es software libre. Las aplicaciones creadas con C++Builder son basadas en eventos. El entorno de desarrollo es completamente visual, y permite el desarrollo de aplicaciones de escritorio, aplicaciones webs, servicios de sistema, servicios Web.

#### **1.2.2.2 Entorno de desarrollo: Visual Estudio. Net.**

La plataforma Visual Estudio un entorno de desarrollo sencillo, pero a la vez muy potente, se pueden desarrollar aplicaciones Web y aplicaciones de escritorio en cualquier entorno que soporte la plataforma .NET. Soporta diferentes lenguajes de programación, tales como, Visual C++, Visual C#, Visual Java y

Visual Basic. Su principal desventajas que es una plataforma para el sistema operativo Windows y no es software libre.

### **1.2.2.3 Entorno de desarrollo: NetBeans.**

El NetBeans es entorno de desarrollo para Java, hecho en Java y código abierto, bajo licencia SPL, que soporta los lenguajes Java, C, C++, HTML y XML, las herramientas Ant y CVS y las tecnologías JSP, RMI, CORBA, JINI, JDBC y servlet. Fue diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. Permite crear interfaces gráficas de forma visual, desarrollo de aplicaciones Web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco sus funcionalidades son ampliables mediante la instalación de paquetes [9].

### **1.2.2.4 Entorno de desarrollo: Eclipse.**

Eclipse es un entorno de desarrollo de código abierto, liberado por IBM, pero desarrollado por varias compañías del sector. Aunque es un IDE para desarrollar aplicaciones en Java, tiene como principal ventaja que está basado en plugins<sup>12</sup>: puedes conseguir cientos de plugins para programar en C, C++, Perl, PHP, XML y Python, y no solo para programar sino también para modelar en UML [10]. Este mecanismo de plugins permite que sea una plataforma ligera para componentes de software.

### **1.2.3 Herramientas CASE: Computer Aided Software Engineering.**

La herramientas CASE (Computer Aided Software Engineering) o (Ingeniería de Software Asistida por Computadoras) son herramientas que tienen el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema. Tienen una alta cultura de ingeniería para desarrollar software. Contribuye a mejorar la calidad y la productividad en el desarrollo de sistemas de información.

---

<sup>12</sup> **Plugins:** Función o utilidad generalmente muy específica. Se adiciona a algún programa para ser ejecutado. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, etc.

### **1.2.3.1 Herramientas CASE: Rational Rose.**

El Rational Rose es una potente herramienta que se utiliza para el diseño de software, utilizando el UML como lenguaje de modelado. Genera código en lenguajes: C + +, Visual Basic, Java, Oracle y CORBA. Una de las ventajas que presenta esta herramienta es que proporciona un desarrollo iterativo de ingeniería de ida y vuelta [11]. Soporta todas las fases de vida de desarrollo del software. Como resultado de su uso se obtiene software de gran calidad y en menos tiempo. La principal desventaja de esta herramienta es que no es multiplataforma, es decir, no funciona en varios sistemas operativos, solamente en Windows, y además es software propietario, por lo que hay que tener licencia para desarrollar software.

### **1.2.3.2 Herramientas CASE: Visual Paradigm.**

El Visual Paradigm es una herramienta profesional que se utiliza para el diseño de software, utiliza UML como lenguaje de modelado, soporta el ciclo de vida del desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue [12]. Ayuda a desarrollar una rápida construcción de aplicaciones de calidad y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. Es una herramienta multiplataforma, se puede trabajar con ella en diferentes sistemas operativo. Es propietaria por lo que hay que tener licencia para desarrollar software en ella.

## **1.3 Conclusiones.**

Para desarrollar el nuevo lenguaje descriptor fue necesario realizar un análisis de los lenguajes descriptores de estructuras químicas existentes en el mundo, para entender su funcionamiento, características y ver sus principales dificultades. En este capítulo se da una segunda aproximación de lo que será el nuevo lenguaje descriptor, algunas de sus características, las cuales posteriormente en el próximo capítulo se tratará de una forma más abarcadora y descriptiva.

También se realizó un estudio de los software y lenguajes de programación, con el objetivo de seleccionar los software para el desarrollo del lenguaje descriptor.

# MATERIALES Y MÉTODOS



Los elevados costos de fabricación de un nuevo fármaco ocasionó la necesidad de obtener una vía para facilitar y reducir estos. Es así que comienza el desarrollo de una diferente y renovadora forma de representar las estructuras moleculares, que posibiliten la visualización y edición de estructuras químicas para permitir el análisis del comportamiento de las mismas, además de contar con la capacidad de poder predecir la actividad biológica de compuestos orgánicos.

En este capítulo se realiza una selección de la tecnología actual con el objetivo de realizar un buen desarrollo del lenguaje descriptor. Se justifica porque fueron seleccionadas estas y no otras. También se brinda la redefinición y reglas gramaticales del lenguaje descriptor.

## **2.1 Materiales para el desarrollo del Lenguaje Descriptor.**

Las principales características que deben cumplir el software para el desarrollo del Lenguaje Descriptor son:

- Multiplataforma.
- Software libre o software debidamente legalizado.
- Portables.

### **2.1.1 ¿Por qué Software Libre?**

Se decidió desarrollar el nuevo lenguaje descriptor en herramientas que fueran software libre por los siguientes motivos:

- Libre distribución.

- Seguras.
- Amplia disponibilidad de herramientas y librerías.
- Accesibilidad y modificación al código para realizar tareas específicas.

### **2.1.2 Lenguaje Representativo: UML.**

UML (Unified Modeling Language) o (Lenguaje Unificado de Modelado), es un lenguaje para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de un software. Es un lenguaje estándar para escribir planos de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. Es una notación orientada a objeto, que se compone por distintos diagramas, que representan las diferentes etapas del desarrollo de un proyecto de software.

Los principales motivos que condicionaron el uso de UML en la modelación del nuevo lenguaje descriptor son: lenguaje que no tiene propietario, contiene todos los conceptos necesarios para utilizar en un proceso moderno iterativo, con el objetivo de construir una arquitectura sólida. Capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Es un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes, brinda una visión más gráfica de cómo debe estar estructurado el software que se va a desarrollar en varias etapas del mismo al equipo de desarrollo [13]. UML es utilizado y adaptable a muchas metodologías de desarrollo de software.

### **2.1.3 Lenguaje de Programación Java.**

El lenguaje de programación que se utilizó para el desarrollo del lenguaje descriptor fue Java, debido a que la primera definición del lenguaje descriptor se realizó con este lenguaje, además de que es utilizado por el proyecto en donde se desarrolla el nuevo lenguaje descriptor por todas las ventajas y bondades que brinda este lenguajes.

### **2.1.4 Entorno de desarrollo: Eclipse.**

El entorno de desarrollo que se eligió para la implementación del nuevo lenguaje descriptor, debido a que es una herramienta muy potente y de libre distribución fue Eclipse. Se ejecuta sobre máquinas virtuales lo que le permite que este IDE se adapte a distintas arquitecturas de hardware y a distintos sistemas

operativos, su uso más popular es como plataforma de programación para Java. Es una plataforma portable que proporciona un marco de trabajo para desarrollar aplicaciones y herramientas. Forma parte de la comunidad de código abierto o software libre.

### 2.1.5 Visual Paradigm.

Como herramienta Case se utilizó Visual Paradigm, porque se integra muy bien con el Eclipse y genera código en Java, aunque en esta versión esta bondad no se utilizó porque ya existía una primera definición del lenguaje. Proporciona al modelado de procesos de negocio un objeto-relacional y aporta los últimos estándares de anotaciones de JAVA y UML. Es una herramienta muy flexible, se pueden crear todos los diagramas del UML o los que solo sean necesario para el desarrollo del software, y se utiliza legalmente.

## 2.2 Lenguaje Descriptor. Métodos y Definiciones para su desarrollo.

En el proceso de identificación de grupos farmacóforos<sup>13</sup> está demostrado que en un compuesto molecular solo es responsable de la actividad biológica una o algunas parte de este, por lo que para el desarrollo del "LDEQODF" se establecieron importantes agrupaciones o fragmentos atómicos relacionados a las propiedades biológicas estudiadas [14].

- Centros Descriptores (**CD**): Son pequeñas, medianas o grandes agrupaciones de átomos que se encuentran estrechamente relacionados, que pueden o no, constituir funcionalidad o influencia en cuanto al comportamiento de un compuesto orgánico.
- Caminos de Unión (**CU**): Son aquellas agrupaciones de átomos interconectadas, que unen a dos CDs y que en su composición no contiene a otro CD.
- Fragmentos: Dos CD relacionados por un CU.

### 2.2.1 Índice del Estado Refractotopológico Total.

El Índice del Estado Refractotopológico Total que se utiliza en el lenguaje descriptor, es un índice híbrido, resultado de la modificación del índice del estado electrotopológico de Kier y Hall ponderando la matriz de

---

<sup>13</sup> Farmacóforos: Esqueleto" molecular que "transporta" (*phoros*) los elementos esenciales responsables de que un compuesto tenga una actividad biológica determinada (*pharmacón*).

conectividad de los vértices del grafo químico por los valores de refractividad particionada en la molécula por el método de Ghose y Crippen. Este índice refleja por lo tanto la influencia de las fuerzas de dispersión de cada átomo sobre cada uno de los restantes en la molécula, modificado por la topología molecular [15]. Por lo que es un identificador único de cada átomo según la molécula en que se encuentre, y se calcula por la expresión:

$$R_i = A_{Ri} + \Delta A_{Ri}$$

Donde  $A_{Ri}$  es el valor intrínseco de refractividad del átomo<sup>14</sup>  $i$  y  $\Delta A_{Ri}$  es el efecto de la perturbación sobre el átomo  $i$  producido por los restantes átomos presentes en la molécula, y se calcula por la expresión:

$$\Delta A_{Ri} = (A_{Ri} - A_{Rj}) / r_{2ij}$$

Donde  $r_{ij}$  es la distancia euclidiana<sup>15</sup> entre los átomos  $i$  y  $j$  tomada de la matriz de distancias, correspondiente a la configuración de mínimo energético calculada por algún método semiempírico [15].

Cuando se desea calcular el Índice del Estado Refractotopológico Total de un CD, se hace a través de la suma de todos los índices de los átomos que lo componen. En [15] se demuestra que la suma de los valores correspondientes a cada átomo de una molécula dada, coincide con el valor de la refractividad molecular de dicha molécula. Como resultado, se ha planteado que puede demostrarse que, la suma de los valores del Índice del Estado Refractotopológico Total de cada átomo en un fragmento determinado, coincide con la refractividad de dicho fragmento en esa molécula; y por lo tanto, fragmentos topológicamente iguales pueden ser diferenciables por el valor de su Índice del Estado Refractotopológico Total. Este principio constituye la base teórica del nuevo lenguaje descriptor [16].

Las principales características y ventajas de este nuevo índice es que considera la influencia de los átomos de hidrógeno y las contribuciones de estos al valor intrínseco de los átomos pesados a diferencia de los demás índices que no tienen en cuenta la influencia de los átomos de hidrógeno. Esto refleja la capacidad potencial de interacción total del grupo con una supuesta proteína ligando y esto es importante,

---

<sup>14</sup> **Refractividad atómica:** es el valor de la refractividad de un átomo.

<sup>15</sup> **Distancia euclidiana:** es la raíz cuadrada de la suma de las diferencias cuadradas en los valores para cada variable.

no solamente para el valor intrínseco del átomo al cual está directamente enlazado, sino para los otros grupos presentes en la molécula [1].

### 2.2.2 Reglas gramaticales del lenguaje.

Las reglas gramaticales son el conjunto de definiciones que se establecen para regir la estructura del lenguaje, así como la especificación de los CDs y los CUs entre estos. Estas reglas están estrechamente relacionadas con características y propiedades químicas de compuestos orgánicos y para la definición de las mismas se tuvo en cuenta el análisis y criterio de especialistas químicos.

Se definen como CDs:

- Ciclos o anillos, de 3 a 12 átomos.
- Clúster de orden 3 y 4.
- Heteroátomos<sup>16</sup>.
- Metilo.
- Metileno.
- Metino.

Se define como CU al conjunto de átomos que relacionan a dos CDs y que en su composición no contiene a otro CD.

Se define como fragmento a dos CDs y el CU que los une, varios fragmentos pueden estar conformados por los mismo CDs pero unidos con diferentes CU.

Los CD y CU están ponderados por el Índice del Estado Refractotopológico Total.

---

<sup>16</sup> Heteroátomo: Átomos distintos del Carbono e Hidrógeno.



### 2.3 Codificación de Centros Descriptores y Caminos de Unión.

La codificación se realiza a partir de un fichero de extensión .mol que contiene información referente a la molécula. A este fichero se le incorpora el valor del Índice del Estado Refractotopológico Total obtenido de la base de datos del proyecto. En el proceso de codificación solo se toma de este fichero los símbolos de los átomos, la matriz de conectividad y el valor del Índice del Estado Refractotopológico Total.

El lenguaje da un único código a cada CD y CU en dependencia de la molécula en que se encuentren. En aquellos compuestos químicos simétricos puede darse el caso de que el código se repita. Para la codificación se hace uso de una secuencia de caracteres imprimibles de longitud finita y sin espacios. Esta representará el código que identifique a los CDs y CUs. Este código está compuesto por n caracteres, el valor de n está determinado por el tipo de CD o el CU.

En caso de que sean CDs de tipo: anillos, clúster o heteroátomos, n es igual a 12, de estos se destinan dos para identificar el tipo de CD, dos para características propias y los ocho restantes para el valor del Índice del Estado Refractotopológico Total, este es un número real comprendido entre -99.9999 y 999.9999. En caso de que sea un CD de tipo: metilo, metileno y metino o un CU el valor de n es 11, debido a que en estos casos el número de caracteres destinados a identificar el tipo de CD es uno, c para el caso de los CU y m para los grupos estructurales metilo, metino y metileno.

Se destinan dos caracteres para identificar el tipo de CD: anillos, clúster y heteroátomos debido a que:

- Los anillos no exceden de dos cifras, con un valor 12 como máximo, ya que este es el valor límite de átomos que tendrán los anillos en esta redefinición del lenguaje.
- En caso de los clústeres se les ponen dos caracteres en el primer identificador, debido a que este lenguaje pueda ser extendido a otras investigaciones que sí necesiten de estos caracteres.

En caso de valores menores que 10, se completarán los dos caracteres destinados a identificar el tipo de CD con el valor cero y el número que lo identifica, esto ocurre con los clúster, heteroátomos y gran parte de los anillos.

De igual manera se destinaron dos caracteres para reflejar características propias del CD, por la naturaleza de la información que estos caracteres representan, solo son suficientes con esta cantidad de caracteres.

La Tabla # 2.1 muestra una serie de códigos válidos y códigos no válidos para mayor comprensión.

Códigos	Válido	no Válido	Justificación
0001012.3476	x		
1203015.4500	x		
0212-04.1267	x		
000112.3476		x	Falta un caracter antes del valor 12
073015.4500		x	Falta un caracter antes del valor 3
212-04.1267		x	Falta el primer caracter.

**Tabla # 2.1. Comparación de caracteres.**

Cuando se crea la codificación de los CDs y los CUs se crea también una cadena de caracteres imprimibles de longitud finita y sin espacio. Esta depende en gran medida de la composición estructural del CD, contiene información topológica que será utilizada para decodificar la molécula. Para conformar esta cadena a cada átomo y enlace se le da una codificación, los heteroátomos mantienen su codificación y al carbono se le da valor 1 o 01 porque son dos caracteres. Los enlaces toman la codificación: s, d, t, a en dependencia de la naturaleza del enlace:

- Simple- s.
- Doble- d.
- Triple- t.
- Aromáticos- a.

Para esta codificación se destinan cuatro caracteres, dos para la codificación del átomo y las dos restantes para la posición que este ocupa en el fichero .mol, haciendo estos un total de 4 caracteres que contienen la información del átomo. En esta codificación, se toma también en cuenta el criterio planteado en la codificación de CDs y CUs con respecto a valores menores de 10.

Por ejemplo la codificación de un átomo de carbono que esté en posición 1 del fichero .mol será:

**0101**

La forma de representar un enlace existente entre dos átomos será:

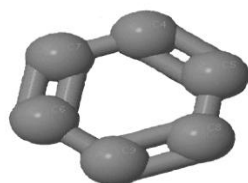
- Si existiera un enlace simple entre un átomo de carbono que se encuentre en la posición dos y otro átomo de nitrógeno que se encuentre en la posición tres, la cadena será:

**0102s0203.**

Como resultado del proceso de codificación se obtiene un fichero de extensión .Ide que es propio del lenguaje. Este fichero contiene toda la información referente a los CDs, CUs y fragmentos que puedan ser identificados en la molécula, así como información topológica de los mismos. Este fichero será explicado con detalle en el Capítulo 3: Resultados y Discusión.

### **2.3.1 Codificación del Centro Descriptor: Ciclo o Anillo.**

Un ciclo o anillo es un conjunto de átomos enlazados entre sí, donde el átomo inicial es igual al átomo final. La Figura # 2.1 muestra un ejemplo de anillo.



**Figura # 2.1. Anillo de 6 átomos de carbono.**

El código para representar a los ciclos o anillos se confecciona de la siguiente manera:

- El primer identificador del código indica la cantidad de elementos (átomos) por los que está conformado el anillo. Si este está compuesto por n elementos, entonces el primer identificador será: n, n podrá tener valores entre 3 y 12, ya que se tendrán en cuenta los anillos que contengan esta cantidad de átomos.

- El segundo identificador del código indica la cantidad de heteroátomos que está involucrada en la estructura del ciclo. Si el ciclo en su estructura consta de x heteroátomos, entonces dicho identificador será x, en caso contrario será 00.
- El tercer identificador será el Índice del Estado Refractotopológico Total (IRTT) asociado a dicho ciclo. Suponiendo que el ciclo tiene como IRTT el valor 12.34, entonces el tercer identificador será: 012.3400.

Atendiendo a los parámetros anteriormente mencionados y teniendo en cuenta como está constituido cada identificador el código para representar ciclos o anillos quedaría de la siguiente manera:

- Si al valor de n y de x son menores de 10 el código será: **0n0x012.3400**.
- Si al valor de n y de x son mayores de 9 el código será: **nx012.3400**.

Para crear la cadena de caracteres que proporciona información topológica de los anillos, se tienen en cuenta todos los átomos que lo componen, así como los enlaces entre ellos. Esta cadena comenzará con la información del átomo que se toma como partida en la conformación del anillo y continuará con la información de cómo y con quien está enlazado este átomo, así seguirá creciendo en longitud esta cadena por tantos átomos que compongan al anillo. La Figura # 2.2 muestra la información topológica codificada de un anillo de 6 átomos. En esta figura los elementos señalados con un cuadro representan la información de los átomos que componen al anillo, las flechas representan los enlaces y las letras(s, d, t, a), entre un átomo y otro, la manera en la que están conectados los átomos.

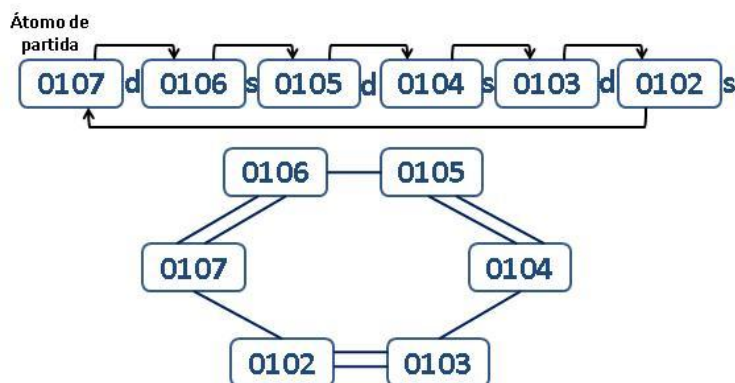


Figura # 2.2. Cadena de un anillo de 6 átomos.

### 2.3.2 Codificación del Centro Descriptor: Clúster.

Los clúster son agrupaciones de átomos donde se encuentra un elemento interconectado con más de uno. Este número de elementos conectados a este solo elemento determina el orden del clúster. Se tomaron como CDs todos aquellos clúster que tengan orden 3 y orden 4, es decir, que en su composición existan 3 o 4 átomos conectados a un solo átomo. En la Figura # 2.3 y la Figura # 2.4 se muestran un clúster de orden 3 y de orden 4 respectivamente.

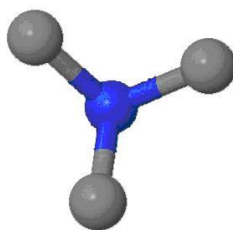


Figura # 2.3. Clúster de orden 3.

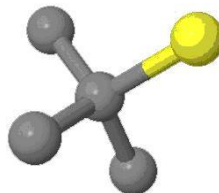


Figura # 2.4. Clúster de orden 4.

El código para representar a los clúster se confecciona de la siguiente manera:

- El primer identificador será el número 0 ó el número 1, en dependencia del orden del clúster.
  - El número 0 identifica a los clústeres de orden 3
  - El número 1 identifica a los clústeres de orden 4.
- El segundo identificador representa la cantidad de heteroátomos que tiene el clúster en su estructura. Supongamos que un clúster en su composición está integrado por n heteroátomos, el identificador estará representado entonces por la cantidad n.

- El tercer identificador será el Índice del Estado Refractotopológico Total asociado a dicho clúster. Suponiendo que el clúster tiene como Índice del Estado Refractotopológico Total el valor 12.34, entonces el tercer identificador será: 012.3400.

Atendiendo a los parámetros anteriormente mencionados y teniendo en cuenta como está constituido cada identificador, el código para representar clústeres quedaría de la siguiente manera:

- clúster de orden 3: **000n012.3400**
- clúster de orden 4: **010n012.3400**

En estos casos se mantuvieron dos caracteres para representar el primer identificador cuando solamente puede ser representado por un solo carácter, por la razón de que este lenguaje pueda ser extendido a otras investigaciones que si necesite de estos caracteres.

Para crea la cadena de caracteres que contiene la información topológica del clúster, se tienen en cuenta todos los átomos que lo componen, así como los enlaces entre ellos. Esta cadena comienza con la información del átomo que se encuentra en el centro del clúster y le sigue la información de todos los átomos que se enlazan a él y de cómo están enlazados. Esta cadena será tan larga como tantos átomos tenga el CD clúster, su mínima longitud será 19 y su máxima será 24 caracteres. La Figura # 2.5 muestra la información topológica de un clúster de orden 3.

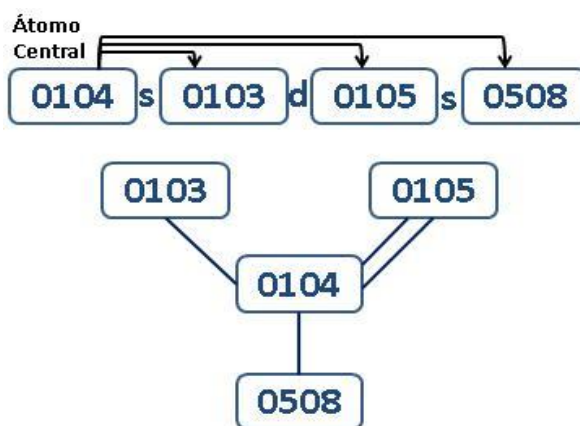


Figura # 2.5. Cadena de un clúster.

### 2.2.3 Codificación del Centro Descriptor: Heteroátomos.

En el lenguaje descriptor se tomaron en cuenta los siguientes heteroátomos:

- Flúor (F).
- Bromo (Br).
- Cloro (Cl).
- Yodo (I).
- Fósforo (P).
- Nitrógeno (N).
- Oxígeno (O).
- Azufre (S).
- Arsénico (As).
- Selenio (Se).
- Mercurio (Hg).
- Boro (B).
- Silicio (Si).
- Germanio (Ge).
- Antimonio (Sb).
- Telurio (Te).
- Polonio (Po).
- Astatato (At).
- Plomo (Pb).
- Estaño (Sn).

El código para representar a los heteroátomos se confecciona de la siguiente manera:

- El primer identificador va a estar representado por el número 02, este será el identificador en todo momento para identificar que estamos en presencia de un heteroátomo.
- El segundo identificador será un número entre 02 y 21 en dependencia del tipo de heteroátomo.
  - El número 02 representa al compuesto químico Nitrógeno.
  - El número 03 representa al compuesto químico Fósforo.
  - El número 04 representa al compuesto químico Azufre.
  - El número 05 representa al compuesto químico Oxígeno.
  - El número 06 representa al compuesto químico Flúor.
  - El número 07 representa al compuesto químico Cloro.
  - El número 08 representa al compuesto químico Yodo.
  - El número 09 representa al compuesto químico Bromo.
  - El número 10 representa al compuesto químico Arsénico.
  - El número 11 representa al compuesto químico Selenio.
  - El número 12 representa al compuesto químico Mercurio.
  - El número 13 representa al compuesto químico Boro.
  - El número 14 representa al compuesto químico Silicio.
  - El número 15 representa al compuesto químico Germanio.
  - El número 16 representa al compuesto químico Antimonio.
  - El número 17 representa al compuesto químico Telurio.
  - El número 18 representa al compuesto químico Polonio.
  - El número 19 representa al compuesto químico Astatio.
  - El número 20 representa al compuesto químico Plomo.
  - El número 21 representa al compuesto químico Estaño.



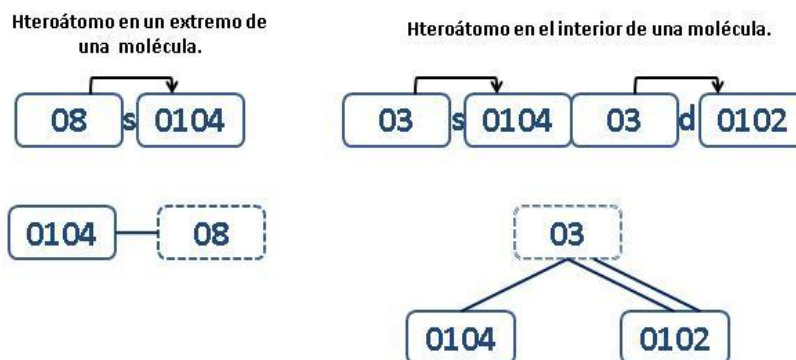
- El tercer identificador será el Índice del Estado Refractotopológico Total asociado a dicho heteroátomo. Suponiendo que el heteroátomo tiene como Índice del Estado Refractotopológico Total el valor 12.34, entonces el tercer identificador será: 012.3400.

El código para identificar a los heteroátomo será en dependencia del tipo de compuesto químico. Atendiendo a los parámetros anteriormente mencionados y teniendo en cuenta como está constituido cada identificador el código para representar heteroátomos quedaría de la siguiente manera:

- En caso de ser el Cloro será: **0207012.3400**
- En caso de ser el Flúor será: **0203012.3400**
- En caso de ser el Oxígeno será: **0205012.34.00**

En la cadena de caracteres que contiene la información topológica de los heteroátomos no se especifica el tipo de elemento, debido a que esta información está en la codificación del CD.

Un heteroátomo puede encontrarse en un extremo terminar de la molécula o en el interior de la misma. Si se encuentra en un extremo de la molécula, la cadena de caracteres con información topológica estará dada por la codificación del átomo, el tipo de enlace y el átomo que se encuentra en el otro extremo del enlace. En caso de que no se encuentre en un extremo de la molécula, la cadena estaría constituida por el heteroátomo, el tipo de enlace y uno de los átomos al que está conectado, luego el heteroátomo, el tipo de enlace y el otro átomo al que está conectado. La Figura # 2.6 muestra la información topológica de ambos casos.



**Figura # 2.6. Cadena de los Heteroátomos.**

### 2.3.4 Codificación del Metilo (CH<sub>3</sub>), Metileno (CH<sub>2</sub>), Metino (CH).

Los CDs metilo, metileno y metino se encuentra en extremos terminales de las moléculas, debido a que estos tipos de CDs están estructurados por un solo átomo de carbono enlazado con otro elemento cualquiera y los restantes enlaces son con hidrógenos. La Figura # 2.7 muestra la estructura de estos compuestos.

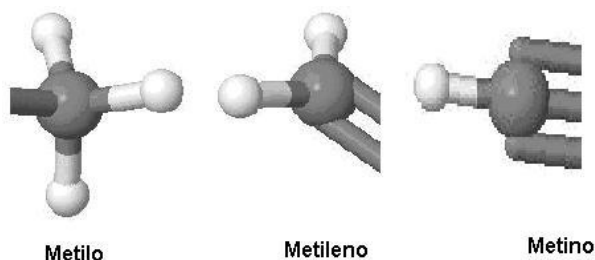


Figura # 2.7. Metilo, metileno y metino.

La codificación que se le da a este tipo de CD está compuesta por 3 identificadores los cuales son:

- El primer identificador es la letra m para determinar que es un metilo, metileno o metino.
- El segundo es la cantidad de átomo de hidrógenos que estos tienen enlazados.
  - Metilo tiene 3 átomos de hidrógenos -03.
  - Metileno 2 átomos de hidrógenos-02.
  - Metino 1 átomos de hidrógenos-01.
- El tercer identificador es el Índice del Estado Refractotopológico Total.

Suponiendo que el Índice del Estado Refractotopológico Total es igual a 0.6 y teniendo en cuenta como está constituido cada identificador, el código para representar a este tipo de CD quedaría de la siguiente manera:

- Metilo: **m03000.6000.**
- Metileno: **m02000.6000.**
- Metino: **m01000.6000.**

La cadena de caracteres que contiene la información topológica de estos CDs comienza por el código del átomo de carbono, le sigue el tipo de enlace con que este se conecta y la codificación del átomo que no es hidrógeno al cual esta enlazado. La Figura # 2.8 muestra la información topológica de estos CDs.

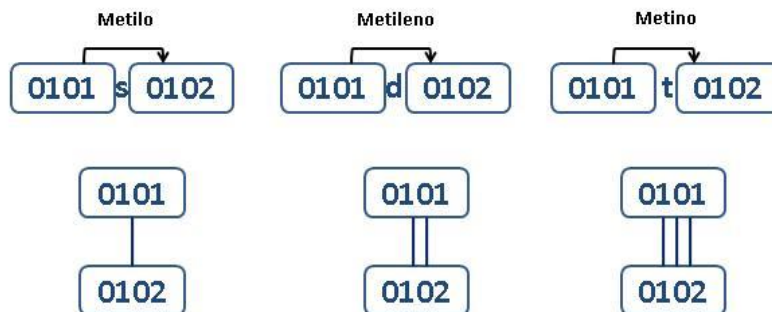


Figura # 2.8. Cadena del metilo, metileno y el metino.

### 2.3.5 Codificación del Camino de Unión.

Los CU no son más que cadenas de átomos que unen a los CDs formando los fragmentos. La Figura # 2.9 muestra un ejemplo de un CU entre dos CDs.



Figura # 2.9. Camino de Unión entre dos CD.

Los CU están identificados por tres identificadores:

- El primer identificador será la letra c.
- El segundo identificador representa la longitud del camino. Suponiendo que un camino tiene de longitud  $n$ , entonces el segundo identificador será  $n$ , si el valor de  $n$  es menor que 10 antes del número se pone 0 sino el número que tiene dos caracteres.

- El tercer identificador será el Índice del Estado Refractotopológico Total, asociado a dicho CU. Suponiendo que el camino tiene como Índice del Estado Refractotopológico Total el valor 12.34, entonces el tercer identificador será: 012.3400.

Atendiendo a los parámetros anteriormente mencionados el código para identificar a un CU será:

- Si al valor de n y de x son menores que 10 el código será: **c0n012.3400**.
- Si al valor de n y de x son mayores que 9 el código será: **cn012.3400**.

La cadena de caracteres que contienen información de los CUs, está conformada por todos los átomos que contiene este, comienza por un átomo que se toma como partida, hasta llegar al átomo donde termina, manteniendo la información de cómo son los enlaces entre cada átomo. La Figura # 2.10 muestra la información topológica de un CU.

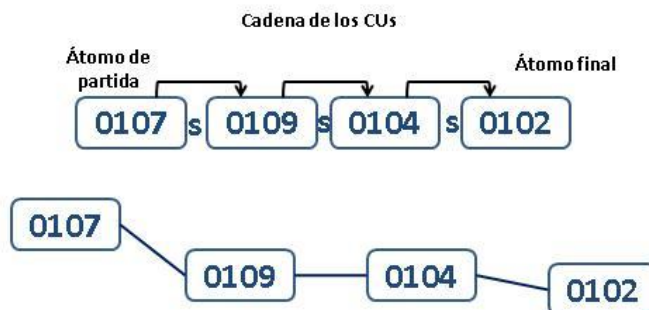
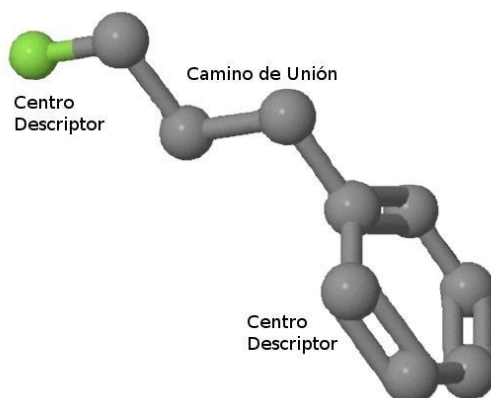


Figura # 2.10. Cadena de un Camino de Unión.

## 2.4 Codificación de un Fragmento.

Los fragmentos están conformados como por dos CDs y el CU que los une. La Figura # 2.11 ilustra como está constituido un fragmento.



**Figura # 2.11. Composición de un Fragmento.**

Para la creación del código de un fragmento es necesario tener las codificaciones de los CDs y el CU que lo componen. Debido a que este código está conformado por la concatenación de las codificaciones de los CDs y la del CU.

Ejemplo: Codificación del fragmentos de la Figura # 2.11.

- Se tiene como primer CD un heteroátomo, suponiendo que este fuese un Cloro (Cl) y su índice fuese de 1.5, la codificación de este sería (0207001.5000) según las reglas planteada anteriormente.
- La codificación del CU sería de la siguiente forma (c03000.6000) según las reglas planteadas anteriormente, suponiendo que el índice fuese de 0.6.
- Se tiene como segundo CD un anillo, suponiendo que su índice fuese de 21.4 la codificación de este sería (0600021.4000), según las reglas planteadas anteriormente.

La codificación del fragmento se conformaría de la siguiente forma:

**CD1 (0207001.5000) + CU (c03000.6000) + CD2 (0600021.4000)**

Codificación final para este fragmento: **0207001.5000c03000.60000600021.4000.**

## 2.5 Decodificar al fichero .mol.

La decodificación se realiza a partir del fichero .dle creado por el propio lenguaje, el cual contiene información de cómo están conformados los CDs , CU, y de la molécula en su totalidad.

Este proceso de decodificación consiste en ir leyendo del fichero .Ide todas las cadenas de caracteres que contienen la información topológica y generar un fichero .mol. En el fichero .Ide es donde se encuentra la información sobre los átomos: el símbolo de estos, como están conectados y con quién.

El fichero .mol no se genera en su formato original, puesto este contiene información de la matriz de conectividad de la molécula, de las coordenadas en tres dimensiones de los átomos y los símbolos de todos los átomos de la molécula. De estas atributos cuando se decodifica solo se obtienen la matriz de conectividad y el símbolo de lo átomos que son diferentes de los hidrógenos, debido a que es un lenguaje desprovisto de hidrógenos. El resto de la información, es decir los valores de las coordenadas para graficar no se obtienen dado que en esta redefinición del lenguaje no se tienen en cuenta, y por tanto no se encuentra reflejada esta información en el fichero .Ide.

Por tal motivo el proceso de decodificación solo genera un fichero .mol sin coordenadas, es decir, coordenadas en cero y como alternativa, el fichero generado es pasado por un convertidor para darle las coordenadas en dos dimensiones. La Figura # 2.12 muestra un fichero .mol sin coordenadas y la Figura # 2.13 muestra al mismo fichero pero con coordenadas en dos dimensiones.

```

4 3
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 Cl 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 Cl 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 1
1 4 1
2 3 1
M END

```

Figura # 2.12. Fichero .mol sin coordenadas.

```

4 3
  0.0000  0.0000  0.0000 C  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0.5000  0.8666  0.0000 C  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1.5000  0.8666  0.0000 cL 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0.5000 -0.8666  0.0000 cL 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 1
1 4 1
2 3 1
M END

```

Figura # 2.13. Fichero .mol con coordenadas.

## 2.6 Conclusiones.

Para el desarrollo del lenguaje descriptor se seleccionó la tecnología para su desarrollo, siempre y cuando cumpliera con un requisito muy importante, que fueran software libre o sino que su uso estuviera legalmente permitido los software que fueron seleccionados se utilizan además por todas las ventajas anteriormente expuestas por que se utilizan en el proyecto. Por lo que la selección de las herramientas quedó de la siguiente forma:

- Como lenguaje representativo se escogió UML porque brinda una visión gráfica de cómo debe estar estructurado el software.
- El lenguaje de programación escogido fue Java por su robustez, seguridad y por ser multiplataforma.
- El entorno de desarrollo escogido fue Eclipse por ser una herramienta muy potente para Java y de libre distribución.
- El Visual Paradigm como herramienta CASE para realizar el modelado del software, utilizando UML, se integra muy bien con el Eclipse y brinda la posibilidad de generar código en el lenguaje de programación Java.

También se presentó una amplia descripción de las reglas gramaticales del “LDEQODF”, las cuales describen como se le dará una codificación y decodificación a cada CD, CU y los fragmentos. Se adicionaron nuevos CDs, que en la primera definición realizada no se tomaron en cuenta, como por ejemplo:

- El metilo (CH<sub>3</sub>), metileno (CH<sub>2</sub>) y metino (CH).
- Nuevos heteroátomos, incluyendo los halógenos, pues anteriormente solo se tomaban en cuenta estos últimos, despreciando la información que brindaban los restantes heteroátomos.



# RESULTADOS Y DISCUSIÓN



El presente capítulo trata sobre los resultados obtenidos de las modificaciones realizadas al Lenguaje Descriptor de Estructuras Químicas Orientadas al Diseño de Fármacos, y a la vez se discute sobre la importancia y relevancia de los mismos. Se presenta además un estudio estadístico sobre el comportamiento del valor del índice en CDs.

## 3.1 Modelo conceptual.

Un modelo conceptual brinda una idea global de cómo se plantea un problema, ya sean elementos, situaciones y acontecimientos que son importantes en este. En la Figura # 3.1 se muestra el modelo conceptual del lenguaje descriptor de estructuras químicas.

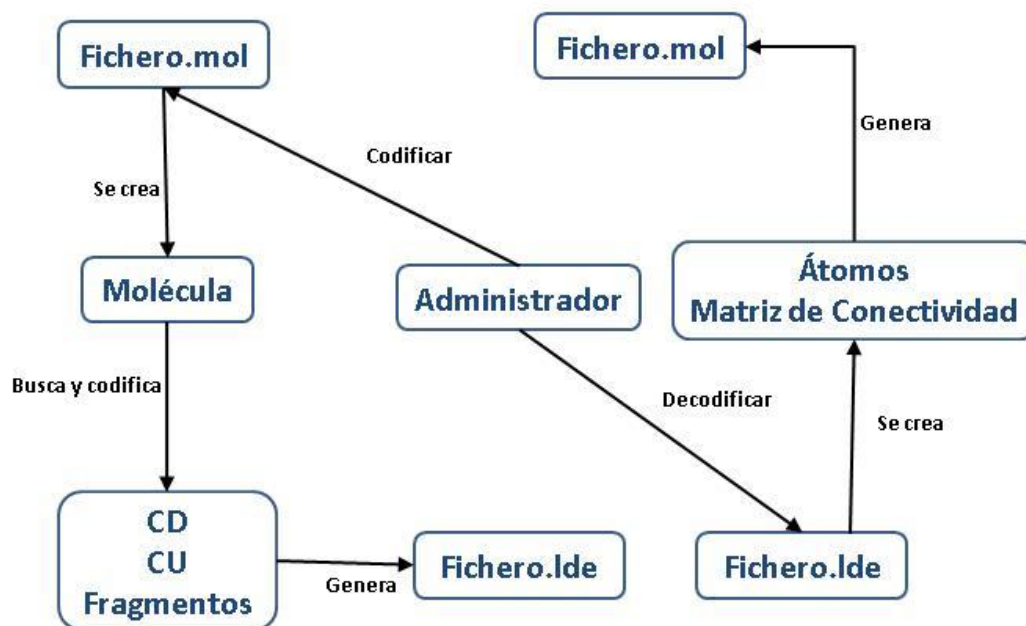


Figura # 3.1. Modelo conceptual.

El modelo plantea las opciones que puede realizar el administrador, las cuales son codificar y decodificar estructuras químicas representadas en ficheros. Si se desea codificar, el sistema leerá ficheros con extensiones .mol, a partir de los cuales se crea la molécula, se identifican CDs y CU, conformando así fragmentos. Posteriormente se genera un fichero con extensión .lde el cual contiene toda la información de la molécula codificada. Si la opción es decodificar, el sistema leerá el fichero .lde generado por la codificación creando la información necesaria para crear el fichero .mol.

### 3.2 Diseño de clases.

En las Figura # 3.2 y Figura # 3.3 evidencias los diagramas de clase que se realizaron para implementar los algoritmos de codificar y decodificar estructuras químicas.

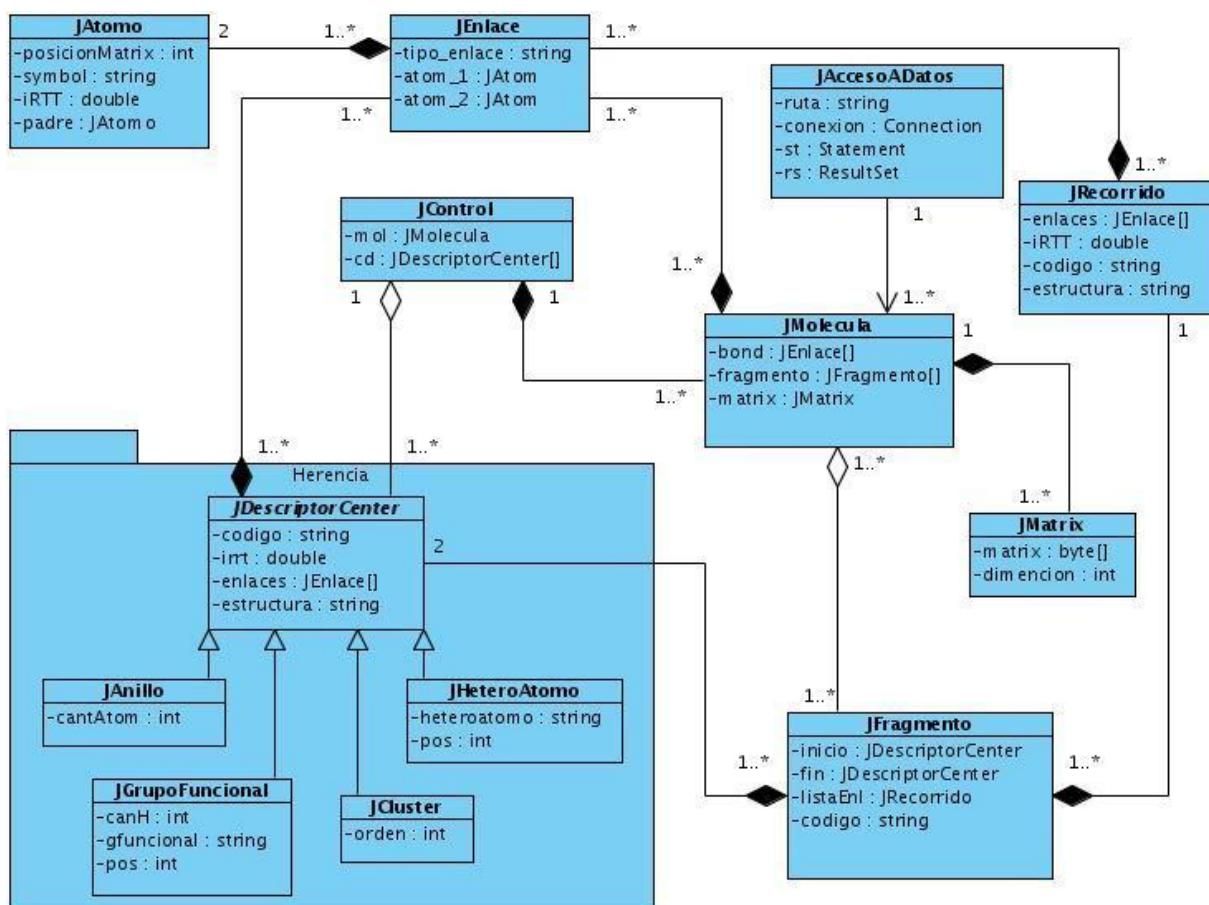


Figura # 3.2. Diagrama de Clases para codificar Estructuras Químicas.

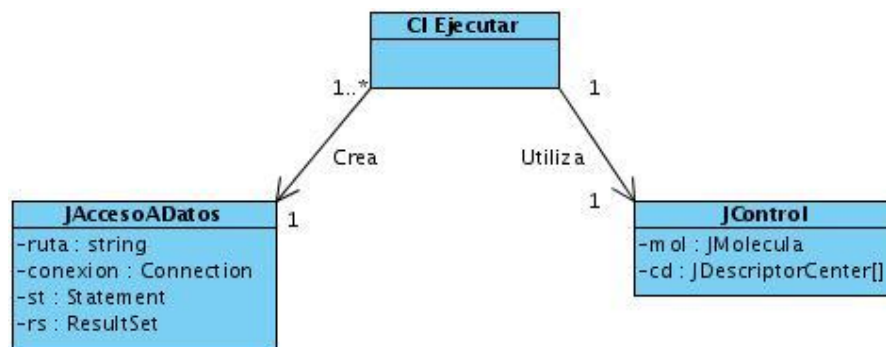


Figura # 3.3. Diagrama de Clases para decodificar a Ficheros .mol.

### 3.3 Descripción de los Algoritmos del Lenguaje.

Para la realización del presente lenguaje descriptor, fue necesaria la implementación de algoritmos de búsqueda para identificar CDs y CU. Para ello se tomó la molécula como un grafo químico, tomando como vértices a los átomos y como aristas a los enlaces que los unen. De este grafo es posible obtener la matriz de adyacencia, la cual es una matriz cuadrada  $m$  de tamaño  $n$ , donde  $n$  es el número de vértices.

#### 3.3.1 Búsqueda de CD Anillo.

Para la búsqueda de ciclos o anillos se tuvo en cuenta que un ciclo es un camino o sucesión de vértices (átomos) adyacentes, donde no se recorre más de dos veces la misma arista, y donde se regresa al vértice (átomo) inicial. El algoritmo que se utilizó para esta búsqueda fue el algoritmo búsqueda a lo ancho utilizado en la teoría de grafos<sup>17</sup>, que fue adaptado para identificar este tipo de estructuras, incluyéndole la particularidad de que cada elemento explorado se marca como visitado, imposibilitando así que este fuese visitado nuevamente y evitando que la búsqueda se mantenga en un ciclo infinito. La búsqueda termina cuando todos los elementos ya han sido explorados o que el número de vértices (átomos) sea mayor que

<sup>17</sup> Teoría de grafos: estudia las propiedades de los grafos, que son colecciones de objetos llamados vértices (o nodos) conectados por líneas llamadas aristas (o arcos) que pueden tener orientación (dirección asignada).

12 o cuando el elemento visitado sea igual al elemento inicial, en este caso se crea el anillo, siempre y cuando no exista uno igual a él.

### **3.3.2 Búsqueda de CD Clúster.**

La búsquedas de los clúster se realiza a partir de la matriz de adyacencia del grafo químico, es posible identificar sin problema alguna la presencia de CDs de tipo clúster, solo basta con recorrer la matriz y verificar que la cantidad de vértices (átomos) conectados a uno en específico es 3 o 4 siempre y cuando los vértices (átomos) sean diferentes de hidrógenos.

### **3.3.3 Búsqueda de CD Heteroátomo.**

Para la búsqueda de heteroátomos, se recorre la lista de vértices (átomos) del grafo químico y por cada uno se verifica que sea uno de los heteroátomos definidos como CD en las definiciones del propio lenguaje. Si este cumple con el criterio entonces se crea el heteroátomo.

### **3.3.4 Búsqueda de los CD metilo, metileno y metino.**

En la identificación de los CDs: metilo, metileno y metino también se recorre la lista de vértices (átomos) del grafo, donde solo se tienen en cuenta los átomos de carbono, ya que este tipo de estructura solo contiene en su composición un solo átomo de carbono, además de que tiene que cumplir que se encuentre en un extremo terminar del grafo y que no esté incluido en la composición de otro CD.

### **3.3.5 Búsqueda de CU.**

En la búsqueda de CU, se tomo la filosofía de la búsqueda de ciclos o anillos, pero teniendo como diferencia de que en este tipo de búsqueda el vértice (átomo) inicial no es igual al vértice (átomo) final, sino que hay una lista de vértices (átomos) finales a los cuales se pretende llegar para conformar el camino. Esta lista de vértices (átomos) corresponde a los vértices (átomos) extremos de un CD.

## **3.4 Algoritmos del Lenguaje.**

En este epígrafe se evidencia algunos de los algoritmos desarrollados del lenguaje. Para ver más algoritmos ver anexos 1.

```

private void buscarHeteroAtom(){
    ArrayList<JAtomo> atom=mol.obtenerAtomos();
    int iterar=atom.size();
    for(int i=0;i<iterar;i++){
        JAtomo temp=atom.get(i);
        String sT=temp.getSymbol();
        if(!sT.equals("C")) {
            ArrayList<JEnlace> aux=mol.enlacesCon(temp);
            int pT=temp.getPosicionMatrix();
            JCentroDescriptor h=new JHeteroAtomo(sT,pT,aux);
            h.calculoIRTT();
            h.crearCodigo();
            h.crearEstructura();
            adicionarCentroDescriptor(h);
        }
    }
}

```

Figura # 3.4. Algoritmo para buscar heteroátomos.

```

private void buscarGrupoFuncional(ArrayList<JAtomo> list){
    ArrayList<JEnlace> aux=new ArrayList<JEnlace>();
    int iterar=list.size();
    for(int i=0;i<iterar;i++){
        JAtomo temp=list.get(i);
        String sT=temp.getSymbol();
        int pT=temp.getPosicionMatrix();
        boolean extremo=mol.esExtremo(temp);
        if(sT.equals("C") && extremo){
            aux=mol.enlacesCon(temp);
            int iterarl=aux.size();
            for(int j=0;j<iterarl;j++){
                JEnlace bond=aux.get(j);
                String bTE=bond.getTipo_enlace();
                JCentroDescriptor grup=null;
                if(bTE=="simple")
                    grup=new JGrupFuncional(3,sT,pT,aux);
                else if(bTE=="doble")
                    grup=new JGrupFuncional(2,sT,pT,aux);
                else if(bTE=="triple")
                    grup=new JGrupFuncional(1,sT,pT,aux);
                grup.crearCodigo();
                grup.crearEstructura();
                adicionarCentroDescriptor(grup);
            }
        }
    }
}

```

Figura # 3.5. Algoritmo para buscar Grupo Funcional.

### 3.5 Fichero de salida del lenguaje descriptor.

Como resultado de la codificación se obtiene un fichero con una extensión del propio lenguaje la cual es (.lde), este fichero tiene un formato determinado el cual es estándar para todas los ficheros .mol que se codifiquen con este lenguaje. La escritura en este fichero se realiza cuando ya han sido identificados y codificados todos los CDs, CUs y fragmentos de la molécula. Luego se escribe en el fichero una línea para cada uno de ellos, esta línea contiene la codificación y como están estructurados, separados por dos espacios. En el fichero de salida se pueden identificar cinco partes importantes, para mayor comprensión se muestra en la Figura # 3.6 la molécula de la hidroxipiridin-2.tiona y su fichero de salida correspondiente al proceso de codificación con todas sus secciones identificadas en él, en la Figura # 3.7.

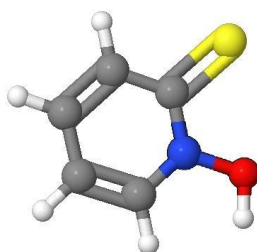


Figura # 3.6. Molécula del hidroxipiridin-2.tiona.

<b>Encabezado.</b>	{	006007		
<b>Centros Descriptores.</b>	{	0002022.5107	0103s0102d0404s0205	}
		0002008.0509	0205s0103s0106s0508	
		0601021.2578	0107d0106s0205s0103s0102d0101s	
		0204016.1240	04d0103	
		0202001.2904	05s010305s010605s0508	
		0205001.2423	08s0205	
		c00002.5327	0205s0508	
<b>Caminos de Unión.</b>	{	c00016.4291	0103d0404	}
		c00016.4291	0103d0404	
		c00002.5327	0205s0508	
		c05037.3818	0404d0103s0102d0101s0107d0106s0205	
		c01017.7195	0404d0103s0205	
		c00002.5327	0205s0508	
		<b>Fragmentos.</b>	{	
0002008.0509c00016.42910204016.1240				
0601021.2578c00016.42910204016.1240				
0601021.2578c00002.53270205001.2423				
0204016.1240c05037.38180202001.2904				
0204016.1240c01017.71950202001.2904				
0202001.2904c00002.53270205001.2423				

Figura # 3.7. Estructura de un Fichero del "LDEQODF" de la codificación del hidroxipiridin-2.tiona.

- Encabezado.

El encabezado del fichero de salida brinda información sobre la cantidad de CDs y CUs que hay presentes en la molécula. Esta información es la primera línea del fichero. A continuación un ejemplo.

**aaabbb**

**Figura # 3.8. Encabezado del Fichero .Ide.**

- aaa= número de CDs identificados en la molécula.
- bbb= número de CUs identificados en la molécula.

- Bloque de CD.

En el bloque de los CDs se encuentran los identificadores de los CDs identificados en la molécula. Los CDs estarán ordenados de la siguiente forma, primeramente se encontrarán los clústeres ya sean de orden 3 o de orden 4, seguidos de los anillos, luego los heteroátomos y finalmente los metilo, metileno, y los metino.

- Bloque de CU.

En el bloque de los CU se encuentran todos lo CU codificado con las reglas gramaticales del lenguaje según las reglas planteadas en el capítulo 2.

- Bloque de Fragmentos.

En el bloque de los fragmentos se encuentran todos los fragmentos que existen en la molécula.

- Cadena de Caracteres.

Contiene la información topológica de cómo están enlazados los átomos. Estas se crean cuando se codifica la molécula. Cada átomo tiene un valor asociado el cual lo identifica, esta información es muy valiosa en el momento en que se quiera decodificar a ficheros con extensión .mol, para mayor información ver como se conforman estas cadena de caracteres en el capítulo 2.

### 3.6 Pruebas.

El lenguaje deberá pasar por distintas fases de pruebas antes de que pueda ser utilizado, estas pruebas presentan como principal objetivo evaluar que los resultados obtenidos estén de acuerdo a las reglas gramaticales planteadas en el capítulo 2. Las pruebas que se le aplicaron al software fueron pruebas para verificar su funcionamiento, como codifica y decodifica las moléculas.

#### 3.6.1 Pruebas de Codificación.

##### Primera Prueba.

En la primera se codificaron 100 ficheros .mol prueba, en la cual se detectaron errores como:

- El IRTT de los CU se calculó de forma entera, por lo que no se obtuvieron resultados exactos, los valores fueron redondeados.
- Se detectó un átomo que estaba presente en moléculas orgánicas y no estaba contemplado dentro de las reglas del lenguaje descriptor el cual se incluyó dentro de las reglas del lenguaje.

##### Segunda Prueba.

En la segunda prueba se codificaron los 100 ficheros .mol nuevamente, pero con los errores anteriores arreglados en el sistema. En esta prueba se erradicaron los errores que se detectaron en la primera prueba. Todas las moléculas fueron codificadas según las reglas del lenguaje.

##### Tercera Prueba.

En la tercera prueba luego de hacer algunos cambios y arreglos detectados en el código del programa se codificaron 296 ficheros .mol. De las 296 moléculas codificadas se tomaron y revisaron 10 ficheros aleatorios y se encontró el siguiente error de codificación que anteriormente no se habían detectado.

- Se detectaron errores como problemas en la codificación de los clúster de orden 3 y 4, pues al calcular la cantidad de heteroátomos se calculaba un átomo de más.

##### Cuarta Prueba.

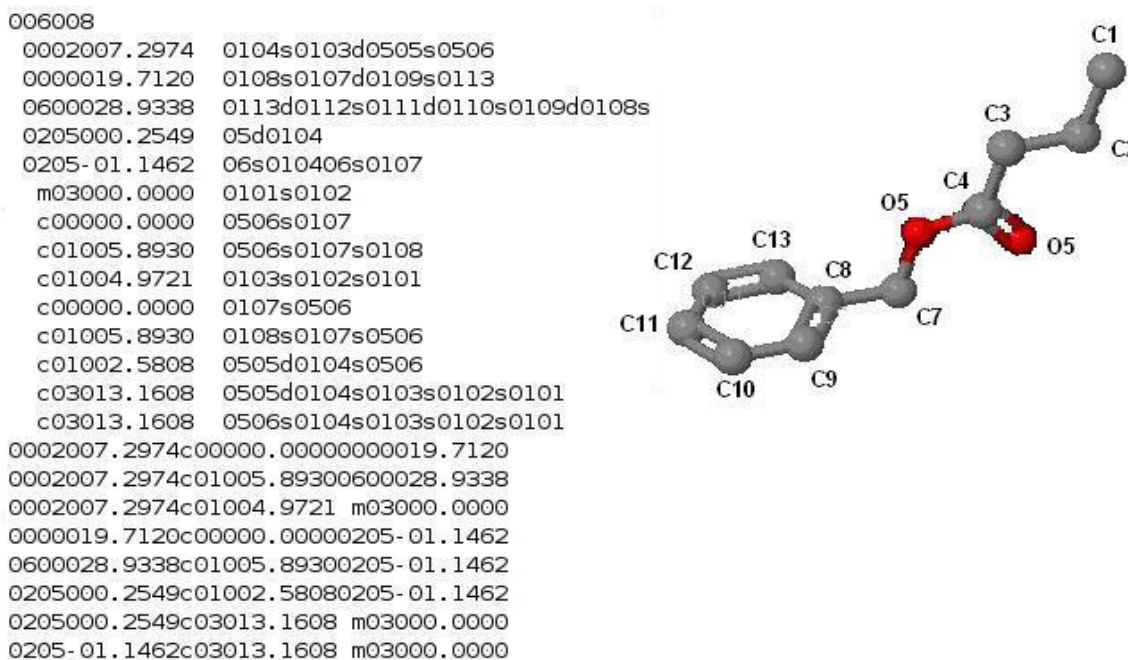
En esta prueba se erradicaron los errores encontrados en las pruebas anteriores, se codificaron los 296 ficheros nuevamente. Es importante aclarar que los errores gramaticales planteadas en las prueba 3 fueron erradicado.



**Quinta Prueba.**

Se codificaron 417 moléculas, se detectaron nuevos átomos que no se tenían en cuenta los cuales fueron el selenio, mercurio, el boro y otros, los cuales fueron incorporados a las gramáticas del lenguaje en el capítulo 2. Se detectó un error en moléculas que en su composición no tienen átomos de hidrógenos, pues en el momento de crear la molécula entraban en un ciclo infinito. Estos errores fueron erradicados posteriormente a esta prueba se codificaron 1367 moléculas y fueron codificadas correctamente según las reglas gramaticales del lenguaje.

A continuación en las Figura # 3.9 y Figura # 3.10 se muestran imágenes donde se puede observar la codificación dada por el lenguaje descriptor y la molécula a la que este pertenece, aquí se puede apreciar que el lenguaje cumple con las reglas planteadas en su sintaxis. La cadena que se encuentra después de los CD y los CU es información sobre cómo están compuestos y enlazados los átomos en los CDs y CU si desea ver más moléculas codificadas ver anexos # 2.



**Figura # 3.9. Codificación y Molécula del benzyl butanoate.**

```

006007
0002022.5107 0103s0102d0404s0205
0002008.0509 0205s0103s0106s0508
0601021.2578 0107d0106s0205s0103s0102d0101s
0204016.1240 04d0103
0202001.2904 05s010305s010605s0508
0205001.2423 08s0205
c00000.0000 0205s0508
c00000.0000 0103d0404
c00000.0000 0103d0404
c00000.0000 0205s0508
c05019.9674 0404d0103s0102d0101s0107d0106s0205
c01000.3052 0404d0103s0205
c00000.0000 0205s0508
0002022.5107c00000.00000205001.2423
0002008.0509c00000.00000204016.1240
0601021.2578c00000.00000204016.1240
0601021.2578c00000.00000205001.2423
0204016.1240c05019.96740202001.2904
0204016.1240c01000.30520202001.2904
0202001.2904c00000.00000205001.2423
    
```

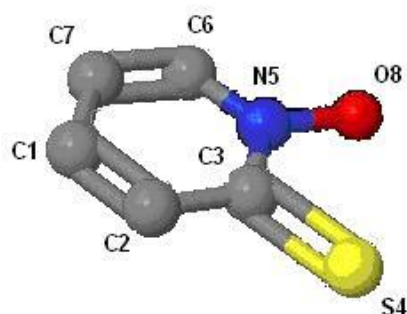


Figura # 3.10. Codificación y Molécula del 1-hydroxypyridine-2-thione.

### 3.6.2 Pruebas de Decodificación.

#### Primera Prueba.

La primera prueba de decodificación se realizó con 20 ficheros generados por el lenguaje descriptor. En esta se detectó un error, el cual era un problema de tipo de datos, pues en la columna del fichero donde se guardan los símbolos de los átomos cuando se genera el fichero .mol se guardaba un tipo de dato char, es decir un solo carácter, no se tenían en cuenta cuando el símbolo del átomo estuviera compuesto por dos letras.

#### Segunda Prueba.

En la segunda prueba se erradicaron los problemas que salieron a relucir en la primera prueba, se decodificaron nuevamente los 20 ficheros generados por el lenguaje y no se detectaron ningún problema en ellos. Los ficheros se decodificaron y posteriormente pasaron por el convertidor, que le adiciona las coordenadas en dos dimensiones.

#### Pruebas de visualización.

## CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

Para realizar esta prueba se tomo el fichero .mol de la molécula: (1-nitro-2-phenyl-vinyl) benzene, que fue codificado y decodificado por el lenguaje. En la Figura # 3.11 se mostrará el fichero .mol antes de ser codificado y en la Figura # 3.12 el fichero .mol que se obtuvo de la decodificación del mismo fichero, el cual no contiene información de la dimensión z.

```
YMconverte03110816193D 1 1.00000 0.00000 0
CORINA 2.64 0044 29.07.2002
28 29 0 0 0 0 999 V2000
0.1560 1.3492 1.2662 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.1157 0.2771 0.3922 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.2757 -0.3919 0.0622 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2.4938 0.0156 0.6124 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2.5268 1.0993 1.4939 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.3586 1.7584 1.8150 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.7378 -0.6945 0.2650 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.7178 -2.0203 0.0555 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4.9728 -2.7416 -0.2222 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.1990 -2.0778 -0.1308 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7.3693 -2.7594 -0.3917 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7.3312 -4.0972 -0.7436 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.1205 -4.7611 -0.8360 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4.9425 -4.0940 -0.5724 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2.4351 -2.7580 0.1034 N 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.8792 -3.0807 -0.9311 O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.9357 -3.0421 1.1774 O 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-0.7550 1.8719 1.5178 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-0.8265 -0.0368 -0.0322 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.2427 -1.2289 -0.6195 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.4646 1.4196 1.9232 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.3825 2.5962 2.4961 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4.6655 -0.1482 0.1790 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.2302 -1.0336 0.1430 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8.3178 -2.2477 -0.3215 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8.2507 -4.6256 -0.9474 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.0984 -5.8051 -1.1115 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.9984 -4.6135 -0.6446 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 6 1 0 0 0 0 0
1 2 2 0 0 0 0 0
1 18 1 0 0 0 0 0
2 3 1 0 0 0 0 0
2 19 1 0 0 0 0 0
3 4 2 0 0 0 0 0
3 20 1 0 0 0 0 0
4 5 1 0 0 0 0 0
4 7 1 0 0 0 0 0
5 6 2 0 0 0 0 0
5 21 1 0 0 0 0 0
6 22 1 0 0 0 0 0
7 8 2 0 0 0 0 0
7 23 1 0 0 0 0 0
8 9 1 0 0 0 0 0
8 15 1 0 0 0 0 0
9 14 1 0 0 0 0 0
9 10 2 0 0 0 0 0
10 11 1 0 0 0 0 0
10 24 1 0 0 0 0 0
11 12 2 0 0 0 0 0
11 25 1 0 0 0 0 0
12 13 1 0 0 0 0 0
12 26 1 0 0 0 0 0
13 14 2 0 0 0 0 0
13 27 1 0 0 0 0 0
14 28 1 0 0 0 0 0
15 16 2 0 0 0 0 0
15 17 1 0 0 0 0 0
M CHG 2 15 1 17 -1
M END
$$$$
```

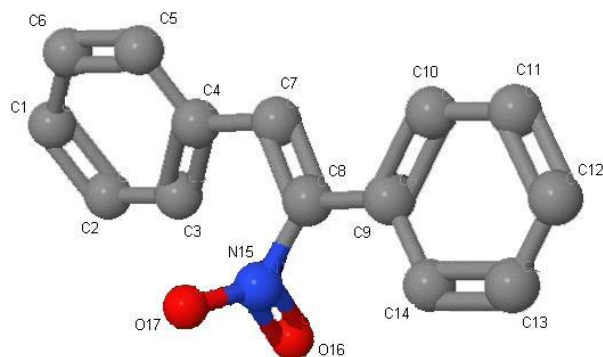
Figura # 3.11. Fichero .mol antes de codificar.

## CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

```
17 18
 1.0000  0.0000  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0.5000  0.8666  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-0.5000  0.8666  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-1.0000  0.0000  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-0.5000 -0.8666  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0.5000 -0.8666  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-2.0000  0.0000  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-2.5000  0.8666  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-3.5000  0.8666  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-4.0000  1.7333  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-5.0000  1.7333  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-5.5000  0.8666  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-5.0000  0.0000  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-4.0000  0.0000  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-2.0000  1.7333  0.0000 N  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-1.0000  1.7333  0.0000 O  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
-2.5000  2.6000  0.0000 O  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
1  2  2
1  6  1
2  3  1
3  4  2
4  5  1
4  7  1
5  6  2
7  8  2
8  9  1
8 15  1
9 10  2
9 14  1
10 11  1
11 12  2
12 13  1
13 14  2
15 16  2
15 17  1
M  END
```

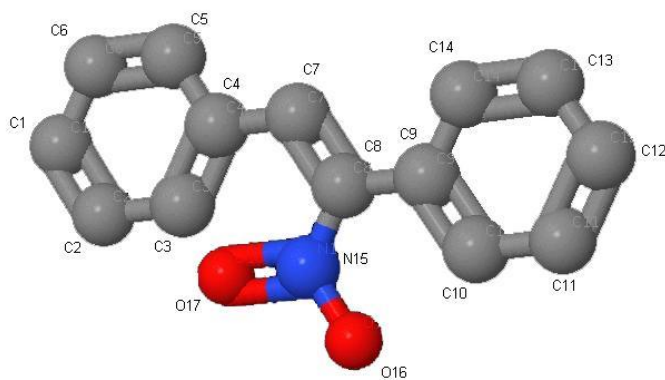
**Figura # 3.12. Fichero .mol después de decodificar.**

Posteriormente estos ficheros fueron visualizados con el visualizador la plataforma bioGRATO, y se obtuvieron las siguientes imágenes de la molécula.



**Figura 3.13. Visualización de un fichero .mol antes de codificar.**

La Figura # 3.11 es de un fichero .mol visualizado, con toda la información del fichero antes de ser codificado, esta molécula está graficada en tres dimensiones. Es importante aclarar que en esta imagen los átomos de hidrógenos están ocultos.



**Figura 3.14. Visualización de un fichero .mol después de ser decodificado.**

La Figura # 3.12 representa el fichero .mol después de ser codificado y decodificado por el lenguaje, esta molécula está graficada en dos dimensiones debido a que el lenguaje cuando decodifica un fichero luego es procesado por el convertidor que le atribuye información de las coordenadas en dos dimensiones.

Las representaciones visualizadas de las Figura # 3.13 y Figura # 3.14 son semejantes, con la única diferencia de que una está graficada en tres dimensiones y la otra en dos, por lo que se puede llegar a la conclusión de que la molécula final es la molécula inicial pero desprovista de hidrógenos.

### 3.7 Estudio Estadístico.

Por ser un lenguaje nuevo orientado al objetivo específico de utilizarlo en estudios de relación-estructura actividad, se vio la necesidad de realizar un análisis estadístico que permitiera afirmar que el fundamento teórico del mismo se correspondía con los resultados. Para ello se realizó un análisis preliminar sobre una muestra de 1367 compuestos tomados de manera totalmente aleatoria de la base de datos del proyecto y codificadas con el lenguaje descriptor.

El objetivo de este estudio es analizar el comportamiento del valor del Índice del Estado Refractotopológico Total de los CDs y demostrar, en una primera aproximación que:

- CDs de iguales topologías, difieren en el valor del índice y por lo tanto en su capacidad de interacción con otras moléculas.
- CDs con diferentes composición e igual topología, deben presentar similar comportamiento químico-físico en interacciones entre moléculas.
- CDs de diferentes topologías, deben presentar similar comportamiento químico-físico en interacciones entre moléculas.

Primeramente se realizó un estudio general teniendo en cuenta solo anillos de 6, 5, 4, y 3 átomos. La cantidad de anillos según la cantidad de átomos encontrados en la muestras se evidencian en la Tabla # 3.1.

	Anillos de 6.	Anillos de 5.	Anillos de 4.	Anillos de 3.	Total.
Cantidad	1350	236	6	15	1607

**Tabla # 3.1. Cantidad de anillos según la cantidad de átomos encontrados en la muestra.**

En la Figura # 3.15 se observa una distribución casi normal de las diferentes poblaciones. En todos los casos, los anillos, excepto los de 4 átomos, distribuyen su valor en todo el espacio de valores del Índice. Esto indica que esos anillos cambian su comportamiento en dependencia del entorno molecular. Es evidente también que las poblaciones de diferentes anillos se comportan, de acuerdo al valor del Índice del Estado Refractotopológico Total, de manera similar. Esto sugiere que en estudios experimentales

orientados al diseño de fármacos, se puede sustituir en un compuesto dado, un CD por otro y mantener la misma capacidad de interacción de este con otras moléculas.

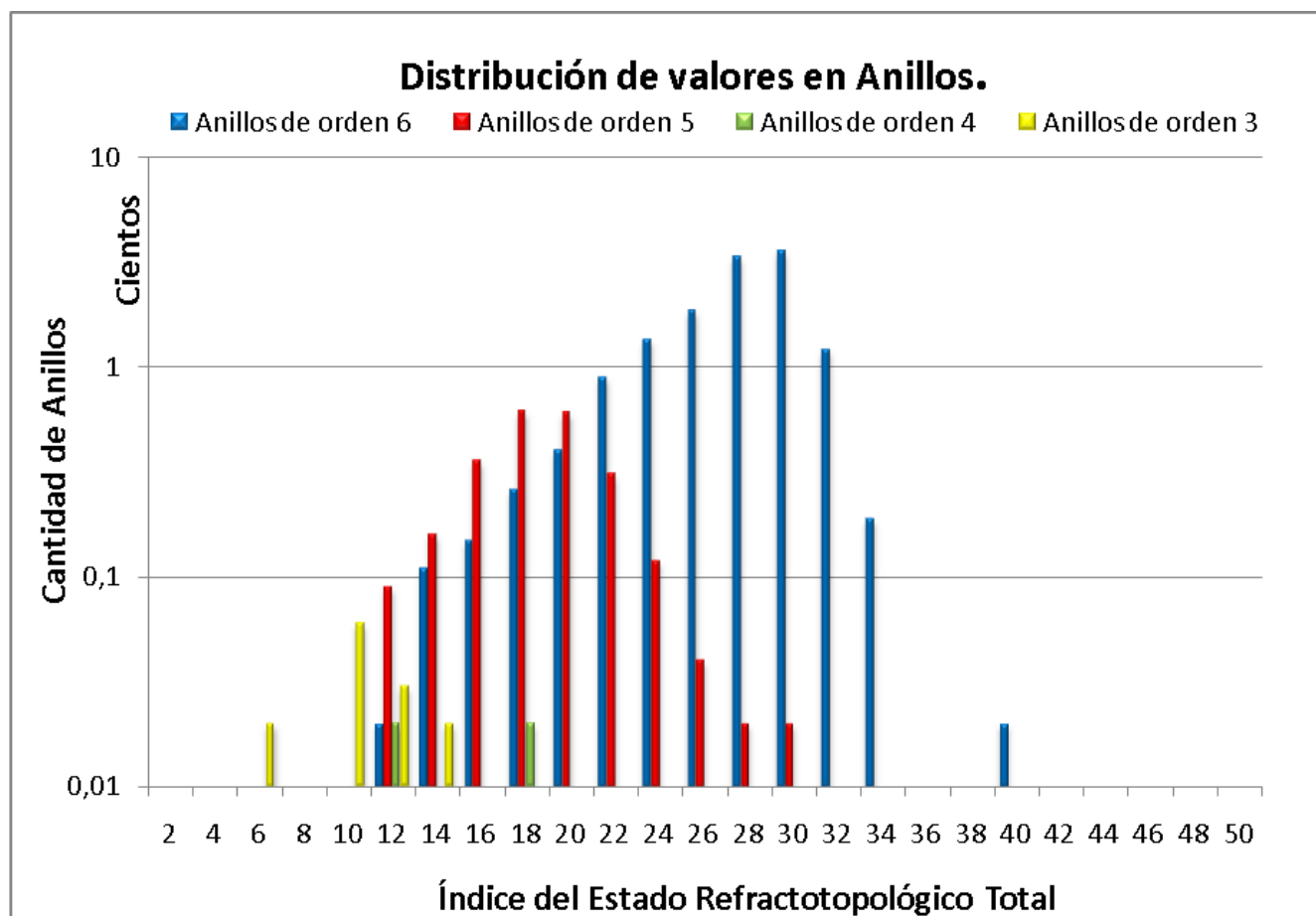


Figura # 3.15. Gráfica de la distribución del Índice de anillos de 6, 5, 4 y 3 átomos.

También se evidencia que CDs de diferentes topología y composición, presentan similares valores del Índice del Estado Refractotopológico Total, por lo que presentan iguales comportamientos según en el entorno en que se encuentren. Esto sugiere a los químicos que en estudios experimentales orientados al diseño de fármacos se puedan realizar sustituciones de CDs en un compuesto dado, de diferentes topologías por otros CDs, teniendo en cuenta que siempre se conserve el valor total del índice en el nuevo entorno.

### 3.7.1 Análisis de la muestra de Anillos de 6 átomos.

En la Tabla # 3.2 se muestran la distribución, en cuanto a composición, en anillos de 6 átomos. Esta tabla se organiza los datos descendentemente según la cantidad de heteroátomos que contengan los anillos.

	Anillos sin Heteroátomo	Anillos con 1 Heteroátomo	Anillos con 2 Heteroátomo	Anillos con 3 Heteroátomo
Cantidad	1104	135	94	17

Tabla # 3.2. Cantidad de anillos de 6 según la cantidad de Heteroátomos.

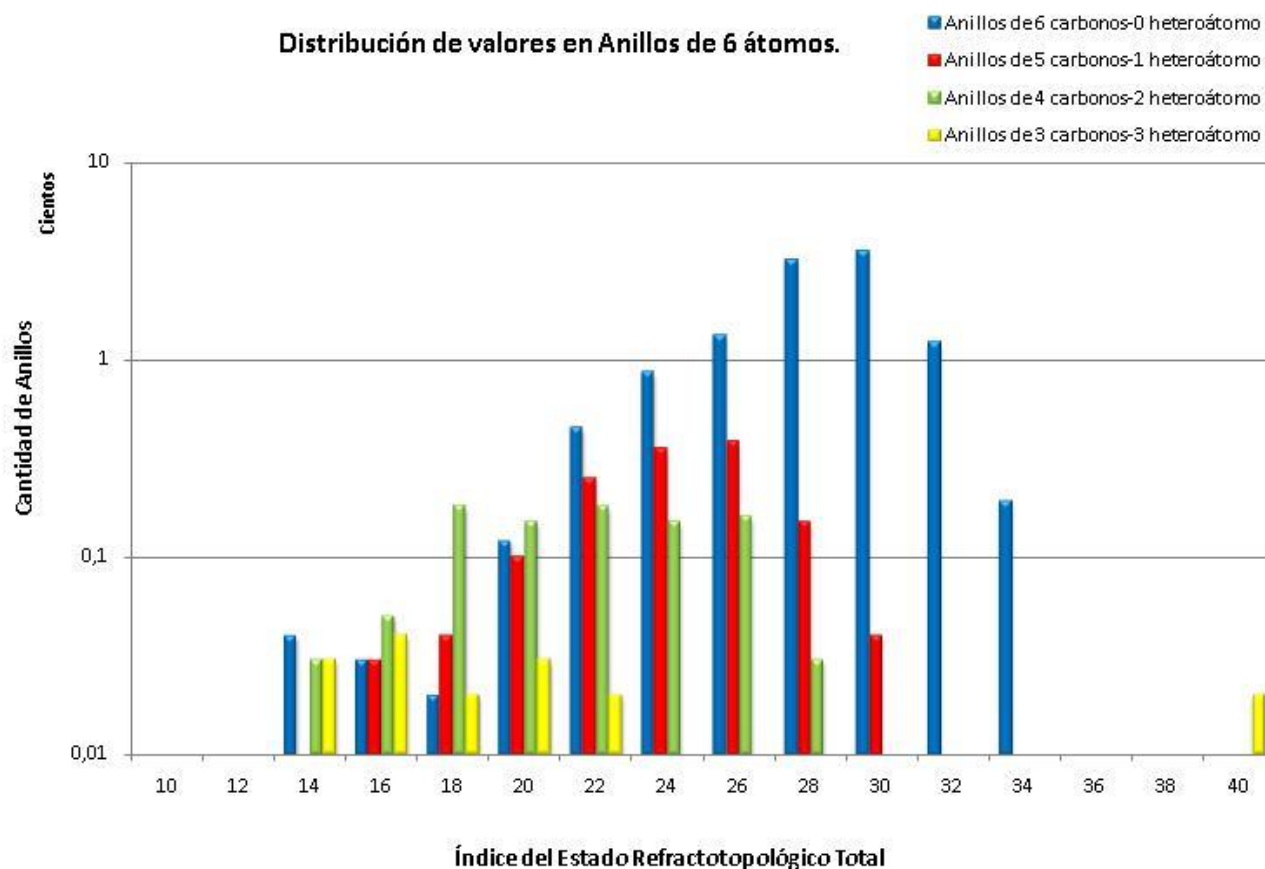


Figura # 3.16. Gráfica de la distribución del Índice en anillos de 6 átomos según su composición.



De la Figura # 3.16 se observa que para determinados valores del índice sería posible sustituir un anillo por otro conteniendo o no heteroátomos, siempre que se conserve el valor total del índice. Esto pudiera explicar además, comportamientos análogos que se observan entre compuestos con diferente composición y similar topología. Este fenómeno se conoce en química medicinal como isostería.

### 3.7.2 Análisis de la muestra de Anillos de 3 a 5 átomos.

En la Tabla # 3.3 se muestran la distribución, en cuanto a composición, en anillos de 3 a 5 átomos.

	Total	Anillos con 0 Heteroátomo	Anillos con 1 Heteroátomo	Anillos con 2 Heteroátomo	Anillos con 3 Heteroátomo
Anillos de 5	236	25	114	92	3
Anillos de 4	6	3	2	1	0
Anillos de 3	15	4	11	0	0

**Tabla # 3.3. Cantidad de anillos de 3 a 5 según la cantidad de Heteroátomos.**

Para las muestras de anillos de 5 y 3 átomos se observan comportamientos similares a la de 6 átomos. La muestra de anillos de 4 átomos no se analizará debido al tamaño reducido de la muestra. A continuación se muestra las Figura # 3.17 y Figura # 3.18 donde se ilustran el comportamiento del índice de los anillos de 5 y 3 átomos según la cantidad de heteroátomos que contengan.

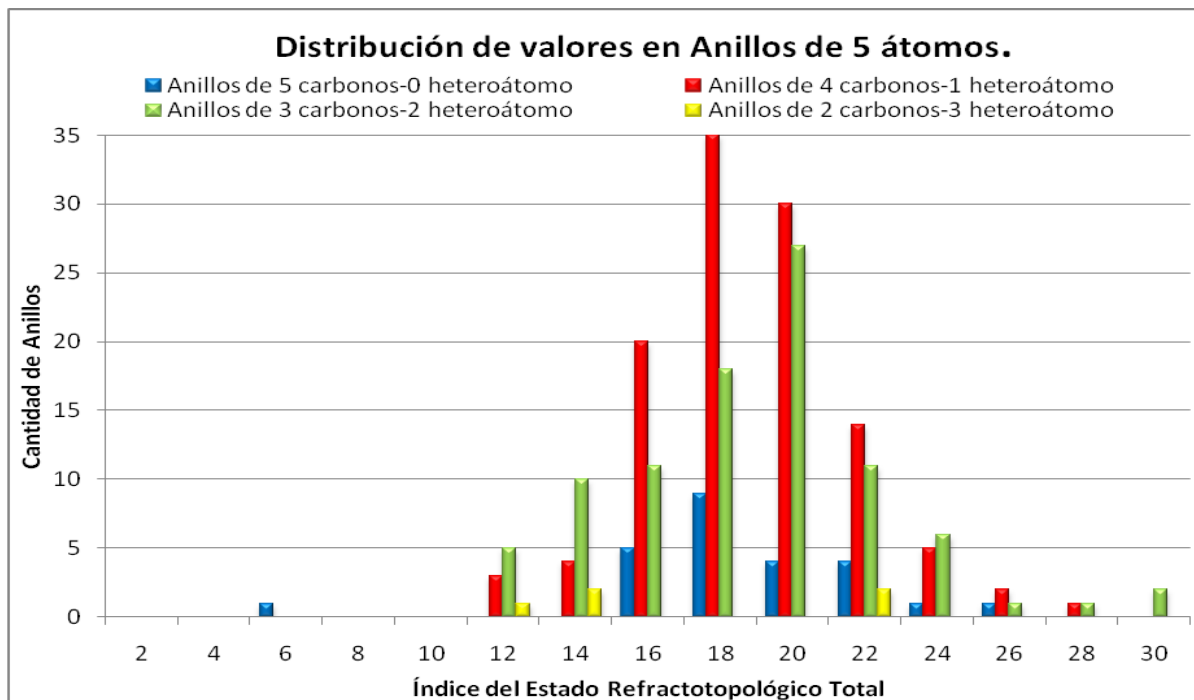


Figura # 3.17. Gráfica de la distribución del Índice en anillos de 5 átomos según su composición.

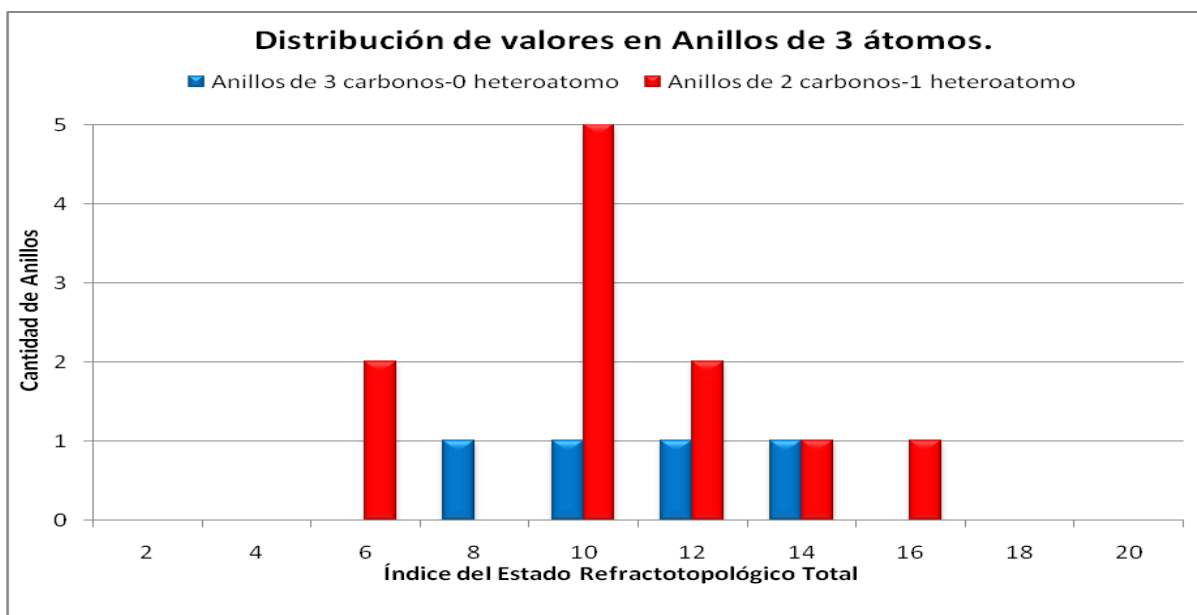


Figura # 3.18. Gráfica de la distribución del Índice en anillos de 3 átomos según su composición.

Las Figura # 3.17 y Figura # 3.18 confirman todo lo expuesto en los análisis anteriormente realizados, con todos los anillos y con la muestra de anillos de 6 átomos.

### 3.8 Conclusiones.

Durante el desarrollo del lenguaje descriptor se obtuvieron algoritmos para la búsqueda de CD y CU. Se codificaron un número de moléculas, obteniéndose ficheros a los cuales se le realizaron pruebas con el fin de garantizar una buena codificación de estas. Se decodificaron ficheros generados por el lenguaje, obteniéndose ficheros .mol de los cuales se puede graficar en dos dimensiones. Se realizó una descripción detallada del fichero .Ide para facilitar el entendimiento de este. Se demostró mediante un estudio estadístico que:

- CDs de iguales topologías pueden tener diferentes comportamientos según su entorno
- CDs de igual topología y diferente composición pueden tener iguales comportamientos.
- CDs de diferentes topologías pueden tener iguales comportamientos.

Esto indica que en estudios experimentales orientados al diseño de fármacos, se puede sustituir en un compuesto dado, un CD por otro y mantener la misma capacidad de interacción de este con otras moléculas.

### CONCLUSIONES

Con la realización del presente trabajo se llegaron a las siguientes conclusiones:

- Se implementaron algoritmos para el reconocimiento de CDs, CUs y fragmentos.
- Se redefinió el lenguaje descriptor y se le adicionaron nuevas reglas gramaticales para la codificación de estructuras químicas.
- Se implementaron algoritmos para codificar CDs, CUs y fragmentos a partir de un fichero .mol, y para decodificar a partir de un fichero .Ide.
- Se obtuvo como resultado un fichero de extensión .Ide por cada fichero de extensión .mol que es codificado, el cual contiene toda la información necesaria para estudios químicos orientados al diseño de fármacos.
- Se demostró, mediante un estudio estadístico, que:
  - CDs de iguales topologías pueden tener diferentes comportamientos según su entorno.
  - CDs de igual topología y diferente composición pueden tener iguales comportamientos.
  - CDs de diferentes topologías y composición pueden tener iguales comportamientos.

Por lo que esto indica que en estudios experimentales orientados al diseño de fármacos, se puede sustituir en un compuesto dado, es decir un CD por otro y mantener la misma capacidad de interacción de este con otras moléculas.

### RECOMENDACIONES

- Incluir criterios de simetría en el lenguaje descriptor para reducir la redundancia de información.
- Incluir los restantes elementos de la tabla periódica para generalizar el lenguaje.
- Incluir el Lenguaje Descriptor de Estructuras Químicas a la Plataforma bioGRATO y desarrollar un intérprete que posibilite visualizar los códigos.

## REFERENCIAS BIBLIOGRÁFICAS

- [1] Raida Elinda Pérez Durán y Suanly Ortiz Tornín. “Lenguaje Descriptor de Estructuras Químicas”, Universidad de las Ciencias Informáticas (UCI), Ciudad de la Habana, 2007, Página 45.
- [2] Aliso Viejo. “SMILES - A Simplified Chemical Language”, [Página Web], Última actualización en el año: 2007, [Consultado el 10 de diciembre del 2007], Disponible en: <http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html>
- [3] Aliso Viejo, “SMARTS - A Language for Describing Molecular Patterns”, [Página Web], Última actualización en el año: 2007, [Consultado el 12 de diciembre del 2007], Disponible en: [http://www.daylight.com/dayhtml\\_tutorials/languages/smarts/index.html](http://www.daylight.com/dayhtml_tutorials/languages/smarts/index.html)
- [4] Bea Perks. “The Royal Society of Chemistry (RSC)”, [Página Web], Última actualización en el año: 2005 [Consultado el 9 de enero del 2008]. Disponible en: <http://www.rsc.org/chemistryworld/News/2005/May/16May2005Internationalchemicalidentifiergoesonline.asp>
- [5] Beda Kosata, “[www.InChI.info](http://www.InChI.info)”, [Página Web], Última actualización en el año: 2007, [Consultado el 9 de enero del 2008], Disponible en: [http://www.inchi.info/inchikey\\_overview\\_en.html](http://www.inchi.info/inchikey_overview_en.html)
- [6] V.E.Golender y A.B.Rozenblit. “Logical and Combinatorial Algorithms for Drug Design.”, 58B Station Road, Letchworth, Herts. SG6 3BE, England. Research Studies Press LTD. Letchworth, Hertfordshire, England. 1989. Cantidad de Páginas: 289.
- [7] R.Carrasco-Velaz, D.Trinchet-Almaguer, S.Ortiz-Tornin, y R.E.Pérez-Durán. “LENGUAJE DESCRIPTOR DE ESTRUCTURAS QUÍMICAS ORIENTADO AL DISEÑO DE FÁRMACOS”, [Consultado el 8 de febrero del 2008], Disponible en: <http://karin.fq.uh.cu/quitel33/contribuciones/Trinchet-CU.pdf> .
- [8] Ethan Nicholas, “Introducing Java SE 6 update 10 Beta”, Año: 2008, [Página web], Última actualización en el año: 2008, [Consultado el 25 de marzo de 2008], Disponible en: <http://java.sun.com/developer/technicalArticles/javase/java6u10/index.html>
- [9] Iván Ramirez, “NetBeans IDE”, Año: 2008, [Página web], Última actualización en el año: 2008, [Consultado el 2 de abril de 2008], Disponible en: <http://netbeans-ide.softonic.com/>

## REFERENCIAS BIBLIOGRÁFICAS

---

- [10] Ricardo La Rosa, “[L-unexlug] Pregunta ECLIPSE”, Año: 2005, [Página web], Última actualización en el año: 2005, [Consultado el 2 de abril de 2008], Disponible en: <http://glove.org.ve/pipermail/l-unexlug/2005-May/000292.html>
- [11] Robin Schumacher, “Rational Rose 98”, Año 1998, [Página web], Última actualización en el año: 1998, [Consultado el 25 de marzo de 2008], Disponible en: <http://www.dbmsmag.com/9809d08.html>
- [12] ---, “Visual Paradigm for UML (ME)”, Año: 2007, [Página web], Última actualización en el año: 2007, [Consultado el 25 de marzo de 2008], Disponible en: [http://www.freedownloadmanager.org/es/downloads/Paradigma\\_Visual\\_para\\_UML\\_\(MÍ\)\\_14720\\_p/](http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(MÍ)_14720_p/)
- [13] LARMAN, C. “UML y Patrones.Introducción al Análisis y Diseño Orientado a Objetos.”, Prentice Hall Hispanoamérica, SA, 1999. vol. I, Cantidad de Páginas: 291.
- [14] R.Carrasco Velar, D.Trinchet-Almaguer, S.Ortiz-Tornin, y R.E.Pérez-Durán. “LENGUAJE ESCRIPTOR DE ESTRUCTURAS QUÍMICAS ORIENTADO AL DISEÑO DE FÁRMACOS”, [Consultado el 8 de febrero del 2008], Disponible en: <http://karin.fq.uh.cu/quitel33/contribuciones/Trinchet-CU.pdf> .
- [15] R.Carrasco Velar, “Nuevos descriptores atómicos y moleculares para estudios de estructura actividad: Aplicaciones”, Centro Químico Farmacéutico CQF, Ciudad de la Habana, 2003, Cantidad de Páginas: 114.
- [16] R. Carrasco y cols. QUITEL XXXII. Conferencia de los Químicos Teóricos de Expresión Latina. La Habana, Septiembre/2007(Comunicación personal).

## BIBLIOGRAFÍA

- Aliso Viejo. “Daylight, SMILES Tutorial”, [Página Web], Última actualización en el año: 2007, [Consultado el 11 de diciembre del 2007], Disponible en: [http://www.daylight.com/dayhtml\\_tutorials/languages/smiles/index.html](http://www.daylight.com/dayhtml_tutorials/languages/smiles/index.html)
- Aliso Viejo. “Daylight, SMILES Examples”, [Página Web], Última actualización en el año: 2007, [Consultado el 11 de diciembre del 2007], Disponible en: [http://www.daylight.com/dayhtml\\_tutorials/languages/smiles/smiles\\_examples.html](http://www.daylight.com/dayhtml_tutorials/languages/smiles/smiles_examples.html)
- Aliso Viejo. “Daylight, Cheminformatic”, [Página Web], Última actualización en el año: 2007, [Consultado el 10 de diciembre del 2007], Disponible en: <http://www.daylight.com/smiles/index.htm>
- Aliso Viejo. “Daylight, Molecules and Reactions in A Computer”, [Página Web], Última actualización en el año: 2007, [Consultado el 10 de diciembre del 2007], Disponible en: <http://www.daylight.com/dayhtml/doc/theory/theory.mol.html>
- Aliso Viejo, “Daylight Theory, SMARTS - A Language for Describing Molecular Patterns”, [Página Web], Última actualización en el año: 2007, [Consultado el 12 de diciembre del 2007], Disponible en: <http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>
- Stephen E. Stein, Stephen R. Heller, y Dmitrii Tchekhovskoi, “Internacional Unión of Pure and Applied Chemistry”, [Página Web], Última actualización en el año: 2007, [Consultado el 9 de enero del 2008], Disponible en: <http://old.iupac.org/inchi/>
- Roald Hoffmann, “International Union of Pure and Applied Chemistry”, [Página Web], Última actualización en el año: 2008, [Consultado el 9 de enero del 2008], Disponible en: [http://www.iupac.org/dhtml\\_home.html](http://www.iupac.org/dhtml_home.html)
- A. Monge, A. Arrault, C. Marot, L. Morin-Allory. “ANALYSIS OF A SET OF 2.6 MILLION UNIQUE COMPOUNDS GATHERED FROM THE LIBRARIES OF 32 CHEMICAL PROVIDERS”, [Página Web], Última actualización en el año: 2007, [Consultado el 9 de enero del 2008], Disponible en: <http://www.univ-orleans.fr/icoa/eposter/eccc10/monge/>



## BLIBLIOGRAFÍA

---

- S.Heller, S.Stein, “International Union of Pure and Applied Chemistry”, [Página Web], Última actualización en el año: 2007, [Consultado el 11 de enero del 2008], Disponible en: <http://www.iupac.org/projects/2000/2000-025-1-800.html>
- Wendy A. Warr, “ IUPAC PROJECT MEETINGS: EXTENSIBLE MARKUP LANGUAGE (XML) DATA DICTIONARIES AND CHEMICAL IDENTIFIER”, NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, (NIST), GAITHERSBURG, MARYLAND, USA , Año: 2004, [Consultado el 11 de enero del 2008], Disponible en: <http://www.warr.com/inchi.pdf>
- Peter Murray-Rust, Henry Rzepa, Steve Stein, “The INChI as an LSID for molecules in lifescience”, [Página Web], Última actualización en el año: 2004, [Consultado el 11 de enero del 2008], Disponible en: <http://lists.w3.org/Archives/Public/public-swls-ws/2004Sep/att-0026/inchi.html>
- Antony (Tony) N. Davies, “Chemistry International—Newsmagazine for IUPAC”, [Página Web], Última actualización en el año: 2004, [Consultado el 11 de enero del 2008], Disponible en: [http://www.iupac.org/publications/ci/2004/2604/pp6\\_2002-022-1-024.html](http://www.iupac.org/publications/ci/2004/2604/pp6_2002-022-1-024.html)
- A. McNaught, S. Heller, S. Stein, D. Tchekovskoi, J. Kahovec, A. Yerin, “Unofficial InChI FAQ”, [Página Web], Última actualización en el año: 2007, [Consultado el 9 de enero del 2008], Disponible en: <http://wwmm.ch.cam.ac.uk/inchifaq/#Who%20is%20responsible%20for%20InChI?>
- Bea Perks. “The Royal Society of Chemistry (RSC)”, [Página Web], Última actualización en el año: 2005 [Consultado el 9 de enero del 2008]. Disponible en: <http://www.rsc.org/chemistryworld/News/2005/May/16May2005Internationalchemicalidentifiergoesonline.asp>
- Steve Heller, Alan McNaught, Igor Pletnev, Steve Stein, Dmitrii Tchekhovskoi, “Internacional Unión of Pure and Applied Chemistry”, [Página Web], Última actualización en el año: 2007, [Consultado el 15 de enero del 2008], Disponible en: <http://www.iupac.org/inchi/release102.html>
- Jim Downing, “Unilever Centre for Molecular Informatic, Cambridge - Jim Downing >>Blog Archive >> Why InChIKey?”, [Página Web], Última actualización en el año: 2007, [Consultado el 15 de enero del 2008], Disponible en: <http://wwmm.ch.cam.ac.uk/blogs/downing/?p=126>

## BLIBLIOGRAFÍA

---

- --, "What is Rational Rose?", [Página web], Última actualización en el año: 2005, [Consultado el 25 de marzo de 2008], Disponible en: [http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183\\_gci516025,00.html](http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci516025,00.html)
- ---, "Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME))", Año: 2007, [Página web], Última actualización en el año: 2007, [Consultado el 25 de marzo de 2008], Disponible en: [http://www.freedownloadmanager.org/es/downloads/Paradigma\\_Visual\\_para\\_UML\\_\(M%C3%8D\)\\_14720\\_p/](http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/)
- Visual Paradigm. com, "UML CASE Tools- Free for Learning UML", Año: 2008, [Página web], Última actualización en el año: 2008, [Consultado el 26 de marzo de 2008], Disponible en: <http://www.visual-paradigm.com/product/vpuml/26/03/08>
- Company Headquarters, "Visual Paradigm for UML", Última actualización 2008, [Consultado el 5 de febrero de 2008], Disponible en: <http://www.visual-paradigm.com/product/vpuml/>
- Justo Mendez, "Lenguajes de Programación", Última actualización 1997, [Consultado el 26 de marzo de 2008], Disponible en: <http://www.monografias.com/trabajos/lengprog/lengprog.shtml>
- Ethan Nicholas, "Introducing Java SE 6 update 10 Beta", Año: 2008, [Página web], Última actualización en el año: 2008, [Consultado el 25 de marzo de 2008], Disponible en: <http://java.sun.com/developer/technicalArticles/javase/java6u10/index.html>
- Patxi Echarte, "Introducción a la Plataforma .NET y Mono", Año 2005, [Consultado el 1 de abril de 2008], disponible en: <http://www.eslomas.com/index.php/archives/2005/05/11/introduccion-plataforma-net-y-mono/>
- Enrique Serrano Valle, "Eclipse Tutoria", Año: 2008, [Página web], Última actualización en el año: 2004, [Consultado el 7 de febrero de 2008], Disponible en: <http://eclipsetutorial.forge.os4os.org/in1.htm>
- <http://www.eclipse.org/> 07/02/08
- Xavier Ferré Grau, "Tutorial UML Desarrollo Orientado a Objetos con UML", Año 2004, [Página web], Última actualización en el año: 2004, [Consultado el 5 de febrero de 2008], Disponible en: <http://www.clikear.com/manuales/uml/>

## BLIBLIOGRAFÍA

---

- ---, "InterPlanet - ¿Qué son Linux y el Open Source?", Año 2006, [Página web], Última actualización en el año: 2006, [Consultado el 9 de febrero de 2008], Disponible en: [http://www.interplanet.com.mx/deinteres\\_linux\\_opensource.htm](http://www.interplanet.com.mx/deinteres_linux_opensource.htm) 09/02/08
- Bill Syken, "Innovators-Miguel de Icaza", Año 2000, [Página web], Última actualización en el año: 2000, [Consultado el 5 de febrero de 2008], Disponible en: [http://www.time.com/time/innovators/web/profile\\_icaza.html](http://www.time.com/time/innovators/web/profile_icaza.html)

## ANEXOS

### Anexo #1. Algoritmos del lenguaje descriptor.

```
private void buscarAnillo(){
    ArrayList<JAtomo> estado=new ArrayList<JAtomo>();
    ArrayList<JAtomo> atomos=mol.obtenerAtomos();
    for(int i=0;i<atomos.size();i++){
        JAtomo aux=atomos.get(i);
        String simbolo=aux.getSymbol().toString();
        if(!simbolo.equals("Br") &&
            !simbolo.equals("Cl") &&
            !simbolo.equals("F") &&
            !simbolo.equals("I")) && !mol.esExtremo(aux)){
            ArrayList<JAtomo> hijos=mol.obtenerAdyacentes(aux);
            for(int j=0;j<hijos.size();j++){
                JAtomo temp=hijos.get(j);
                estado.clear();
                mol.padreNulo();
                if(!temp.getSymbol().equals("Cl") &&
                    !temp.getSymbol().equals("Br") && !temp.getSymbol().equals("I") &&
                    !temp.getSymbol().equals("F") && !mol.esExtremo(temp)){
                    temp.setPadre(aux);
                    estado.add(temp);
                    identificarAnillo(temp,aux,estado);
                }
            }
        }
    }
}
```

Figura # A1.1. Algoritmo para buscar CD Anillo.

```

private void buscarCluster(ArrayList<JAtomo> l){
    JMatrix matrix=mol.getMatrix();
    ArrayList<JAtomo> t=l;
    int cont=0;
    int iterar=matrix.getDimencion();
    for(int i=0;i<iterar;i++){
        ArrayList<JEnlace> enlace=new ArrayList<JEnlace>();
        for(int j=0;j<iterar;j++){
            int cell=matrix.getCell(i,j);
            if(cell==1){
                cont++;
                JAtomo a=mol.obtenerAtomoEnPos(i);
                JAtomo b=mol.obtenerAtomoEnPos(j);
                JEnlace temp=mol.obtenerEnlace(a,b);
                enlace.add(temp);
            }
        }
        if(cont==3){
            JCentroDescriptor c3=new JCluster(3,enlace);
            JCluster tem=(JCluster) c3;
            JAtomo importante=tem.atomoImportante();
            String sT=importante.getSymbol();
            if(sT.equals("N") || sT.equals("P") || sT.equals("As") || sT.equals("C"))
                l.remove(importante);
            ArrayList<JAtomo> aux=tem.atomosAdyacentes();
            int iterar1=aux.size();
            for(int j=0;j<iterar1;j++){
                JAtomo a=aux.get(j);
                int iterar2=t.size();
                for(int h=0;h<iterar2;h++){
                    JAtomo b=t.get(h);
                    if(a.elementosIguales(b)){
                        l.remove(a);
                        break;
                    }
                }
            }
            c3.calculoIRTT();
            c3.crearCodigo();
            c3.crearEstructura();
            adicionarCentroDescriptor(c3);
        }
        else if(cont==4){
            JCentroDescriptor c4=new JCluster(4,enlace);
            JCluster cl=(JCluster) c4;
            if(cl.atomoImportante().getSymbol().equals("N") ||
                cl.atomoImportante().getSymbol().equals("P") ||
                cl.atomoImportante().getSymbol().equals("As") ||
                cl.atomoImportante().getSymbol().equals("C"))
                l.remove(cl.atomoImportante());
            ArrayList<JAtomo> aux=cl.atomosAdyacentes();
            for(int j=0;j<aux.size();j++){
                JAtomo a=aux.get(j);
                for(int h=0;h<t.size();h++){
                    JAtomo b=t.get(h);
                    if(a.elementosIguales(b)){
                        l.remove(a);
                    }
                }
            }
        }
    }
}

```

Figura # A1.2. Algoritmo para buscar Clústeres primera parte.

```
                break;
            }
        }
    }
    c4.calculoIRTT();
    c4.crearCodigo();
    c4.crearEstructura();
    adicionarCentroDescriptor(c4);
    for(int k=enlace.size()-1;k>=0;k--){
        ArrayList<JEnlace> list=new ArrayList<JEnlace>();
        JEnlace b1=enlace.get(k);
        for(int p=0;p<enlace.size();p++){
            JEnlace b2=enlace.get(p);
            if(!b1.equals(b2)){
                list.add(b2);
            }
        }
        JCentroDescriptor c4_1=new JCluster(3,list);
        c4_1.calculoIRTT();
        c4_1.crearCodigo();
        c4_1.crearEstructura();
        adicionarCentroDescriptor(c4_1);
    }
    cont=0;
}
}
```

Figura # A1.3. Algoritmo para buscar Clústeres segunda parte.

```
private void buscarCamino(){
    ArrayList<JCentroDescriptor> descriptionCenter=cd;
    ArrayList<JAtomo> listOrigen=new ArrayList<JAtomo>();
    for(int i=0;i<descriptionCenter.size();i++){
        JCluster clusterOrigen=null;
        JAnillo ringOrigen=null;
        JHeteroAtomo heteroAtomoOrigen=null;
        JGrupFuncional grupFuncional=null;
        if(descriptionCenter.get(i) instanceof JCluster){
            clusterOrigen=(JCluster) descriptionCenter.get(i);
            listOrigen.clear();
            listOrigen=mol.asignarArray(clusterOrigen.atomosAdyacentes());
            buscarDestinoDelCamino(i,clusterOrigen,listOrigen);
        }
        if(descriptionCenter.get(i) instanceof JAnillo){
            ringOrigen=(JAnillo) descriptionCenter.get(i);
            listOrigen.clear();
            listOrigen=mol.asignarArray(ringOrigen.obtenerAtomos());
            buscarDestinoDelCamino(i,ringOrigen,listOrigen);
        }
        if(descriptionCenter.get(i) instanceof JHeteroAtomo){
            heteroAtomoOrigen=(JHeteroAtomo) descriptionCenter.get(i);
            listOrigen.clear();
            listOrigen.add(heteroAtomoOrigen.atomoImportante());
            buscarDestinoDelCamino(i,heteroAtomoOrigen,listOrigen);
        }
        if(descriptionCenter.get(i) instanceof JGrupFuncional){
            grupFuncional=(JGrupFuncional) descriptionCenter.get(i);
            listOrigen.clear();
            listOrigen.add(grupFuncional.atomoImportante());
            buscarDestinoDelCamino(i,grupFuncional,listOrigen);
        }
    }
}
```

Figura # A1.4. Algoritmo para buscar CU.



```

private void buscarDestinoDelCamino(int i, JCentroDescriptor descriptorOrigen,
    ArrayList<JAtomo> listOrigen){
    ArrayList<JCentroDescriptor> descriptionCenter=cd;
    ArrayList<JAtomo> listDestino=new ArrayList<JAtomo>();
    ArrayList<JAtomo> visitados=new ArrayList<JAtomo>();
    for(int j=i+1;j<descriptionCenter.size();j++){
        if(descriptionCenter.get(j) instanceof JCluster){
            JCluster clusterDestino=(JCluster) descriptionCenter.get(j);
            listDestino.clear();
            listDestino=mol.asignarArray(clusterDestino.atomosAdyacentes());
            if(descriptorOrigen instanceof JCluster){
                JCluster clusterOrigen=(JCluster) descriptorOrigen;
                clusterToCluster(clusterOrigen, clusterDestino,
                    listOrigen, listDestino, visitados);
                listOrigen.clear();
                listOrigen=mol.asignarArray(clusterOrigen.atomosAdyacentes());
            }
        }
        if(descriptionCenter.get(j) instanceof JAnillo){
            JAnillo ringDestino=(JAnillo) descriptionCenter.get(j);
            listDestino.clear();
            listDestino=mol.asignarArray(ringDestino.obtenerAtomos());
            if(descriptorOrigen instanceof JCluster){
                JCluster clusterOrigen=(JCluster) descriptorOrigen;
                clusterToAnillo(clusterOrigen, ringDestino,
                    listOrigen, listDestino, visitados);
                listOrigen.clear();
                listOrigen=mol.asignarArray(clusterOrigen.atomosAdyacentes());
            }
            if(descriptorOrigen instanceof JAnillo){
                JAnillo ringOrigen=(JAnillo) descriptorOrigen;
                anilloToAnillo(ringOrigen, ringDestino,
                    listOrigen, listDestino, visitados);
                listOrigen.clear();
                listOrigen=mol.asignarArray(ringOrigen.obtenerAtomos());
            }
        }
        if(descriptionCenter.get(j) instanceof JHeteroAtomo){
            JHeteroAtomo heteroAtomoDestino=(JHeteroAtomo) descriptionCenter.get(j);
            listDestino.clear();
            listDestino.add(heteroAtomoDestino.atomoImportante());
            if(descriptorOrigen instanceof JCluster){
                JCluster clusterOrigen=(JCluster) descriptorOrigen;
                clusterToHeteroAtomo(clusterOrigen, heteroAtomoDestino,
                    listOrigen, listDestino, visitados);
                listOrigen.clear();
                listOrigen=mol.asignarArray(clusterOrigen.atomosAdyacentes());
            }
            if(descriptorOrigen instanceof JAnillo){
                JAnillo ringOrigen=(JAnillo) descriptorOrigen;
                anilloToHeteroAtomo(ringOrigen, heteroAtomoDestino,
                    listOrigen, listDestino, visitados);
                listOrigen.clear();
                listOrigen=mol.asignarArray(ringOrigen.obtenerAtomos());
            }
        }
        if(descriptorOrigen instanceof JHeteroAtomo){

```

Figura # A1.5. Algoritmo para buscar el destino de un camino, primera parte.



```
        for(int k=0;k<listOrigen.size();k++){
            JAtomo b=listOrigen.get(k);
            visitados.clear();
            identificarCamino(b,listDestino,b,descriptorOrigen,heteroAtomoDestino,visitados)
        }
    }
}
if(descriptionCenter.get(j) instanceof JGrupFuncional){
    JGrupFuncional gFDestino=(JGrupFuncional) descriptionCenter.get(j);
    listDestino.clear();
    listDestino.add(gFDestino.atomoImportante());
    if(descriptorOrigen instanceof JCluster){
        JCluster clusterOrigen=(JCluster) descriptorOrigen;
        clusterToGrupoFuncional(clusterOrigen, gFDestino,
            listOrigen, listDestino, visitados);
        listOrigen.clear();
        listOrigen=mol.asignarArray(clusterOrigen.atomosAdyacentes());
    }
    if(descriptorOrigen instanceof JAnillo){
        JAnillo anilloOrigen=(JAnillo) descriptorOrigen;
        anilloToGrupoFuncional(anilloOrigen,gFDestino,
            listOrigen, listDestino, visitados);
        listOrigen.clear();
        listOrigen=mol.asignarArray(anilloOrigen.obtenerAtomos());
    }
    if(descriptorOrigen instanceof JHeteroAtomo){
        for(int k=0;k<listOrigen.size();k++){
            JAtomo b=listOrigen.get(k);
            visitados.clear();

            identificarCamino(b,listDestino,b,descriptorOrigen,gFDestino,visitados);
        }
    }
    if(descriptorOrigen instanceof JGrupFuncional){
        JGrupFuncional gFOrigen=(JGrupFuncional) descriptorOrigen;
        JAtomo temp=gFOrigen.atomoImportante();
        visitados.clear();
        identificarCamino(temp,listDestino,temp,gFOrigen,gFDestino,visitados);
    }
}
}
```

Figura # A1.6. Algoritmo para buscar el destino de un camino, segunda parte.

## Anexo # 2. Resultados de Codificación.

```

003002
0502020.6172 0105d0204s0203s0102d0101s
0202003.8026 03s010203s0204
0202002.4762 04s020304d0105
c03014.3384 0203s0102d0101s0105d0204
c00000.0000 0203s0204
0202003.8026c03014.33840202002.4762
0202003.8026c00000.00000202002.4762

```

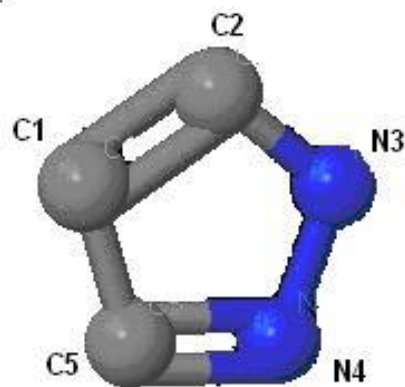


Figura # A2.1. Codificación y Molécula del 1H-pyrazole.

```

004004
0001017.5949 0202s0101s0103s0106
0502018.6007 0106s0205d0104s0103d0202s
0202001.4762 02s010102s010602s0103
0202002.3167 05s010405d0106
c01005.3524 0103d0104s0205
c00000.0000 0106d0205
c02010.6915 0202s0103d0104s0205
c01004.1162 0202s0106d0205
0001017.5949c01005.35240202002.3167
0001017.5949c00000.00000202002.3167
0202001.4762c02010.69150202002.3167
0202001.4762c01004.11620202002.3167

```

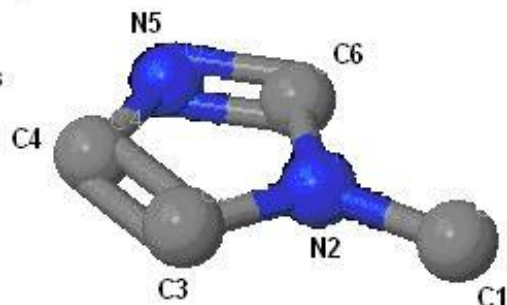


Figura # A2.2. Codificación y Molécula del 1-methylimidazole.

```

306011
0001015.8236 0102s0101s0103s0206
0002008.3425 0103s0102d0504s0505
0204010.6504 07s0101
0202005.4561 06s0102
0205000.5473 04d0103
0205001.8673 05s0103
c00000.0000 0101s0407
c00000.0000 0103d0504
c00000.0000 0103s0505
c01004.4395 0102s0101s0407
c00000.0000 0102s0206
c02007.6711 0407s0101s0102s0206
c03010.3675 0407s0101s0102s0103d0504
c03010.3675 0407s0101s0102s0103s0505
c02005.9280 0206s0102s0103d0504
c02005.9280 0206s0102s0103s0505
c01002.6964 0504d0103s0505
0001015.8236c00000.00000204010.6504
0001015.8236c00000.00000205000.5473
0001015.8236c00000.00000205001.8673
0002008.3425c01004.43950204010.6504
0002008.3425c00000.00000202005.4561
0204010.6504c02007.67110202005.4561
0204010.6504c03010.36750205000.5473
0204010.6504c03010.36750205001.8673
0202005.4561c02005.92800205000.5473
0202005.4561c02005.92800205001.8673
0205000.5473c01002.69640205001.8673

```

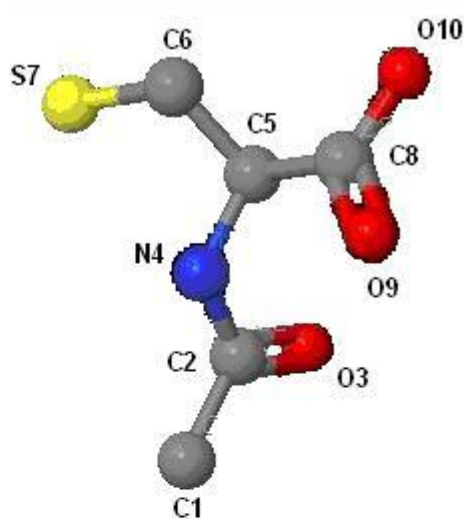


Figura # A2.3. Codificación y Molécula del 2-amino-3-sulfanyl-propanoic acid.

## GLOSARIO DE TÉRMINOS

**ADO:** (ActiveX Data Objects) ó (Objetos de datos ActiveX): es uno de los mecanismos que usan los programas en las computadoras para administrar bases de datos, darles órdenes y obtener resultados de ellas.

**Ant:** es una herramienta Open-Source utilizada en la compilación y creación de programas Java.

**Bioinformática:** es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

**C:** Lenguaje de programación estructurado de propósito general que ofrece control de flujo, estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel.

**Caracteres imprimibles:** son caracteres los cuales no son espacios en blancos ni alfanuméricos.

**CORBA:** (Common Object Request Broker Architecture) arquitectura común de intermediarios en peticiones a objetos, es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

**CVS:** herramienta de control de versiones.

**Fármacos:** es término farmacológico para cualquier compuesto biológicamente activo, capaz de modificar el metabolismo de las moléculas sobre las que hace efecto.

**Fórmula empírica:** En química es una expresión o forma que representa la proporción más simple en la que están presentes los átomos que forman un compuesto químico.

**GRASP:** son patrones Generales de Software para Asignación de Responsabilidades, los cuales se aplican en el diseño de la aplicación.

**Halógenos:** elementos químicos Cl, Br, I, y el F.

**HAVPAC:** Software para realizar cálculos mecánicos cuánticos semiempíricos de la estructura y la reactividad molecular.

**HTML:** (HyperText Markup Language) ó (Lenguaje de Marcas Hipertextuales), lenguaje en el que se programan las páginas web.

**IBM:** International Business Machines, una de las mayores compañías de fabricación de hardware y software en el mundo desde los años cincuenta. Conocida coloquialmente como el Gigante Azul.

**IDE:** Un “Integrate Development Enviroment” es una herramienta de soporte al proceso de desarrollo de software que integra las funciones básicas de edición de código, compilación y ejecución de programas, entre otras.

**Índices híbrido:** Número que se calcula a partir de algoritmos que se aplican a la matriz de adyacencia o de conectividad en el grafo completo donde los vértices de esa matriz de conectividad se ponderan con el valor de una propiedad químico-física del tipo de átomo que separan.

**Índice topológico y topográfico:** Número que se calcula generalmente a partir de la matriz de adyacencia o de distancia de los elementos de un grafo molecular.

**IUPAC:** (International Union of Pure and Applied Chemistry) ó (Unión Internacional de Química Pura y Aplicada).

**JDBC:** (Java DataBase Connectivity) es un conjunto de APIs diseñado para ofrecer acceso a bases de datos Java.

**JINI:** El Jini es una tecnología, desarrollada por Sun Microsystems, que proporciona un mecanismo sencillo para que diversos dispositivos conectados a una red puedan colaborar y compartir recursos sin necesidad de que el usuario final tenga que planificar y configurar dicha red.

**JSP:** Java Server Pages es la tecnología del lado del servidor para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java.

**Licencia GNU:** es un acrónimo recursivo para «GNU No es Unix» y se pronuncia fonéticamente en español. El proyecto GNU fue lanzado en 1984 para desarrollar un completo sistema operativo tipo Unix, bajo la filosofía del software libre: el sistema GNU.

**Licencia GPL:** Licencia Pública General de GNU, llamada comúnmente GNU GPL, la usan la mayoría de los programas de GNU y más de la mitad de las aplicaciones de software libre.

**Licencia SPL:** (Sun Public License) ó (Licencia Pública de Sun).

**Máquinas virtuales:** es un software que emula a un ordenador y puede ejecutar programas como si fuese un ordenador real.

**Matriz de conectividad:** Matriz que se construye a partir de la conexión de cada átomo en la molécula con los adyacentes.

**Mol:** extensión del fichero de entrada del lenguaje descriptor.

**Multiplataforma:** es el programa o dispositivo que puede utilizarse sin inconvenientes en distintas plataformas de hardware y sistemas operativos.

**NIST:** (National Institute for Standards and Technology) ó (Instituto Nacional de Estándares y Tecnología).

**ODBC:** (Open DataBase Connectivity) ó (Conectividad abierta de bases de datos), Una interfaz de programación de aplicaciones (API), que permite a las aplicaciones tener acceso a bases de datos multiplataforma desde diversas especificaciones estándar de orígenes de datos.

**Open Source:** Cualidad de algunos softwares de incluir el código fuente en la distribución del programa. En general se usa para referirse al software libre.

**Oracle:** es un sistema de gestión de base de datos relacional.

**OREX:** Expert System for Drug Design. Sistema Experto desarrollado en 1990 basado en la descomposición topológica de las moléculas en fragmentos estructurales y su asociación a las actividades biológicas.

**Perl:** (Practical Extracting and Reporting Language) ó (Lenguaje Práctico para la Extracción e Informe) es un lenguaje de propósito general que es utilizado para un amplio rango de tareas incluyendo administración de sistemas, desarrollo web, programación en red etc.

**PHP:** (Personal Home Page) Lenguaje de programación de licencia libre, embebido dentro del HTML y ejecutado en el servidor antes de ser enviado al navegador, usado para crear paginas dinámicas.

**Python:** es un lenguaje de programación que facilita la creación de programas compactos y legibles.

**Quirales:** es la propiedad que tienen ciertas moléculas de poder existir bajo dos formas que son imágenes especulares la una de la otra, es decir, una es la imagen reflejada en un espejo de la otra.

**MI:** (Java Remote Method Invocation), Invocación de métodos remotos, consiste en que un objeto acceda a un método (una de las funcionalidades) de otro objeto remoto (que esté situado en otro punto de una red).

**RUP:** Proceso Unificado (Rational Unified Process) metodología para el desarrollo de sistemas informáticos, dirigidos por casos de uso.

**Servlet:** son objetos que corren dentro del contexto de un servidor de aplicaciones.

**Tóxicos:** Sustancias de origen no biológico, presentes en el agua, con capacidad de ser absorbidas, penetrar en el organismo, transformarse y ocasionar diversas alteraciones orgánicas y funcionales en el ser humano.

**XML:** (eXtensible Markup Language) ó (Lenguaje de marcas extencibles), es considerado como un metalenguaje de definición de documentos estructurados mediante marcas o etiquetas.