



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES  
ARAGÓN

**“REMOTIZACIÓN DEL PORTAL UNIVERSITARIO  
DE SIMULACIONES NUMÉRICAS”**

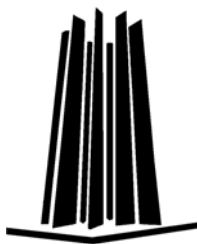
*T R A B A J O E S C R I T O*

**EN LA MODALIDAD DE TESIS  
QUE PARA OBTENER EL TÍTULO DE:  
INGENIERO EN COMPUTACIÓN**

**P R E S E N T A :**

**ERICK DANIEL VALLE VIDAL**

**ASESORES: M. EN I. LILIANA HERNÁNDEZ CERVANTES  
DR. ALFREDO J. SANTILLAN GONZÁLEZ**



MÉXICO, 2007



Universidad Nacional  
Autónoma de México

Dirección General de Bibliotecas de la UNAM

**Biblioteca Central**



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

# AGRADECIMIENTOS

---

A mis padres:

**Benjamín Valle y Genoveva Vidal**

Quienes con sus enseñanzas, consejos, apoyo y amor me han puesto en el camino correcto para conseguir mis metas y lograr ser quien soy.

A mis hermanos:

**Julio, Víctor y Luis**

Quienes son una parte especial en mi vida e influyeron en gran medida a conducirme con esfuerzo, empeño y dedicación en mi vida.

**A toda la familia Valle**

Que siempre tuvieron amor, apoyo y palabras de aliento para mí.

**A Lorena**

Quien con su amor y comprensión siempre me motivo a seguir adelante.

# **A G R A D E C I M I E N T O S**

---

## **A Liliana y Alejandro**

Quienes más que mis maestros fueron mis amigos.

## **Al Dr. Alfredo Santillán**

Que fue una pieza muy importante en el desarrollo de este trabajo, además de un buen amigo.

## **A Eliza**

Cuya amistad me ayudo en los momentos más críticos de mi vida.

## **A la UNAM**

Por brindarme la oportunidad de realizar mi formación académica.

"En esta tesis se presenta un trabajo multidisciplinario de Computación y Astrofísica: la primera parte fue dirigida por la M. en I. Liliana Hernández Cervantes y la segunda por el Dr. Alfredo J. Santillán González, del Instituto de Astronomía y de la Dirección General de Servicios de Cómputo Académico de la UNAM, respectivamente"

# INDICE

Objetivo General.....	1
Objetivos Particulares.....	1
Justificación.....	2

## Capítulo 1

### Introducción.

1.1 Bases de datos.....	3
1.1.1 Tipos de bases de datos.....	4
1.1.2 Modelos de bases de datos.....	5
1.1.3 Sistemas Gestores de bases de datos.....	10
1.1.4 Normalización de bases de datos.....	12
1.2 Herramientas de Programación.....	13
1.2.1 Programas y algoritmos.....	14
1.2.2 Diagramas de flujo.....	15
1.2.3 Programación e ingeniería del software.....	16
1.2.4 Hojas de estilo.....	17
1.3 Bases de datos y programación orientada al Web.....	18
1.3.1 Diseño de Páginas.....	18
1.3.2 Lenguajes de marcas.....	20
1.3.3 Integración de datos en el Web.....	22

## Capítulo 2

### Estructura del Portal.

2.1 Introducción.....	26
2.2 Descripciones de herramientas.....	26
2.2.1 Análisis de lenguajes de programación orientados a Internet.....	26
2.2.1.1 ASP.....	27
2.2.1.2 JSP.....	29
2.2.1.3 AJAX.....	30
2.2.1.4 PHP.....	31
2.2.1.5 Comparación entre lenguajes.....	32
2.2.2 Análisis de bases de datos compatibles con los anteriores lenguajes.....	33
2.2.2.1 PostGreeSQL.....	34
2.2.2.2 SQL Server.....	36
2.2.2.3 MySQL.....	37
2.2.2.4 Comparación entre bases de datos.....	38
2.4 Código numérico utilizado en las simulaciones.....	41
2.4.1 Código Hidrodinámico Zeus 3D.....	41

## Capítulo 3

### Implementación del portal universitario de simulaciones numéricas remotas.

3.1. Generación de la base de datos. ....	43
3.2 Programación del portal.....	44
3.2.1 Página principal.....	44
3.2.2 Desarrollo del catálogo de simulaciones numéricas.....	46
3.2.3.1 Presentación.....	51
3.2.3.2 Páginas que solo requieren html.....	54
3.2.3.3 Búsqueda Avanzada.....	59
3.2.3.4 Búsqueda Personalizada.....	63
3.2.3.5 Motor de búsqueda.....	68
3.2.3 Desarrollo del portal para la realización de simulaciones numéricas.....	76
3.2.3.1 Presentación.....	77
3.2.3.2 Descripción de elementos que siempre son requeridos.....	79
3.2.3.3 Desarrollo de funciones.....	79
3.2.3.3.1 Depuradores.....	80
3.2.3.3.2 Modificación al archivo Inzeus.....	83
3.2.3.3.3 Instrucciones a ejecutar.....	84
3.2.3.3.4 Inserción a la base de datos.....	84
3.2.3.4 Estructura del código principal.....	85

## Capítulo 4

### Aplicación del Portal Universitario a un Problema Astrofísico.

4.1 Nubes de Alta Velocidad o Flujos Supersónicos de Hidrógeno Neutro.....	87
4.2 Portal Universitario de Simulaciones Numéricas: Nubes de Alta Velocidad...87	
4.2.1 Simulaciones Numéricas Remotas.....	88
4.2.2 Catálogo de Simulaciones Numéricas.....	88
4.3 Consultas Básicas.....	90
4.3.1 Características del Medio Interestelar.....	90
4.4 Consultas Avanzadas.....	92

## Capítulo 5

Conclusiones y Trabajo a futuro.....	94
Bibliografías.....	96
Referencias.....	98
Apéndice A.....	99
Apéndice B.....	112
Glosario.....	136

## **OBJETIVO GENERAL**

Implementar una solución eficiente a través de diferentes herramientas computacionales, para el desarrollo del portal universitario de simulaciones numéricas, el cual tiene como característica principal, permitir a usuarios de cualquier parte del mundo hacer simulaciones numéricas remotas y generar una base de datos a través de una intuitiva interfaz *Web*.

## **OBJETIVOS PARTICULARES**

1. Investigar y elegir las posibles herramientas a utilizar para la implementación de la base de datos en *Web* y que contengan los atributos requeridos por el tipo de simulación a ejecutar.
2. Identificar y seleccionar las herramientas adecuadas para desarrollar la interfaz gráfica del portal, de tal modo que esta sea sencilla e intuitiva para el usuario.
3. Ejecutar remotamente a través del portal el código hidrodinámico *Zeus 3D*, que será utilizado para hacer las simulaciones numéricas.
4. Diseñar un mecanismo de integridad de información, con el objetivo de evitar que al momento de ingresar las condiciones iniciales en el portal, se introduzcan valores no válidos o fuera de rango, con ello se hará un uso más eficiente del procesador, ya que de esta forma no se ejecutarán simulaciones que no cumplan con estas características.
5. Generar archivos de salida utilizando herramientas de compresión (archivos tipo *.tar.gz*), así como una pequeña película en formato *mpeg* generada con estos archivos.
6. Diseñar una interfaz intuitiva para hacer diferentes tipos de búsquedas en la base de datos.



## **JUSTIFICACIÓN**

Usualmente el realizar simulaciones numéricas en diferentes áreas de la ciencia e ingeniería, consiste en ejecutar un código numérico, analizar los datos obtenidos, llegar a un producto final y después desecharlos. Gracias al avance en las nuevas Tecnologías de la Información, nos permite desarrollar un sistema como el que se presenta en esta tesis, que consiste básicamente en desarrollar un sistema capaz de generar un catálogo universitario de simulaciones numéricas, las cuales serán ejecutadas remotamente a través de una interfaz gráfica y almacenadas en una base de datos, que estará disponible para toda la comunidad científica internacional. En este trabajo aplicamos dichos conceptos a cálculos numéricos que están vinculados a un problema astrofísico particular: Evolución de flujo supersónico de gas de hidrógeno neutro.

## CAPÍTULO I INTRODUCCIÓN

### 1.1 Bases de datos

La tecnología de las bases de datos se ha escrito como una de las áreas de la ciencia de la computación y la información de más rápido desarrollo.

Como campo comercial ha cobrado importancia teórica y práctica. La cantidad total de datos encomendados a las bases de datos se mide en varios miles de millones de bytes; la inversión financiera al respecto alcanza una cifra igualmente enorme; y se puede afirmar que muchos miles de organizaciones dependen de la operación continuada y eficaz de un sistema de bases de datos.

¿Qué es un sistema de bases de datos? En esencia, no es más que un sistema de mantenimiento de registros basado en computadores, es decir, un sistema cuyo propósito general es registrar y mantener información. Tal información puede estar relacionada con cualquier cosa que sea significativa para la organización donde opera el sistema, en otras palabras, cualquier dato necesario para los procesos de toma de decisiones inherentes a la administración de esa organización.

El concepto de una base de datos, también se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

Desde el punto de vista de la informática, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Una base datos, en un repositorio es tanto *integrada* como *compartida*.

Por “integrada” se entiende que la base de datos puede considerarse como una unificación de varios archivos de datos independientes, donde se elimina parcial o totalmente cualquier redundancia entre los mismos; por ejemplo EMPLEADO, que incluyen el nombre, la dirección, el departamento, el salario, etc., y registros de INSCRIPCIÓN, que representan inscripciones de empleados en cursos de capacitación. Suponiendo que se administra la información de los cursos, se necesita conocer el departamento de cada empleado inscrito, se alcanza a notar que no hay necesidad de incluir esta información en los registros de INSCRIPCIÓN, porque se puede obtener recurriendo a los registros de EMPLEADO correspondientes a nuestra búsqueda.

Por “compartida” se entiende que partes individuales de la base de datos pueden compartirse entre varios usuarios distintos, en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la base de datos y utilizarla con propósitos diferentes. Tal compartimiento es en verdad consecuencia del hecho de que la base de datos es integrada; en el ejemplo de EMPLEADO/INSCRIPCIÓN citado antes, la información sobre los departamentos en los registros de EMPLEADO es compartida por usuarios del departamento de personal y del departamento de educación. Otra consecuencia del mismo hecho (que la base de datos es integrada) se advierte en que cualquier usuario específico, por lo general, tendrá acceso sólo a algún subconjunto de la base de datos completa; además, subconjuntos de diferentes usuarios se trasladarán de muy diversas maneras, es decir, diferentes usuarios percibirán de modos muy distintos una base de datos específica (aunque dos usuarios compartan el mismo subconjunto de la base de datos, sus percepciones o vistas de ese subconjunto pueden diferir mucho a nivel de detalle).

Una base de datos tiene mucha importancia en el ritmo de vida que llevamos en los actuales momentos, ya que, está acelera el ritmo en el momento de realizar una búsqueda de información.<sup>1</sup>

### 1.1.1 Tipos de bases de datos

Las bases de datos pueden clasificarse de distinta manera, de acuerdo al criterio elegido para su clasificación.

1. Según la variabilidad de los datos almacenados.
  - a) **Estáticas:** Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.
  - b) **Dinámicas:** Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.
2. Según el contenido
  - a) **Bases de datos bibliográficas:** Solo contienen un surrogante (representante) de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque sino estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.
  - b) **Bases de datos de texto completo:** Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.
  - c) **Directorios:** Almacenan registros como pueden ser las guías telefónicas.
  - d) **Bases de datos multimedia:** Funcionan como banco de imágenes, sonido, video, etc.
  - e) **Bases de datos o bibliotecas de información:** Son aquellas que almacenan diferentes tipos de información. Puede concebirse que esta clasificación sea un conjunto de todas las demás<sup>2</sup>.

---

<sup>1</sup> Date, C.J., "Introducción a los sistemas de bases de datos", Traducc. Jaime Malpica y Américo Vargas, México, Sistemas Técnicos de Edición, 1986, p. 5-20.

### 1.1.2 Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como "contenedor de datos" (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos son:

#### a) Bases de Datos Jerárquicas

El modelo jerárquico fue desarrollado para permitir la representación de aquellas situaciones de la vida real en las que predominan las relaciones de tipo 1 : N.

Es un modelo muy rígido en el que las diferentes entidades de las que está compuesta una determinada situación, se organizan en niveles múltiples de acuerdo a una estricta relación PADRE/HIJO, de manera que un padre puede tener más de un hijo, todos ellos localizados en el mismo nivel, y un hijo únicamente puede tener un padre situado en el nivel inmediatamente superior al suyo.

Esta estricta relación PADRE/HIJO implica que no puedan establecerse relaciones entre segmentos dentro de un mismo nivel. La representación gráfica de un modelo jerárquico se realiza mediante la estructura de ARBOL INVERTIDO, en la que el nivel superior está ocupado por una única entidad, bajo la cual se distribuyen el resto de las entidades en niveles que se van ramificando. Los diferentes niveles quedan unidos por medio de las relaciones.

Las entidades se denominan en el caso particular del modelo jerárquico SEGMENTOS, mientras que los atributos reciben el nombre de CAMPOS.

Los segmentos, se organizan en niveles de manera que en un mismo nivel estén todos aquellos segmentos que dependen de un segmento de nivel inmediatamente superior.

Los datos se almacenan en la forma de registros, y cada uno de estos consta de un conjunto de campos. Un conjunto de registros con los mismos campos se denomina fichero (*record type*, en inglés).

El modelo jerárquico facilita relaciones padre-hijo, es decir, relaciones 1:N (de uno a varios). Pero las relaciones son unidireccionales. En justicia, dichas relaciones son hijo-padre, pero no padre-hijo. Por ejemplo, el registro de un empleado (nodo hijo) puede relacionarse con el registro de su departamento (nodo padre), pero no al contrario. Esto implica que solamente se puede consultar la base de datos desde los nodos hoja hacia el nodo raíz. La consulta en el sentido contrario requiere una búsqueda secuencial por todos los registros de la base de datos (por ejemplo, para consultar todos los empleados de un departamento). En las bases de datos jerárquicas no existen índices que faciliten esta tarea.

Las relaciones se establecen mediante punteros entre registros. Es decir, un registro hijo contiene la dirección física en el medio de almacenamiento de su registro padre. Esto

---

<sup>2</sup><http://64.233.167.104/search?q=cache:yY2ULQajadAJ:www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf+tipos+de+bases+de+datos&hl=es&ct=clnk&cd=7&gl=mx>

tiene una ventaja fundamental: “el rendimiento”. El acceso de un registro a otro es prácticamente inmediato sin necesidad de consultar tablas de correspondencia.

Los segmentos, en función de su situación en el árbol y de sus características, pueden denominarse como:

1. **Segmento padre:** Es aquél que tiene descendientes, todos ellos localizados en el mismo nivel (Fig. 1.1).

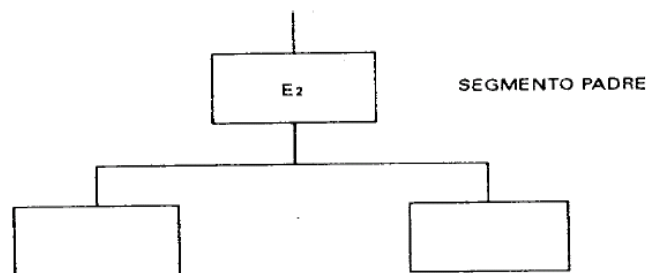


Fig. 1.1 Segmento Padre

2. **Segmento hijo:** Es aquél que depende de un segmento de nivel superior. Todos los hijos de un mismo padre están en el mismo nivel del árbol (Fig. 1.2).

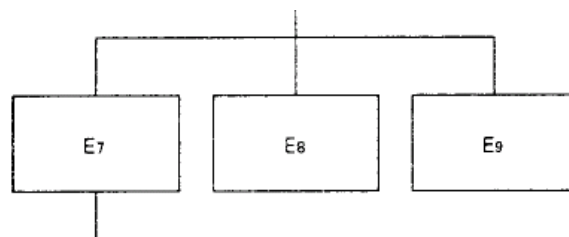


Fig. 1.2 Segmento Hijo

3. **Segmento raíz:** El segmento raíz de una base de datos jerárquica es el padre que no tiene padre. La raíz siempre es única y ocupa el nivel superior del árbol (Fig. 1.3).

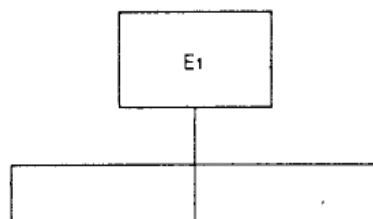


Fig. 1.3 Segmento Raíz

Las relaciones jerárquicas entre diferentes tipos de datos pueden hacer que sea muy sencillo responder a determinadas preguntas, pero muy difícil el contestar a otras.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos, es decir, no se garantiza que dos registros cualesquiera tengan diferentes valores en un subconjunto concreto de campos.<sup>3</sup>

## b) Bases de Datos de Red

El modelo de datos en red general representa las entidades en forma de nodos de un grafo, y las interrelaciones entre estas mediante arcos que unen dichos nodos. En principio esta representación no impone restricción alguna acerca del tipo y el número de arcos que puede haber, con lo que se pueden modelar estructuras de datos tan complejas como sea necesario.

Para definir el modelo de red general con cierta formalización, lo haríamos como un conjunto finito de tipos de entidades:

{E1, E2, ..., En},

Con sus respectivas propiedades o atributos:

{A11, A12, ..A1k, ..., An1, An2, ..., Anm},

Y un conjunto finito de tipos de interrelaciones:

{I<sub>h</sub>  
j,k, ..., n,}

La anterior notación representa la interrelación entre los elementos j, k, ...n, que a su vez pueden ser entidades o interrelaciones, y el superíndice h permite diferenciar dos interrelaciones distintas entre los mismos elementos, ya que se refiere al nombre de la interrelación (Fig. 1.4).

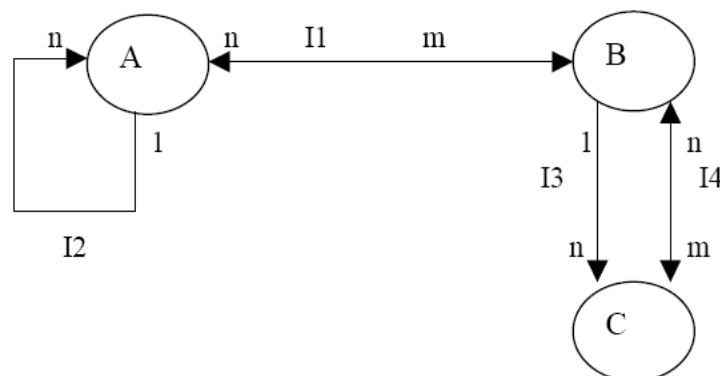


Fig. 1.4 Representación de cómo es el modelo de red.

<sup>3</sup> [http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001\\_MAMoraga.pdf](http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_MAMoraga.pdf)

Este modelo, se forma de una componente estática y otra dinámica. La estática estaría compuesta por los objetos (entidades o nodos y atributos), las interrelaciones o arcos y las restricciones, que a su vez pueden ser inherentes (no tenemos en este modelo) y de usuario (pueden ser reconocidas por el modelo de datos o de responsabilidad exclusiva del usuario). Dentro de la componente estática podemos citar un elemento más que atendería a la representación, y son los grafos. Por otro lado, la componente dinámica estaría compuesta por el aspecto de navegación. El esquema en si representa los aspectos estáticos, es decir, la estructura de los datos, que comprende los tipos de entidades, interrelaciones, etc. Una ocurrencia del esquema son los valores que toman los elementos del esquema en un determinado momento. Estos valores irán variando a lo largo del tiempo debido a la aplicación de los operadores de manipulación de datos a una ocurrencia del esquema.

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico). Una estructura de datos de red, llamada algunas veces estructura de plex, abarca más que la estructura de árbol porque un nodo hijo en la estructura red puede tener más de un padre. En otras palabras, la restricción de que en un árbol jerárquico cada hijo puede tener un solo padre, se hace menos severa.

Por lo visto hasta ahora, el modelo es muy flexible debido a la inexistencia de restricciones inherentes. Esto implica dificultad a la hora de implementarlo físicamente y a la larga será poco eficiente. Es por esto que el modelo sea tan solo teórico, y que a la hora de llevarlo a la práctica se introduzcan restricciones<sup>4</sup>.

### **c) Bases de Datos Relacional**

Una base de datos relacional es un conjunto de dos o más tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre si por un campo en común, en ambos casos posee las mismas características como por ejemplo el nombre de campo, tipo y longitud; a este campo generalmente se le denomina ID, identificador o clave. A esta manera de construir bases de datos se le denomina modelo relacional.

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos. La información puede ser recuperada o almacenada por medio de «consultas» que ofrecen una amplia flexibilidad y poder para administrar la información. El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, *Structured Query Language* o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Las bases de datos relacionales pasan por un proceso al que se le conoce como normalización de una base de datos, la cual es entendida como el proceso necesario para que una base de datos sea utilizada de manera óptima.

En términos tradicionales una relación se asemeja a un archivo, una tupla a un registro, y un atributo a un campo. Pero estas correspondencias son aproximadas, en el mejor de

---

<sup>4</sup> [http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001\\_IGarcia.pdf](http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_IGarcia.pdf)

los casos. Una relación no debe considerarse como "solo un archivo", sino más bien como un archivo disciplinado, siendo el resultado de esta disciplina una simplificación considerable de las estructuras de datos con las cuales debe interactuar el usuario, lo cual a su vez simplifica los operadores requeridos para manejar esas estructuras.

Características principales de los "archivos" relacionales:

- Cada "archivo" contiene solo un tipo de registros.
- Los campos no tienen un orden específico, de izquierda a derecha.
- Los registros no tienen un orden específico, de arriba hacia abajo.
- Cada campo tiene un solo valor.
- Los registros poseen un campo identificador único (o combinación de campos) comúnmente llamado clave primaria.

Así, todos los datos en una base de datos relacional se representan de una y solo una manera, a saber, por su valor explícito (esta se denomina en ocasiones "principio básico del modelo relacional"). En particular, las conexiones lógicas dentro de una relación y entre las relaciones se representan mediante esos valores; no existen "ligas" o apuntadores, ni ordenamientos, tampoco grupos repetitivos visibles para el usuario, etc.

Actualmente algunos de los manejadores de bases de datos, utilizan un sistema de búsqueda con algoritmos de árboles-B (árboles de búsqueda). Pero las búsquedas que se pueden realizar con estos algoritmos son sólo para memoria principal<sup>5</sup>.

#### **d) Bases de Datos Orientadas a Objetos**

Las bases de datos orientadas a objetos (BDOO) son aquellas cuyo modelo de datos está orientado a objetos y almacenan y recuperan objetos en los que se almacena estado y comportamiento. Su origen se debe a que en los modelos clásicos de datos existen problemas para representar cierta información, puesto que aunque permiten representar gran cantidad de datos, las operaciones que se pueden realizar con ellos son bastante simples.

Las clases utilizadas en un determinado lenguaje de programación orientado a objetos son las mismas clases que serán utilizadas en una BDOO; de tal manera, que no es necesaria una transformación del modelo de objetos para ser utilizado por un SGBDOO. De forma contraria, el modelo relacional requiere abstraerse lo suficiente como para adaptar los objetos del mundo real a tablas.

Las bases de datos orientadas a objetos nacen para evitar los problemas que surgen al tratar de representar cierta información, aprovechar las ventajas del paradigma orientado a objetos en el campo de las bases de datos y para evitar transformaciones entre modelos de datos (usar el mismo modelo de objetos).

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

---

<sup>5</sup> <http://www.fisimat.umich.mx/~elizalde/tesis/node15.html>



Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

1. Encapsulación - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
2. Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
3. Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.<sup>6</sup>

### 1.2.3 Sistemas Gestores de bases de datos

Un sistema gestor de bases de datos o SGBD (aunque se suele utilizar más a menudo las siglas *DBMS* procedentes del inglés, *Data Base Management System*) es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos.

En estos sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos, garantizando además la seguridad de los mismos (Fig. 1.5).

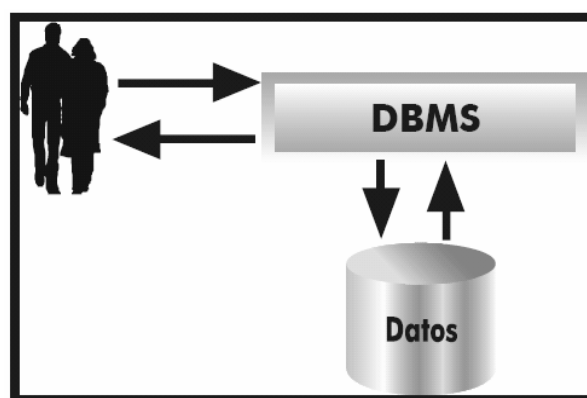


Fig. 1.5 Esquema del funcionamiento y utilidad de un Sistema gestor de bases de datos.

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de información. Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades

<sup>6</sup> <http://alarcos.inf-cr.uclm.es/doc/bbddavanzadas/Funcionalidad%201.pdf>

del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad.

Para ser más preciso, los objetivos que deben de cumplir los SGBD son:

1. **Abstracción de la información.** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
2. **Independencia.** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
3. **Redundancia mínima.** Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante, ya que lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
4. **Consistencia.** En aquellos casos en los que no se ha logrado esta redundancia nula, es necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
5. **Seguridad.** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
6. **Integridad.** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
7. **Respaldo y recuperación.** Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
8. **Control de la concurrencia.** En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos para recuperar información o para almacenarla, y es también frecuente que dichos accesos se realicen de forma simultánea, por eso un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
9. **Tiempo de respuesta.** Es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Se puede mencionar que el SGBD tiene como ventajas la gran velocidad en poco tiempo, independencia del tratamiento de información, facilidad del manejo de grandes volúmenes de información, seguridad de la información, etc.

Pero como todo, también tiene sus desventajas como son el costo de actualización con respecto a hardware y software, salario del administrador de base de datos, y una serie de problemas que acarrearía un mal diseño de la base de datos.

Existen muchos gestores, y estos se pueden dividir en 2; los libres y los comerciales, y por mencionar algunos de ambas divisiones están: *Oracle*, *MySQL*, *SQL Server*, *PostGreeSQL*, *Sybase*, etc.<sup>7</sup>

#### **1.1.4 Normalización de bases de datos**

La normalización es el proceso de organizar los datos en una base de datos. Esto incluye la creación de tablas y que se establezcan relaciones entre aquellas tablas, de acuerdo a las reglas diseñadas para proteger los datos, hace a la base de datos más flexible ya que elimina la redundancia y dependencia incoherente.

Los datos redundantes desperdician espacio en disco y crean problemas de mantenimiento. Si es necesario cambiar datos que aparecen en más de un sitio, el cambio deberá ser exactamente igual en todos estos sitios. Por ejemplo, un cambio de dirección de un cliente es mucho más fácil de implementar si los datos sólo se almacenan en la tabla Clientes y en ningún otro lugar de la base de datos.

Existen unas cuantas reglas para la normalización de bases de datos. Cada regla se denomina "forma normal". Si se cumple la primera regla, se dice que la base de datos está en la "primera forma normal". Si se cumplen las tres primeras reglas, se considera que la base de datos está en la "tercera forma normal". Aunque existen otros niveles de normalización, se considera que la tercera forma normal es el máximo nivel necesario para la mayoría de las aplicaciones.

##### **a) Primera Forma Normal (1FN)**

La primera forma normal (1FN) es una restricción inherente al modelo relacional, por lo que su cumplimiento es obligatorio y afecta al número de valores que pueden tomar los atributos de una relación.

Para que una tabla pueda ser considerada una relación no debe admitir grupos repetitivos, esto es, debe de estar en primera forma normal, esto es, que una relación esta en 1FN cuando cada atributo solo toma un valor del dominio simple subyacente.

Por lo anterior se puede decir que la 1FN tiene por objetivos:

1. Eliminar grupos repetidos en tablas individuales.
2. Crear una tabla diferente para cada conjunto de datos relacionados.
3. Identificar cada conjunto de datos relacionados mediante una clave principal.

##### **b) Segunda Forma Normal (2FN)**

La segunda forma normal (2FN) esta basada en el concepto de dependencia plena y en las interrelaciones existentes entre los atributos principales (que se encuentran en alguna de las clases) y no principales (que no se encuentran en ninguna clave) de una relación. Se dice que una relación esta en 2FN si:

1. Está en 1FN

---

<sup>7</sup> <http://www.jorgesanchez.net/bd/docs2006/sgbd.pdf>

2. Cada atributo no principal tiene dependencia funcional completa respecto a cada una de las claves.

La segunda forma normal no se cumple cuando algún atributo no principal depende funcionalmente de algún subconjunto de la clave.

Se puede decir que los objetivos de 2FN son:

1. Eliminar grupos repetidos en tablas individuales.
2. Crear una tabla diferente para cada conjunto de datos relacionados.
3. Identificar cada conjunto de datos relacionados mediante una clave principal.

### c) Tercera Forma Normal (3FN)

La tercera forma Normal (3FN) esta basada en el concepto de dependencia transitiva.

Un esquema de relación R esta en tercera forma normal si, y solo si:

1. Está en 2FN.
2. No existe ningún atributo no principal que dependa transitivamente de alguna clave R.

La tercera forma normal no se cumple cuando existen atributos no principales que dependen funcionalmente de otros atributos no principales. Esta forma normal tiene como principal objetivo eliminar los campos que no dependan de la clave.

### d) Otras Formas Normales

Existe una cuarta forma normal, llamada también Forma normal de Boyce Codd (BCNF), y una quinta forma normal, pero pocas veces se consideran prácticas en un diseño. La omisión de estas reglas puede dar como resultado una tabla que no sea perfecta, pero no debería afectar a su funcionamiento.<sup>8</sup>

## 1.2 Herramientas de Programación

En el desarrollo de un proyecto informático uno de los aspectos más importantes es la elección del lenguaje más apropiado para su implementación y las herramientas a utilizar. Existen múltiples herramientas informáticas y múltiples lenguajes de programación dependiendo del área de estudio. Una vez elegido el lenguaje, se ha de recabar información sobre las herramientas disponibles y realizar un estudio acerca de cuál se adapta mejor a las necesidades del proyecto a realizar.

Es por eso que a continuación se describen las herramientas que serán ocupadas en el desarrollo del portal, independientemente de cual sea la elección del lenguaje de programación y el gestor de la base de datos.

---

<sup>8</sup> Adoración de Miguel, Piattini, Mario, et. al., “*Diseño de bases de datos relacionales*”, México, Alfaomega, 2000, p 147-160.

### 1.2.1 Programas y algoritmos

Un algoritmo es un método para resolver un problema. Para la creación de un programa (resolver un problema) en un lenguaje de programación hay que sobrepasar al menos cuatro pasos a saber:

1. Definición o análisis del problema: Consiste en conocer a detalle de lo que se va a tratar, es decir, antes de dar el paso siguiente es necesario saber cuál es el problema y como enfrentarlo, buscar las posibles soluciones y escoger la más fiable y/o menos costosa.
2. Diseño del algoritmo: que describe la secuencia ordenada de pasos – sin ambigüedades – que conduzcan a la solución de un problema dado.
3. Expresar el algoritmo como un programa en un lenguaje de programación adecuado.
4. Ejecución y validación del programa por la computadora.

Para llegar a la realización de un programa es necesario el diseño previo de un algoritmo, de modo que sin algoritmo no puede existir un programa. Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta. Sus características fundamentales son:

- a) Debe ser preciso e indicar el orden de realización de cada paso.
- b) Debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- c) Debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea debe tener un número finito de pasos.
- d) Ha de cumplir tres partes: Entrada – Proceso – Salida.

Un programa normalmente implementa (traduce a un lenguaje de programación concreto) un algoritmo. Puede haber programas que no se ajusten a un algoritmo (pueden no terminar nunca), en cuyo caso se denomina procedimiento a tal programa.

Los programas suelen subdividirse en partes menores (módulos), de modo que la complejidad algorítmica de cada una de las partes sea menor que la del programa completo, lo cual ayuda al desarrollo del programa.

Según Niklaus Wirth (científico de la computación) un programa está formado por algoritmos y estructura de datos.<sup>9</sup>

---

<sup>9</sup> <http://www.geocities.com/ymarte/trab/infalgprg.html>

### 1.2.2 Diagramas de flujo

Los diagramas de flujo (o flujogramas) son diagramas que emplean símbolos gráficos para representar los pasos o etapas de un proceso. También permiten describir la secuencia de los distintos pasos o etapas y su interacción.

La creación del diagrama de flujo es una actividad que agrega valor, pues el proceso que representa está ahora disponible para ser analizado, no sólo por quienes lo llevan a cabo, sino también por todas las partes interesadas que aportarán nuevas ideas para cambiarlo y mejorarlo.

Se utilizan principalmente en programación, economía y procesos industriales; estos diagramas utilizan una serie de símbolos con significados especiales. Son modelos tecnológicos utilizados para comprender los rudimentos de la programación lineal.

Otra definición del diagrama de flujo es la siguiente:

"Es un esquema para representar gráficamente un algoritmo. Se basan en la utilización de diversos símbolos para representar operaciones específicas. Se les llama diagramas de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación. Para hacer comprensibles los diagramas a todas las personas, los símbolos se someten a una normalización; es decir, se hicieron símbolos casi universales, ya que, en un principio cada usuario podría tener sus propios símbolos para representar sus procesos en forma de Diagrama de Flujo. Esto trajo como consecuencia que sólo aquel que conocía sus símbolos, los podía interpretar. La simbología utilizada para la elaboración de diagramas de flujo es variable y debe ajustarse a un patrón definido previamente."

#### Ventajas de los Diagramas de Flujo

1. Favorecen la comprensión del proceso a través de mostrarlo como un dibujo. El cerebro humano reconoce fácilmente los dibujos. Un buen diagrama de flujo reemplaza varias páginas de texto.
2. Permiten identificar los problemas y las oportunidades de mejora del proceso. Se identifican los pasos redundantes, los flujos de los reprocesos, los conflictos de autoridad, las responsabilidades, los cuellos de botella, y los puntos de decisión.
3. Muestran las interfases cliente-proveedor y las transacciones que en ellas se realizan, facilitando a los empleados el análisis de las mismas.
4. Son una excelente herramienta para capacitar a los nuevos empleados y también a los que desarrollan la tarea, cuando se realizan mejoras en el proceso.

#### Principales Símbolos

No es indispensable usar un tipo especial de símbolos para crear un diagrama de flujo, pero existen algunos ampliamente utilizados por lo que es adecuado conocerlos y utilizarlos, ampliando así las posibilidades de crear un diagrama más claro y comprensible para crear un proceso lógico y con opciones múltiples adecuadas.

Flecha. Indica el sentido y trayectoria del proceso de información o tarea.

**Rectángulo.** Se usa para representar un evento o proceso determinado. Éste es controlado dentro del diagrama de flujo en que se encuentra. Es el símbolo más comúnmente utilizado.

**Rectángulo redondeado.** Se usa para representar un evento que ocurre de forma automática y del cuál generalmente se sigue una secuencia determinada.

**Rombo.** Se utiliza para representar una condición. Normalmente el flujo de información entra por arriba y sale por un lado si la condición se cumple o sale por el lado opuesto si la condición no se cumple. Lo anterior hace que a partir de éste el proceso tenga dos caminos posibles.

**Círculo.** Representa un punto de conexión entre procesos. Se utiliza cuando es necesario dividir un diagrama de flujo en varias partes, por ejemplo por razones de espacio o simplicidad. Una referencia debe darse dentro para distinguirlo de otros. La mayoría de las veces se utilizan números en los mismos.

Existen además una variedad de formas especiales para denotar las entradas, las salidas, los almacenamientos, etcétera.

## Reglas

De acuerdo al estándar ISO, los símbolos e incluso las flechas deben de tener ciertas características para estar dentro del estándar. En el caso de los círculos de conexión se debe usar sólo cuando se conecta con un proceso contenido dentro de la misma hoja.

También existen conectores de página, que son como una casita y se utilizan para unir actividades que se encuentran en otra hoja.<sup>10</sup>

1. Existe siempre un camino que permite llegar a una solución
2. Existe un único inicio del proceso
3. Existe un único punto de fin para el proceso de flujo, salvo del rombo que indica una comparación con dos caminos posibles y además una gran ayuda
4. Evite sumideros infinitos, burbujas que tienen entradas pero no salidas.
5. Evite las burbujas de generación espontánea, que tienen salidas sin tener entradas, porque son sumamente sospechosas y generalmente incorrectas.

### 1.2.3 Programación e ingeniería del software

Existe una tendencia a identificar el proceso de creación de un programa informático con la programación, que es cierta cuando se trata de programas pequeños para uso personal, y que dista de la realidad cuando se trata de grandes proyectos.

La Ingeniería de Software es el establecimiento y uso de principios de ingeniería para obtener software que sea confiable y que funcione eficientemente en máquinas reales. Además esta como también las disciplinas tradicionales de ingeniería, tiene que ver con el costo y la confiabilidad. Algunas aplicaciones de software contienen millones de líneas de código que se espera que se desempeñen bien en condiciones siempre cambiantes.

---

<sup>10</sup> [http://www.elprisma.com/apuntes/administracion\\_de\\_empresas/quesonlosdiagramasdeflujo/](http://www.elprisma.com/apuntes/administracion_de_empresas/quesonlosdiagramasdeflujo/)

En un nivel más técnico, la Ingeniería de Software comienza con una serie de tareas que hacen modelos y que resultan en una especificación completa de requisitos y una representación comprensiva de diseño del software que será construido. Se han desarrollado muchos métodos para hacer modelos de sistemas de información. Sin embargo, los métodos Orientados a Objeto (OO) van a llegar a ser el estándar.<sup>11</sup>

El proceso de creación de software desde el punto de vista de la Ingeniería tiene los siguientes pasos:

1. Reconocer la necesidad de un programa para solucionar un problema ó identificar la posibilidad de automatización de una tarea.
2. Recoger los requisitos del programa. Debe quedar claro qué es lo que debe hacer el programa y para qué se necesita.
3. Realizar el análisis de los requisitos del programa. Debe quedar claro cómo debe realizar el programa las cosas que debe hacer. Las pruebas que comprueben la validez del programa se pueden especificar en esta fase.
4. Diseñar la arquitectura del programa. Se debe descomponer el programa en partes de complejidad abordable.
5. Implementar el programa. Consiste en realizar un diseño detallado, especificando completamente todo el funcionamiento del programa, tras lo cual la codificación debería resultar inmediata.
6. Implantar (instalar) el programa. Consiste en poner el programa en funcionamiento junto con los componentes que pueda necesitar (bases de datos, redes de comunicaciones, etc.).

La Ingeniería del Software se centra en los pasos de planificación y diseño del programa, mientras que antiguamente (programación artesanal) la realización de un programa consistía únicamente en escribir el código.<sup>12</sup>

#### 1.2.4 Hojas de estilo

Las hojas de estilo son un estándar para la *web* que se empezó a aplicar a partir de los navegadores en versión 3 y 4. Es necesario remarcar que Explorer es mucho más avanzado en este aspecto que otros navegadores, con soporte mucho más completo y fiable de las especificaciones de las hojas de estilo. Sin duda, versiones posteriores aún mejorarán más este soporte.

Las hojas de estilo para páginas *web html* se denominan en inglés *Cascading Style Sheets*, y se conocen mejor por sus siglas (CSS). Son "en cascada", puesto que se puede definir el estilo a diferentes niveles, cada uno de los cuales se impone al anterior. La forma en que se define el estilo es semejante al lenguaje de *html*, en el sentido de que consta de una serie de códigos que se redactan en forma de texto simple y se incorporan a la página *web* o se enlazan en forma de un fichero separado, con la extensión .CSS.

El poder que tiene esta forma de definir el formato de las páginas es enorme, ya que es posible cambiar el aspecto de una web entera, con todas las páginas que tenga, con sólo modificar una sola hoja de estilo. Es decir, puede convertir la tarea de reformatear todo

---

<sup>11</sup> <http://www.sistemas.unam.mx/software.html>

<sup>12</sup> <http://es.wikipedia.org/wiki/Programaci%C3%B3n>



en coser y cantar. Aparte de la comodidad y poder que tiene usar este sistema de formato, las hojas de estilo son una herramienta bastante flexible que permite hacer cosas imposibles con el *html* por sí solo.<sup>13</sup> Por ejemplo:

1. Los enlaces en una página, con *html* son siempre del mismo color. Con una hoja de estilo podemos definir tantos tipos de enlace como deseemos. Por ejemplo, un estilo para una barra de menú, otro para la tabla lateral, otro para el pie de la página y otro para los enlaces dentro del cuerpo del texto.
2. El efecto de cambio de color y estilo que se observa al pasar el ratón por encima de un enlace de texto en *Internet Explorer* se crea -muy fácilmente- con hojas de estilo.
3. En *html* los enlaces aparecen siempre subrayados. Algo que se atraganta a muchos diseñadores, y no se puede cambiar. Pero con una simple orden de la hoja de estilo, aparecerán sin subrayar.
4. Las hojas de estilo permiten un control tipográfico muy completo, en comparación con el basto manejo de las fuentes que hace el *html*. Se puede definir con mucha más precisión y variedad el tamaño, el estilo, el interlineado, aplicar color o imagen de fondo...
5. Se puede controlar con precisión la disposición de los elementos de la página y, si es preciso, se pueden fijar posiciones absolutas e invariables.
6. Junto con el lenguaje de programación *Javascript*, las hojas de estilo permiten crear páginas dinámicas, sin que sean necesarios gráficos: elementos móviles, menús desplegables, etcétera (para mayor referencia consultar el sitio <http://www.w3.org>).

### 1.3 Bases de datos y programación orientada al Web

Desde hace mucho tiempo atrás, las comunicaciones son de lo más importante para las sociedades humanas, para el hombre, la comunicación es una necesidad que debe de cubrir y lo ha sabido hacer muy bien, ya que actualmente se cuenta con una infinidad de sistemas de comunicación, desde ya los viejos señalamientos de tránsito, hasta las modernas tecnologías de comunicación a larga distancia, como un teléfono celular.

Cuando hablamos de comunicaciones hacemos referencia a muchas tecnologías, tanto del siglo pasado como de este. La década pasada en México, hablar de Internet era hablar de algo relevante referente a las comunicaciones, en esta década y las siguientes, hablar de Internet será hablar de lo más normal en comunicaciones, ya que el futuro estará fuertemente ligado a este medio de comunicación.

#### 1.3.1 Diseño de Páginas

El diseño *web* es una actividad que consiste en la planificación, diseño e implementación de sitios *web* y páginas *web*. No es simplemente una aplicación del diseño convencional sobre Internet ya que requiere tener en cuenta cuestiones tales como navegabilidad, interactividad, usabilidad, arquitectura de la información y la interacción de medios como el audio, texto, imagen y vídeo, además no sólo aporta a la comunicación textual (contenidos) existente en Internet una faceta visual, sino que

---

<sup>13</sup> [http://platea.pntic.mec.es/jmas/manual/html/estilo\\_con\\_css.html](http://platea.pntic.mec.es/jmas/manual/html/estilo_con_css.html)

obliga a pensar una mejor estructuración de los mismos en este soporte. La unión de un buen diseño con una jerarquía bien elaborada de contenidos aumenta la eficiencia del sitio como canal de comunicación e intercambio de datos, que brinda posibilidades como el contacto directo entre el productor y el consumidor de contenidos, característica destacable del medio Internet.

El diseño de páginas *web* es una amplia área de aplicación del diseño gráfico en la cual se integran conocimientos propios del diseño como son la composición, el uso de color y la tipografía con conocimientos técnicos del medio como son los lenguajes *html* (*HiperText Markup Language*) y *CSS* (*Cascading Style Sheets*), así como conocimientos sobre usabilidad, accesibilidad y organización de un sitio Web.

También se trata básicamente de realizar un documento con información hiper-enlazado con otros documentos y asignarle una presentación para diferentes dispositivos de salida (en una pantalla de ordenador, en papel, en un teléfono móvil, etc.).

Para el diseño de páginas *web* debemos tener en cuenta dos etapas:

La primera es el diseño visual de la información que se desea editar. En esta etapa se trabaja en el papel distribuyendo el texto, los gráficos, los vínculos a otros documentos y otros objetos multimedia que se consideren pertinentes.

La segunda, una vez que se tiene este boceto se pasa a 'programar' la página *web*. Para esto, y fundamentalmente para manejar los vínculos entre documentos, se creó el lenguaje de marcación de hipertexto o *html*. Los enlaces que aparecen subrayados en algún sitio son ejemplos de hipertexto, puesto que al pulsar sobre ellos conducen a otras páginas con información relacionada.

Un correcto diseño *web* implica conocer cómo se deben utilizar cada una de las etiquetas permitidas en *html*, es decir, hacer un uso correcto de este lenguaje dentro de los estándares establecidos por la W3C (consorcio internacional de estándares para la WWW) y en lo referente a la *web* semántica. Debido a los permisos que otorgan algunos navegadores *web* como *Internet Explorer*, se ha perdido la primicia original. Por ejemplo, este navegador permite que no sea necesario cerrar las etiquetas del marcado, utiliza código propietario, etc. Esto impide que ese documento *web* sea universal e independiente del medio que se utilice para ser mostrado.

La *web* semántica, por otra parte, aboga por un uso lógico de las etiquetas según el significado para el que fueron concebidas. Por ejemplo se utilizará la etiqueta <P> para marcar párrafos, y <TABLE> para tabular datos. En su última instancia, esto ha supuesto una auténtica revolución en el diseño *web* puesto que apuesta por separar totalmente el contenido del documento de la visualización.

De esta forma se utiliza el fichero *html* únicamente para organizar y estructurar la información y las hojas de estilo *CSS* para indicar como se mostrará dicha información en los diferentes medios (como por ejemplo, una pantalla de ordenador, un móvil, impreso en papel, leída por un sintetizador de voz, etc...). Por lógica, esta metodología beneficia enormemente la accesibilidad del documento.

También existen páginas dinámicas, las cuales permiten mayor interactividad entre la *web* y el visitante, proporcionándole herramientas dinámicas tales como buscadores, chat, foros, sistema de encuestas, etc. y poseen de un panel de control de administración de contenidos. Este permite crear, actualizar y administrar cantidades ilimitadas de contenido en la misma.<sup>14</sup>

### 1.3.2 Lenguajes de marcas

Todo comenzó con la intención de estandarizar el formato en el que los diversos documentos se encuentran en las computadoras. A cualquiera de nosotros nos ha pasado alguna vez que alguien nos envíe, por ejemplo, un texto para revisar y no podemos verlo debido a que está hecho con un programa de texto distinto al que tenemos instalado en nuestra computadora. O podemos verlo, pero con un aspecto bastante distinto al que quería darle su autor. En eso, seguramente, estaba pensando Charles F. Goldfab cuando en los años 70 la empresa estadounidense IBM le encargó el trabajo de "describir documentos".<sup>15</sup>

Los lenguajes de marcas se llaman así por la práctica tradicional de marcar los manuscritos con instrucciones de impresión en los márgenes. En la época de la imprenta, esta tarea ha correspondido a los marcadores, que indicaban el tipo de letra, el estilo y el tamaño, así como la corrección de errores, para que otras personas compusieran la tipografía. Esto condujo a la creación de un grupo de marcas estandarizadas. Con la introducción de los ordenadores, se trasladó un concepto similar al mundo de la informática.

Se suele diferenciar entre tres clases de lenguajes de marcado, aunque en la práctica pueden combinarse varias clases en un mismo documento, sin embargo se dividen en:

1. Marcado de presentación: Es aquel que indica el formato del texto.
2. Marcado de procedimientos: Esta enfocado hacia la presentación del texto
3. Marcado descriptivo o semántico: Utiliza etiquetas para describir fragmentos de texto, pero sin especificar como deben de ser representados.

Cabe mencionar que los lenguajes de marcas que sentarían las bases actuales son 4, en los cuales evoluciona uno del otro como se marca a continuación:

GML → SGML → XML → Dialectos XML

Como se aprecia el primero en surgir es GML, y evoluciona hasta tener a los dialectos XML, pero veamos en que consiste cada uno:

#### **GML**

GML ("*General Markup Language*") fue uno de los primeros lenguajes de marcación que fue diseñado para componer estructuras de datos descriptivas, esto es, un meta-lenguaje, que quiere decir estructuras de datos describiendo otras estructuras de datos.<sup>16</sup>

<sup>14</sup> [http://es.wikipedia.org/wiki/Dise%C3%B1o\\_de\\_p%C3%A1ginas\\_web](http://es.wikipedia.org/wiki/Dise%C3%B1o_de_p%C3%A1ginas_web)

<sup>15</sup> [http://www.inclusiondigital.net/albergue/dweb/leng\\_marcas.html](http://www.inclusiondigital.net/albergue/dweb/leng_marcas.html)

<sup>16</sup> <http://www.osmosislatina.com/xml/basico.htm>

## SGML

El GML evoluciona y se convierte en SGML que son las siglas de "*Standard Generalized Markup Language*" o "Lenguaje de Marcación Generalizado", que consiste en un sistema para la organización y etiquetado de documentos y sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.<sup>17</sup>

## XML

Aunque sus siglas en inglés *eXtensible Markup Language* («lenguaje de marcas extensible»), hagan suponer lo contrario, XML no es sólo un lenguaje, sino un conjunto de reglas y definiciones para crearlos. Por esta razón, XML es considerado como un metalenguaje. Dichas reglas se caracterizan por ser simples y estrictas. XML no está orientado a almacenar algún tipo de dato en particular, sus fines son generales y puede ser utilizado por cualquier tipo de organización. Fue creado y es mantenido por la W3C (*World Wide Web Consortium*) con el objeto de superar las limitaciones de otro lenguaje de marcado, *html*.

XML deriva a su vez de SGML, que, similar en cuanto a presentaciones y principios, perdió la oportunidad de convertirse en un estándar popular debido a su gran cantidad de opciones disponibles, lo que se tradujo en complejidad (en ocasiones innecesaria) para los desarrolladores. Suele decirse que XML, es algo así como SGML simplificado, de manera que una aplicación no necesita comprender SGML, por completo para interpretar un documento, sino sólo un subconjunto de las reglas que este define. Los editores y desarrolladores de SGML pueden comprender XML.

Las ventajas que ofrece XML son la facilidad de aprendizaje en base a sus reglas claras, además de ofrecer las características que fueron la base de su desarrollo:

1. Intercambio de datos.
2. Una estructura, múltiples presentaciones.
3. Búsquedas más específicas.
4. Programación en capas.
5. Jerarquías de la información.
6. Es auto descriptivo.

XML ofrece estructurar documentos, así como también validar los datos que ellos contienen, darles formato de salida elegante, extraer los datos que se necesiten, etc.<sup>18</sup>

## Dialectos XML

Se dice de un dialecto específico que sigue las reglas XML, como puede ser el XHTML, GML (Representación Geográfica), XBRL (Información financiera), etc.

<sup>17</sup> [http://www.inclusiondigital.net/albergue/dweb/leng\\_marcas.html](http://www.inclusiondigital.net/albergue/dweb/leng_marcas.html)

<sup>18</sup> Minera, Francisco, "*XML. La guía total del programador*", Argentina, MP Ediciones, 2006, p.14-30.

De aquí se toma la tecnología XHTML que es de uso para el Internet y el cual será la clave para poder desarrollar cualquier aplicación que se necesite, pero como se explica anteriormente es solo un dialecto de XML, por lo cual es necesario el *html*.

## HTML

El *html* es creado en 1991 por Tim Berners-Lee como una opción de transferencia de información para la iniciativa WWW. En 1993 Dan Conelly presenta el primer DTD (Definición en un documento que especifica restricciones en la estructura y sintaxis del mismo) de *html*, en el cual se describe el lenguaje y posteriormente en 1995, presenta una nueva versión, a la cual se le denomina *HTML*.

Pero para esa entonces el WWW no era ya un medio de transmisión de información educativo, si no que comienza la comercialización electrónica, obligando a innovar al *html* para otras necesidades, tales como, mayor interactividad y diseño atractivo, esto hace que Netscape líder en este ramo en aquel entonces realice nuevas opciones para la *web* y así, sus competidores, obligando al IETF (Grupo de Trabajo en Ingeniería de Internet) a realizar una nueva estandarización del *html*, pues eran ya muchas las opciones diferentes para cada visualizador de páginas, pero el estudio presentado era muy extenso y no fue aceptado comercialmente, así que las grandes compañías tales como IBM, Netscape, SUN, Microsoft, etc., crean una organización denominada W3C, que es el actual encargado de la designación de versiones del *html*.

Después de una breve historia de *html* pasemos a lo que significa y es *Hyper Text Markup Lenguaje* o lenguaje de marcas de hipertexto, es decir, documentos de texto realizados en forma estructurada que conducen a otros documentos o formas de información, tales como, archivos, videos, sonidos, imágenes, bases de datos, etc.

Este tipo de documentos es utilizado generalmente en WWW que es un conjunto de servidores ofrecidos en todo el mundo a través de Internet, el WWW esta basado en el protocolo HTTP (*Hyper Text Transfer Protocol*) que es el protocolo de transferencia de hipertexto mediante un modelo cliente-servidor.<sup>19</sup>

### 1.3.3 Integración de datos en el Web

En la actualidad, muchas instituciones se han dado cuenta de la importancia que el *web* tiene en el desarrollo de sus potencialidades, ya que con ello pueden lograr una mejor comunicación con personas o instituciones situadas en cualquier lugar del mundo.

Gracias a la conexión con la red mundial Internet, poco a poco, cada individuo o institución va teniendo acceso a mayor cantidad de información de las diversas ramas de la ciencia con distintos formatos de almacenamiento.

La mayor parte de información es presentada de forma estática a través de documentos *html*, lo cual limita el acceso a los distintos tipos de almacenamiento en que ésta pueda encontrarse, pero en la actualidad surge la posibilidad de utilizar aplicaciones que permitan acceder a información de forma dinámica, tal como a bases de datos, con contenidos y formatos muy diversos.

---

<sup>19</sup> Leal, Héctor, “*Manual de HTML*”, México, Centro de Investigación Cibernética, 2005, p. 76.

Una de las ventajas de utilizar el *web* para este fin, es que no hay restricciones en el sistema operativo que se debe usar, permitiendo la conexión entre sí, de las páginas *web* desplegadas en un navegador del *web* que funciona en una plataforma, con servidores de bases de datos alojados en otra plataforma. Además, no hay necesidad de cambiar el formato o estructura de la información dentro de las bases de datos.

Tradicionalmente en el *web* se han utilizado documentos *html* estáticos para los cuales se creaban las posibles respuestas ante requisiciones del cliente. Este método requiere de un gran desarrollo de aplicaciones y de mantenimiento de las mismas. Al interactuar con las bases de datos, este proceso se complica aún más.

Como la necesidad de acceder a bases de datos desde el *web* se ha incrementado, han sido creadas también interfaces que manipulan sus escritos para procesar la información, teniendo como punto común la ejecución de sentencias SQL para requerir datos a la base.

Aplicaciones de interfaz para la interacción de bases de datos con el *web* han surgido ya. Los productos iniciales son simplemente modelos del ambiente cliente/servidor, con una capa adicional para crear resultados *html* que pueden ser vistos a través del Web, por medio de un procesamiento de los datos de la forma introducidos por el cliente. Además, al usar estas interfaces se puede crear el programa principal de la aplicación, y estas herramientas permiten construir poderosas aplicaciones en el *web*, pero se requiere que programadores experimentados logren un desarrollo a gran escala. También, el mantenimiento de las mismas es significativamente más complejo y extenso. Una de las estrategias más famosas para la creación de aplicaciones de interacción con el *web*, es la de descargar del *web*, aplicaciones o componentes funcionales que se ejecutarán dentro del *browser*. Con ellas se realizará un procesamiento complejo del lado del cliente, lo cual requiere un gran esfuerzo para crear las piezas de la aplicación. Estas estrategias poseen dos características principales: garantizan la seguridad tanto en los sistemas de distribución como en la comunicación que se establece con tales aplicaciones, a través de Internet. Además han aparecido bibliotecas que incluyen motores propios de servidor que corren de forma conjunta con el Servidor Web, lo cual facilita el desarrollo de nuevas aplicaciones.

Una aplicación que posibilita interconectar al *web* con una base de datos tiene muchas ventajas, además de que las funciones que cumplen actualmente los Servidores Web y las herramientas de desarrollo de aplicaciones *web*, hacen más fácil que nunca la construcción de aplicaciones más robustas. Tal vez el mayor beneficio del desarrollo de estas sea la habilidad de que sean para múltiples plataformas, sin el costo de distribuir múltiples versiones del software. Cada una de las interfaces para comunicar al *web* con bases de datos, ha sido creada basándose en una tecnología de integración especial, a través de procesos de interconexión especiales.

Cuando se utiliza una interfaz para lograr la integración del *web* con cierta base de datos, se puede verificar que los procesos seguidos varían, dependiendo de la tecnología que se esté utilizando.<sup>20</sup>

Entre estas tecnologías se tienen las siguientes:

---

<sup>20</sup> <http://www.uca.edu.sv/investigacion/bdweb/tecnolog.html>

### **a) El Common Gateway Interface (CGI)**

Actualmente, ésta es la solución que más se está utilizando para la creación de interfaces Web/DBMS. Fue probada por primera vez en el servidor NCSA.

Se ha comprobado que si el Servidor Web recibe un URL con una llave, para devolver un documento *html* como respuesta, tendrá que cargar el servicio (programa) que le indique las variables de ambiente y de la forma *html*. La mayoría de las veces dicha llave es el "cgi-bin".

Entre las ventajas de la programación CGI, se tiene su sencillez, ya que es muy fácil de entender, además de ser un lenguaje de programación independiente, ya que los escritos CGI pueden elaborarse en varios lenguajes. También es un estándar para usarse en todos los servidores Web, y funcionar bajo una arquitectura independiente, ya que ha sido creado para trabajar con cualquier arquitectura de servidor Web.

Como la aplicación CGI se encuentra funcionando de forma independiente, no pone en peligro al servidor, en cuanto al cumplimiento de todas las tareas que éste se encuentre realizando, o al acceso del estado interno del mismo. Pero el CGI presenta cierta desventaja en su eficiencia, debido a que el Servidor Web tiene que cargar el programa CGI y conectar y desconectar con la base de datos cada vez que se recibe una requisición. Además, no existe un registro del estado del servidor, sino que todo hay que hacerlo manualmente.

### **b) Interfaz de Programación de Aplicaciones (API)**

Es un conjunto de rutinas, protocolos y herramientas para construir aplicaciones de interfaz. Una buena API hace más fácil el trabajo de desarrollo de un programa, ya que debe proveer todos los bloques para construirlo. El programador lo único que hace es poner todos los bloques juntos.

API está diseñado especialmente para los programadores, ya que garantiza que todos los programas que utilizan API, tendrán interfaces similares. Asimismo, esto le facilita al usuario aprender la lógica de nuevos programas. Cuando se realiza una requisición, el servidor llamará al API, brindando la ventaja de disponer de una mayor cantidad de servicios.

### **c) Interfaz de Programación de Aplicaciones del Servidor Internet (ISAPI)**

Es la interfaz propuesta por Microsoft como una alternativa más rápida que el CGI, y ya está incluida en el Servidor Microsoft Internet *Information* (IIS). Así como los escritos CGI, los programas escritos usando ISAPI habilitan un usuario remoto para ejecutar un programa, busca información dentro de una base de datos, o intercambia información con otro software localizado en el servidor.

Los programas escritos usando la interfaz ISAPI son compilados como bibliotecas de enlace dinámico (DLL - *Dynamic Link Library*), ya que son cargados por el servidor Web cuando éste se inicia. Dichos programas se vuelven residentes en memoria, por lo que se ejecutan mucho más rápido que las aplicaciones CGI, debido a que requieren menos tiempo de uso de CPU al no iniciar procesos separados. Uno de los programas ISAPI más usados es el HTTPODBC.DLL que se usa para enviar y/o devolver información hacia y desde las bases de datos, a través de ODBC.

Además, ISAPI permite realizar un procesamiento previo de la solicitud y uno posterior de la respuesta, con lo cual manipula la solicitud/respuesta HTTP. Los filtros ISAPI pueden utilizarse para aplicaciones tales como autenticación, acceso o apertura de sesión.<sup>21</sup>

---

<sup>21</sup> <http://www.uca.edu.sv/investigacion/bdweb/tecnolog.html>



## CAPÍTULO II ESTRUCTURA DEL PORTAL

### 2.1 Introducción

Como se planteo en los objetivos, la solución más óptima para la estructura del portal es hacer un sitio de Internet que tenga la capacidad de satisfacer las necesidades de este proyecto, y que pueda ahorrar el tiempo de cómputo, por ello es necesario conocer las herramientas que nos permitan hacer este tipo de desarrollo, es decir, que permitan interactuar con los datos almacenados en una base, además de incorporar una interacción con el usuario de una manera agradable.

Para realizar todo esto nos tendremos que valer de lenguajes de programación que estén orientados al *web*, y de servidores de bases de datos que puedan interactuar con dichos lenguajes, todo esto para poder tener un resultado óptimo.

Todo lo anterior es inútil si no tenemos como principio básico que todo lo que se requiere para poder mostrar información en la *web* no servirá si se deja de contemplar los lenguajes de marcas, para ser más específico, el *html* que se ocupara de manera que pueda ser validado por la W3C, es decir que usaremos un dialecto XML, que es llamado XHTML. Por todo lo mencionado el *html* será la base principal sobre lo cual se regirá todo lo que se contempla mostrar, por lo que es necesario conocerlo bien para poder desarrollar el sitio. Por ello se a puesto especial énfasis en que se consulte algún manual<sup>1</sup> por cualquier duda que pueda surgir de cualquier parte del código fuente, aquí solo mencionaré uno, pero puede ser encontrado algún otro tutorial en el Internet, y cualquiera que sea que consulten podrá aclarar lo que no se entiende.

Una vez explicado en que nos basaremos, se podrá pasar a evaluar las herramientas que harán dinámico al *html* y que son una posibilidad real de utilizarlas, si es que en las comparaciones que se hagan entre ellas resultan ser las más fiables dentro de varias posibilidades que se contemplan, o bien que resulten ser compatibles con lo que se esta requiriendo en el desarrollo del sitio.

### 2.2 Descripciones de herramientas

Es esta sección se hará la evaluación de las diferentes opciones estudiadas para desarrollar el sitio, por tal motivo solo se hará mención de aquellas que puedan funcionar para este proyecto. Es necesario tomar en cuenta que algunas de las herramientas aquí mencionadas funcionan en un sistema operativo diferente a *Linux* y que para su implementación se requerirá de un *software* que pueda servir de intérprete, sin embargo también se incluyeron porque podrían ser una opción.

#### 2.2.1 Análisis de lenguajes de programación orientados a Internet

Sabiendo que existen muchos lenguajes de programación en el mundo, capaces de interactuar con Internet y cualquiera de sus navegadores (Explorer, Firefox, Opera, etc.), se decidió optar por los más conocidos en el presente, que han ido evolucionando de tal forma que han alcanzado la meta de satisfacer al programador, haciendo un lenguaje

---

<sup>1</sup> <http://es.tldp.org/Manuales-LuCAS/doc-curso-html/doc-curso-html/>

fácil de utilizar y con la capacidad de poder llevar a cabo la programación que se necesita para cumplir con este proyecto.

Para ahondar más en el tema no queda otra opción que comenzar a describir los lenguajes de los que se hizo referencia, sin dejar de considerar que no son las únicas opciones en el mercado y que se pueden contemplar aun más, pero para fines didácticos solo nos quedamos con lo que abajo se desarrolla.

### 2.2.1.1 ASP

*Active Server Pages* (ASP), es una tecnología propietaria de Microsoft. Se trata básicamente de un lenguaje de tratamiento de textos (*scripts*), basado en Basic, y que se denomina VBScript (*Visual Basic Script*). Se utiliza casi exclusivamente en los servidores Web de Microsoft (*Internet Information Server* y *Personal Web Server*). Los *scripts* ASP se ejecutan, por lo tanto, en el servidor y puede utilizarse conjuntamente con *html* y *Javascript* para realizar tareas interactivas y en tiempo real con el cliente.

La filosofía de ASP resulta muy sencilla, en pocas palabras se puede definir de la siguiente forma: las páginas ASP, también llamadas páginas activas, son páginas que contienen código *html*, *script* de cliente y un *script* que se ejecuta en el servidor, dando como resultado código *html*. Por lo tanto al cargar una página ASP en nuestro navegador, en realidad no estamos cargando la página ASP como tal, sino el resultado de la ejecución de la página ASP, es decir la salida de la página ASP, se trata de código *html*. Es decir, son páginas que se ejecutan en el servidor enviando como resultado al cliente código *html*.

Antes de seguir vamos a definir de forma sencilla lo que se considera un lenguaje de *script* o de secuencia de comandos. Un lenguaje de *script* es un subconjunto de otro lenguaje más general y que se utiliza para un entorno muy determinado, en este caso el entorno es la *web*.

Una página ASP podrá contener los siguientes elementos: texto, componentes *ActiveX* (Aplicación desarrollada por Microsoft que ofrece dinamismo en páginas de Internet.), código HTML y comandos de *script*. Este *script* puede ser de dos tipos: *script* de cliente o *script* de servidor. El *script* de servidor es la nueva idea que introduce ASP, se debe tener en cuenta que en el *script* de servidor se tiene acceso a diferentes objetos y no está orientado a eventos.

El *script* de servidor utilizado en ASP maneja la misma sintaxis que el *script* de cliente, la diferencia está en que con ASP el *script* de servidor es compilado y procesado por el servidor Web antes de que la página sea enviada al navegador.

ASP no es un lenguaje de *script*, ASP ofrece un entorno para procesar *scripts* que se incorporan dentro de páginas *html*, es decir, un entorno de procesamiento de *scripts* de servidor.

La propia Microsoft define ASP de la siguiente manera: "...es un entorno de secuencias de comandos en el lado del servidor que puede utilizar para crear y ejecutar aplicaciones de servidor Web dinámicas, interactivas y de alto rendimiento...".

Realmente, ASP es un componente (asp.dll) que se instala en un servidor Web y cuya misión es la de procesar ficheros que terminan con la extensión .asp y transmitir el

resultado al cliente que solicitó la página ASP. El *script* de servidor incluido en una página ASP empieza a ejecutarse cuando un navegador solicita el archivo .asp al servidor Web. El servidor Web llama entonces a ASP, el cual lee el archivo solicitado de arriba a abajo, ejecuta los comandos y envía una página *html* al explorador. ASP incluye un motor de interpretación de *scripts* del lado del servidor. Las páginas ASP son ficheros con la extensión asp. Crear un fichero .asp resulta muy sencillo, se puede crear a partir de una página *html* existente, simplemente renombrando el fichero .html o .htm a un fichero .asp. Para hacer esta página ASP disponible para los usuarios de la *web*, el fichero .asp se debe almacenar en un directorio de publicación en Internet, se debe tener en cuenta que el directorio virtual asociado debe tener permisos de ejecución de secuencias de comandos.

Con ASP se pueden realizar fácilmente páginas de consulta de bases de datos, funciones sencillas como obtener la fecha y la hora actual del sistema servidor, cálculos matemáticos simples, etc.

El desarrollo que se ha venido dando a lo que es ASP ha sido bastante amplio. Entre sus funciones principales están el acceso a base de datos, envío de correo electrónico, creación dinámica de gráficos y otros. Básicamente, muchas cosas que podemos realizar por medio de CGI pueden ser realizadas con esta tecnología. Esto debido a que el ASP es tan eficiente con escribir código directamente a la interfase de aplicación del servidor, con la ventaja de que es más eficiente que el CGI que depende de un compilador ya que el ASP corre como un servicio en el servidor, tomando ventaja de la arquitectura de multitareas.<sup>2</sup>

ASP se ayuda de dos lenguajes de *script*, como son *JavaScript* y *VBScript* para implementar toda lo necesario para que se vea ASP como un lenguaje de programación.

El esquema de funcionamiento de ASP es una máquina cliente que realiza una petición de una página ASP. Esta petición llega a una máquina servidor la cual interpreta el código de esa página ASP. Dicho código puede tener accesos a ficheros o bases de datos. El resultado de interpretar la página ASP es una página *html*, la cual se le envía al usuario. Es decir, el usuario no llega a ver el código ASP, sino que ve el resultado de interpretar dicho código: una página *html*. La respuesta a la petición de una página ASP es una página *html*. Es por ello que dentro de una página ASP podemos encontrarnos con:

1. **Código ASP**, devolveremos aquel código que sea susceptible de cambiar, o el encargado de acceder a una base de datos.
2. **Código HTML**, partes del código HTML que permanezcan inmutables, esas partes de código se incluirán sin más, como si de una página HTML se tratara.

Ambos códigos se mezclarán dentro de la página ASP sin ningún orden.

---

<sup>2</sup> *www.LaLibreriaDigital.com*, "Programación de aplicaciones para Internet con ASP 3", España, Grupo Eidos, 2000, p. 11-43.

Se puede decir que una aplicación en ASP tiene como objetivo diseñar una página *web*. Todas las salidas de información que se realicen en una página ASP serán de código *html* o texto.<sup>3</sup>

### 2.2.1.2 JSP

Las páginas JSP o *Java Server Pages* es una tecnología desarrollada por *Sun Microsystems* como respuesta a la aparición de las páginas ASP (*Active Server Pages*) por parte de Microsoft. Una página JSP no es más que una página *web* normal y corriente que contiene porciones de código en Java y porciones de código en *html* junto con otros elementos que proporcionan información adicional al terminal en el que la página va a ser visualizada.

Cada página JSP es compilada automáticamente en un *servlet* por el motor JSP. (sólo se puede usar JSP en servidores que sean compatibles con JSP). La creación y compilación automática del *servlet* ocurre la primera vez que se accede a la página. Dependiendo del comportamiento del servidor Web, el *servlet* será grabado durante algún periodo de tiempo para utilizarlo una y otra vez sin necesidad de recrearlo y recompilarlo. Por eso, la primera vez que se accede a la página, podría haber una pausa mientras que el servidor Web crea y compila el *servlet*. Después de esto, los accesos a la página serán muchos más rápidos.<sup>4</sup>

El enlace entre una página JSP y su correspondiente *servlet* se realiza mediante dos clases contenidas dentro del paquete *javax.servlet.jsp*, estas dos clases son *JSPPage* y *HttpJspPage* que son las clases que nos permitirán crear la interfaz de la JSP compilada o dicho de otro modo del *servlet*. Una página JSP no es más que un fichero de texto (y cuya extensión es *.jsp*) en cuyo interior hacemos uso de determinados elementos para generar una respuesta en el terminal del cliente, estos elementos en su mayoría son vistos por el motor de JSP como objetos y como tales objetos tienen un alcance o duración en el transcurso de la cual van a ser accesibles. Estos alcances son de cuatro tipos diferentes:

1. **De página (*page*):** Los objetos que se declaran a este nivel son solo accesibles dentro de la página en la cual fueron creados. Las referencias de los objetos creados con alcance de página se almacenan en otro objeto denominado *pagecontext*.
2. **De petición (*request*):** En este caso los objetos declarados a este nivel son accesibles por todas las páginas que van a procesar la misma petición (*request*). Las referencias creadas sobre estos objetos se almacenan en el objeto *request*.
3. **De sesión (*session*):** Los objetos con alcance de sesión son accesibles por todas aquellas páginas que están procesando peticiones dentro de la misma sesión en la cual el objeto fue creado. Las referencias de los objetos creados se almacenan dentro del objeto *session*.
4. **De aplicación (*application*):** Es el nivel más alto en el sentido en que los objetos creados a este nivel son accesibles por todas las páginas que procesan peticiones dentro de la misma aplicación. Las referencias se almacenan en el objeto *application*.

<sup>3</sup> <http://www.aulambra.com/ver2.asp?id=20&tipo=>

<sup>4</sup> [http://www.programacion.net/java/tutorial/servlets\\_jsp/12/](http://www.programacion.net/java/tutorial/servlets_jsp/12/)

En resumen JSP hace posible combinar las mejores capacidades del *HTML* con los componentes de software reutilizables para crear aplicaciones del lado del servidor.<sup>5</sup>

### 2.2.1.3 Ajax

AJAX son las siglas de **A**synchronous **J**avaScript **A**nd **X**ML. No es un lenguaje de programación sino un conjunto de tecnologías (*html-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML*) que nos permiten hacer páginas de internet más interactivas.

La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar podemos enviar información al servidor.<sup>6</sup>

AJAX incorpora:

1. Presentación basada en estándares usando XHTML y CSS.
2. Exhibición e interacción dinámicas usando el *Document Object Model*.
3. Intercambio y manipulación de datos usando XML y XSLT.
4. Recuperación de datos asincrónica usando *XMLHttpRequest*.
5. *JavaScript*.

Una aplicación AJAX elimina la naturaleza “arrancar-frenar- arrancar-frenar” de la interacción en la *web* introduciendo un intermediario -un motor AJAX- entre el usuario y el servidor. En vez de cargar una página *web*, al inicio de la sesión, el navegador carga al motor AJAX (escrito en *JavaScript* y usualmente “sacado” en un *frame* oculto). Este motor es el responsable por renderizar la interfaz que el usuario ve y por comunicarse con el servidor en nombre del usuario.

El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo.

Cada acción de un usuario toma la forma de un llamado *JavaScript* al motor AJAX. Cualquier respuesta a una acción del usuario que no requiera un viaje de vuelta al servidor (como una simple validación de datos, edición de datos en memoria, incluso algo de navegación) es manejada por su cuenta.

Si el motor necesita algo del servidor para responder (sea enviando datos para procesar, cargar código adicional, o recuperando nuevos datos) hace esos pedidos asincrónicamente, usualmente usando XML, sin frenar la interacción del usuario con la aplicación.

AJAX, en resumen, se enfoca a cargar y renderizar una página, después mantenerse en esa página mientras los *scripts* y rutinas van al servidor buscando en “*background*”, los datos que son usados para actualizar la página solo re-renderizando a esta última y mostrando u ocultando porciones de la misma.<sup>7</sup>

<sup>5</sup> <http://www.wmlclub.com/articulos/jsp.htm>

<sup>6</sup> <http://www.ajaxya.com.ar/temarios/descripcion.php?cod=8&punto=1>

<sup>7</sup> <http://www.maestrosdelweb.com/editorial/ajax/>

#### 2.2.1.4 PHP

Es un lenguaje de programación usado generalmente para la creación de contenido para sitios *web*. El significado de las siglas PHP es "*PHP Hypertext Pre-processor*".

PHP es un lenguaje interpretado por el servidor que se caracteriza por su potencia, versatilidad, robustez y modularidad. Los programas escritos en PHP son embebidos directamente en *html* y ejecutados en el servidor Web a través de un intérprete antes de transferir al cliente que lo ha solicitado un resultado en forma de *html* puro. Al ser un lenguaje que sigue la corriente *OpenSource* (Código Abierto), tanto el intérprete como su código son totalmente accesibles de forma gratuita en la red Internet.

Por su flexibilidad, PHP resulta un lenguaje muy sencillo de aprender; especialmente para programadores familiarizados con lenguajes como *C*, *Perl* o *Java*, debido a las similitudes de sintaxis entre ellos. Por supuesto, es un lenguaje multiplataforma; los programas funcional igual sobre cualquier plataforma, trabajando sobre la mayoría de servicios *web* y estando preparado para interactuar con más de 20 tipos de bases de datos (*InterBase*, *mSQL*, *MySQL*, *Oracle*, *Informix*, *PostgreSQL*, entre otras.). No obstante, al ser un lenguaje concebido inicialmente para entornos *Unix*, es sobre este sistema sobre el cual se pueden aprovechar mejor sus prestaciones.<sup>8</sup>

Con PHP se puede hacer cualquier cosa que podemos realizar con un *script* CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de *cookies* y páginas dinámicas. Las aplicaciones dinámicas para el *web* son frecuentes en los sitios comerciales *e-commerce*, donde el contenido visualizado se genera de la información alcanzada en una base de datos u otra fuente externa.

PHP también ofrece la integración con las varias bibliotecas externas, que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en pdf hasta analizar código XML. También brinda una solución simple y universal para las paginaciones dinámicas del *web* de fácil programación. Su diseño elegante lo hace perceptiblemente más fácil de mantener y ponerse al día.

Debido a su amplia distribución, PHP esta perfectamente soportado por una gran comunidad de desarrolladores. Como producto de código abierto goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

PHP es la opción natural para los programadores en máquinas con Linux que ejecutan servidores Web con *Apache*, pero funciona igualmente bien en cualquier otra plataforma de *UNIX* o de Windows, con el software de *Netscape* o del *web server* de Microsoft.<sup>9</sup>

---

<sup>8</sup> Cobo, Ángel, Gómez, Patricia, *et. al*, "*PHP y MySQL. Tecnologías para el desarrollo web*", España, 2005, p. 504.

<sup>9</sup> <http://www.maestrosdelweb.com/editorial/phpintro/>

### 2.2.1.5 Comparación entre lenguajes

En esta sección se pondrá en una balanza lo que ofrecen los lenguajes de programación orientados a Internet que se describieron anteriormente. Cabe señalar que debe de cubrir con las necesidades que se tienen y los medios con los que se cuentan, es por eso que la principal característica que debe de cubrir aquel lenguaje que se seleccione, es la de correr bajo una plataforma Linux, ya que es el sistema operativo en el que se implementaran los servicios de *web* y en el que corre el código numérico a utilizar; por ello debe de ser compatible con este sistema.

Así es que considerando ese requisito y por las características que se mencionan arriba, podríamos descartar a ASP, pero como todo, existe un remedio, ya que este puede ser integrado por medio de un proyecto llamado *MONO*, por ello es que este lenguaje sigue en los concursantes.

Otro perfil que debe de cubrir es que tiene que ser capaz de poder interactuar con gestores de bases de datos, ya que es necesario que se puedan realizar consultas con los datos correspondientes a la simulación numérica, es por ello que hay que voltear a ver de nuevo lo que ofrece cada uno de estos lenguajes.

ASP proporciona acceso a datos apoyándose en los objetos ADO (*ActiveX Data Objects*) y ODBC. El uso de la interfaz ODBC le permite a ASP trabajar sobre cualquier sistema gestor de bases de datos que proporcione un controlador o *driver* (*MySQL, SQL Server, Oracle, Informix, etc.*).

JSP se apoya en la tecnología JDBC de Java, para ello se precisa un controlador o *driver* que proporcione el acceso a la base de datos subyacente (*MySQL*).

AJAX no tiene acceso a bases de datos ya que es un conjunto de varias tecnologías, así que depende de cualquiera de los otros 3.

PHP ofrece interfaces propias de acceso a multitud de fuentes de datos: *MySQL, mSQL, PostgreSQL, Oracle 8, etc.*

Por lo anterior podemos eliminar al lenguaje Ajax, ya que el acceso a bases de datos depende de cualquier otro lenguaje. Con respecto a los demás hay que señalar que ASP y PHP llevan ventaja sobre JSP, ya que tienen más servidores de bases de datos por escoger.

ASP necesita un el proyecto *MONO* (nombre de un proyecto de código abierto impulsado por Novell para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con *.NET* según lo especificado por el *ECMA*).<sup>10</sup>

JSP necesita el software que implementa las especificaciones *Java Servlet 2.1* o *2.2* y *Java Server Pages 1.0* ó *1.1*. Se puede obtener una versión gratuita de *Sun*, conocida como "*JavaServer Web Development Kit*" (*JSWDK*), o bien se puede obtener el paquete *Tomcat* (contenedor de *servlets* realizado por la fundación Apache).

---

<sup>10</sup> <http://www.gxopen.com/commwiki/servlet/hwiki?Proyecto+Mono>

PHP requiere de la instalación de un servidor, que puede ser cualquiera existente en la red.

Como alcanzamos a apreciar todos estos lenguajes requieren de un servidor, y debido a que ASP necesita de un proyecto que apenas esta en crecimiento (*Mono*), podemos descartarlo ya que la información existente sobre *mono* no es lo suficientemente amplia, con respecto a nuestros 2 candidatos restantes podemos pasar a un criterio más de evaluación.

El costo por la obtención de estos paquetes podría ser un siguiente filtro, pero debido a que en la versión de *Linux Fedora Core 5* se incluyen estos paquetes (*Tomcat* y *PHP*), listos para su configuración, no sería un buen criterio.

Debido a que los 2 pueden dar el mismo resultado, ocupare un criterio muy sencillo y es analizar cual de los 2 es más fácil de utilizar y contiene más soporte. Dado que JSP requiere de más líneas de código que PHP para capturar datos enviados a través de un formulario, se toma la decisión por lógica.

Así es que por agregar a la ventaja de que PHP es compatible con más servidores de bases de datos que JSP, tenemos que es más fácil de usar, además de que la mayoría de los sitios que se encuentran en la actualidad son hechos con PHP, y por ende hay más información de cómo se realizan las cosas con este lenguaje, sumándole que es completamente *OpenSource*, es que elegimos a este lenguaje para desarrollar la **Remotización del Portal Universitario de Simulaciones Numéricas**.

Con esta decisión no se quiere indicar que PHP es mejor que JSP, si no que sencillamente es más práctico.

### 2.2.2 Análisis de bases de datos compatibles con los anteriores lenguajes

Es de suma importancia que tomemos como base el lenguaje donde se va a programar para poder elegir una base de datos, ya que hay que verificar la compatibilidad que tiene con los servidores de bases de datos.

Así es que tomando como inicio que será programado el proyecto de **Remotización del Portal Universitario de Simulaciones Numéricas** con PHP, hay que verificar con que servidores es compatible, y hemos encontrado que son muchos, de los cuales podemos mencionar:

*InterBase, mSQL, MySQL, Oracle, Informix, PostgreSQL, ODBC, DB2, Microsoft SQL Server, Firebird, SQLite, etc.*

Así es que habrá que consultar que nos puede ofrecer cada una de estas, pero antes hay que reducir las posibilidades, y simplemente lo haremos de la manera en que son más comerciales, por lo que solo se elegirá a 3 candidatos y los compararemos, de ahí saldrá el servidor que utilizaremos.

Eligiendo a los 3 candidatos, no puede faltar el hecho por Microsoft, ya que existe mucha información relacionada con él, por lo que *Microsoft SQL Server* será nuestra primera elección, quedando así 2 plazas para continuar.



Consultando las funciones de bases de datos que tiene PHP es ineludible anexar en una siguiente plaza a *MySQL* ya que estos van de la mano. Pero otro servidor de bases de datos que tiene contemplado PHP en muchas de sus funciones es *PostgreSQL*, por lo cual lo anexaremos.

Ya que tenemos a nuestros 3 candidatos, veremos sus características más a fondo, para que al último las comparemos y se pueda obtener lo que utilizaremos junto con PHP.

### 2.2.2.1 PostgreSQL

*PostgreSQL* es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (*ORDBMS*) que ha sido desarrollado de varias formas desde 1977. Comenzó como un proyecto denominado *Ingres* en la Universidad Berkeley de California. *Ingres* fue más tarde desarrollado comercialmente por la *Relational Technologies/Ingres Corporation*.

En 1986 otro equipo dirigido por Michael Stonebraker de Berkeley continuó el desarrollo del código de *Ingres* para crear un sistema de bases de datos objeto-relacionales llamado *Postgres*. En 1996, debido a un nuevo esfuerzo de código abierto y a la incrementada funcionalidad del software, *Postgres* fue renombrado a *PostgreSQL*, tras un breve paso como *Postgres95*. El proyecto *PostgreSQL* sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. *PostgreSQL* está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre.

Algunas de sus principales características son:

1. **DBMS Objeto-Relacional:** PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transiciones, optimización de consultas, herencia, y arreglos.
2. **Altamente Extensible:** PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.
3. **Soporte SQL Comprensivo:** PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (*joins*) SQL92.
4. **Integridad Referencial:** PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
5. **API (Application Programming Interface) Flexible:** La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen *Object Pascal*, *Python*, *Perl*, *PHP*, *ODBC*, *Java/JDBC*, *Ruby*, *TCL*, y *C/C++*,
6. **Lenguajes Procedurales:** *PostgreSQL* tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado *PL/pgSQL*. Este lenguaje es comparable al lenguaje procedural de *Oracle*, *PL/SQL*. Otra ventaja de *PostgreSQL* es su habilidad para usar *Perl*, *Python*, o *TCL* como lenguaje procedural embebido.
7. **MVCC:** MVCC, o Control de Concurrencia Multi-Versión (*Multi-Version Concurrency Control*), es la tecnología que *PostgreSQL* usa para evitar bloqueos innecesarios. Mediante el uso de MVCC, *PostgreSQL* evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque

un lector nunca es bloqueado por un escritor. En su lugar, *PostgreSQL* mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. *PostgreSQL* es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

8. **Cliente/Servidor:** *PostgreSQL* usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a *PostgreSQL*.
9. **Write Ahead Logging (WAL):** La característica de *PostgreSQL* conocida como *Write Ahead Logging* incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura *GiST* de *PostgreSQL*. Algunos ejemplos son los tipos de datos *GIS* (Sistema de Información Geográfica) creados por el proyecto PostGIS.

#### Otras características

- a) Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (*foreign keys*).
- b) Disparadores (*triggers*).
- c) Vistas.
- d) Integridad transaccional.
- e) Herencia de tablas.
- f) Tipos de datos y operaciones geométricas.

#### Funciones

Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientación a objetos o la programación funcional. Los disparadores (*triggers* en inglés) son funciones enlazadas a operaciones sobre los datos. *PostgreSQL* soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (*stored procedures*).<sup>11</sup>

<sup>11</sup> <http://www.sobl.org/traduccion/practical-postgres/node12.html>

### 2.2.2.2 SQL Server

*Microsoft SQL Server* es un sistema de gestión de bases de datos relacionales (SGBD) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea que permite tener las ventajas que a continuación se pueden describir.

**1. Disponibilidad:** Las inversiones en tecnologías de alta disponibilidad, las soluciones de copia de seguridad y recuperación adicionales y las mejoras en los sistemas de réplica permiten desarrollar e implantar aplicaciones de gran fiabilidad. Algunas características innovadoras que aumentan la disponibilidad, como las operaciones de reflejo (*mirroring*) de bases de datos, la configuración en clúster de conmutación tras error (*failover*) y las operaciones en línea minimizarán los tiempos de inactividad y ayudarán a garantizar que los sistemas más importantes se mantengan accesibles.

**2. Escalabilidad:** Los adelantos a este respecto, como la visión de partición de tablas, el aislamiento de instantáneas y el soporte de 64-bit le permiten desarrollar e implementar las aplicaciones más demandadas mediante *SQL Server*. La partición de tablas e índices mejoran sensiblemente el resultado de las consultas, al contrario de lo que ocurre con bases de datos muy extensas.

**3. Seguridad:** Con ajustes de la configuración predeterminada de seguridad y un modelo de seguridad mejorado, se facilita el logro de más altos niveles de seguridad de la información.

**4. Facilidad de gestión:** Un conjunto de herramientas, funciones de auto-sintonización ampliadas y un potente modelo de programación facilita a los administradores de datos llevar un control flexible de las operaciones diarias de bases de datos. Ayuda también a los administradores de datos a poner a punto sus servidores en situación de óptimo rendimiento.

**5. Interoperabilidad:** Los adelantos tecnológicos que se incorporan en *SQL Server* permiten optimizar su inversión tanto en nuevos sistemas como en los existentes a través de la integración y conexión de aplicaciones y bases de datos descentralizadas. Mediante un fuerte apoyo a los estándares del sector, los servicios *web* y la plataforma *Microsoft .NET Framework*, *SQL Server* admite la interoperabilidad con múltiples plataformas, aplicaciones y dispositivos.

Todo lo anterior esta enfocado a poder dar los siguientes resultados:

1. Desarrollar e implantar aplicaciones empresariales más escalables, fiables y seguras.
2. Optimizar la productividad reduciendo la complejidad en la creación, implantación y administración de las aplicaciones de bases de datos.
3. Aumentar las capacidades de los desarrolladores con un entorno de desarrollo valioso, flexible y actual para que creen bases de datos más seguras.
4. Compartir datos a través de múltiples plataformas, aplicaciones y dispositivos para facilitar la interconexión entre sistemas internos y externos.
5. Ofrecer soluciones de inteligencia empresarial que ayuden a tomar decisiones con fundamento y aumentar la productividad por toda la empresa.

6. Controlar los costes sin sacrificar el rendimiento, la disponibilidad ni la fiabilidad.

De esta forma se completa una potente base de datos (Microsoft SQL Server) con un entorno de desarrollo cómodo y de alto rendimiento (*VBA (Visual Basic for application Access)*) a través de la implementación de aplicaciones de dos capas mediante el uso de formularios.<sup>12</sup>

### 2.2.2.3 MySQL

*MySQL* es un sistema gestor de bases de datos relacionales en *SQL*, esto significa que permite la gestión de los datos de una BBDD relacional usando un lenguaje de consulta estructurado. Y, por tanto, que a partir de una oración, *MySQL* llevará a cabo una determinada acción sobre nuestra base de datos.

*MySQL* es una aplicación de código abierto y por lo tanto es gratuita, nos permite redistribuir una aplicación que la contenga y nos permite incluso modificar su código para mejorarla o adaptarla a nuestras necesidades. Además, existe la seguridad de contar con una importante cuota de mercado y de saber que es una solución estable, mantenida por un buen equipo de desarrolladores y e incluso con soporte de pago.<sup>13</sup>

*MySQL* es muy rápido, seguro y fácil de usar, también ha desarrollado un conjunto de características muy prácticas, en estrecha cooperación con otros usuarios. Este gestor fue desarrollado para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha sido usado exitosamente en ambientes de producción con altas demandas, por varios años. Aunque está bajo un desarrollo constante, siempre ofrece conjunto de funciones muy poderosas y eficientes. La conectividad, velocidad y seguridad hace de *MySQL* una *suite* poderosa para acceder a bases de datos en Internet.

Es un sistema Cliente/Servidor que consta de un servidor SQL multi-hilo que soporta diferentes *backends* (Modo administrativo protegido por usuario y contraseña), variados programas cliente y de librerías, administrador de herramientas y un programa de interfase.

Es probable que se encuentre que diversas aplicaciones soportan a *MySQL*. Sus valores centrales son:

- La mejor y más usada base de datos en el mundo.
- Disponible y Accesible para todos.
- Fácil de usar.
- Se está perfeccionando continuamente mientras permanece rápida y segura.
- Divertida para usar y perfeccionar.
- Libre de molestias.

Entre sus nuevas características tenemos:

- Escrito en *C* y *C++*. Usa GNU *autoconf* para portabilidad.
- Clientes *C*, *C++*, *Eiffel*, *PHP*, *Python*, *JAVA*, *Perl*, *TCL*.

<sup>12</sup> <http://www.microsoft.com/latam/sql/2005/productinfo/>

<sup>13</sup> <http://www.programacionweb.net/articulos/articulo/?num=184>

- Multiproceso, es decir puede usar varias CPU si éstas están disponibles.
- Puede trabajar en distintas plataformas y S.O. distintos.
- Sistema de contraseñas y privilegios, muy flexible y seguro.
- Todas las palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- 16 índices por tabla, cada índice puede estar compuesto de 1 a 15 columnas o partes de ellas con una longitud máxima de 127 bytes.
- Todas las columnas pueden tener valores por defecto.
- Utilidad *Isamchk* para chequear, optimizar y reparar tablas.
- Todos los datos están grabados en formato ISO8859\_1.
- Los clientes usan *TCP* o *UNIX Socket* para conectarse al servidor.
- Todos los comandos tienen *-help* o *-?* Para las ayudas.
- Soporta diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, etc.
- ODBC, se puede utilizar *ACCESS* para conectar con el servidor.
- Muy rápida usando *joins* (operaciones en la base de datos relacionando 2 o más columnas), optimizada para un barrido multi-*joins*.
- Todas las funciones y operadores soportan en el *SELECT* y *WHERE* como partes de consultas.
- Todas las cláusulas SQL soportan *GROUP BY* y *ORDER BY*.

*MySQL* es muy utilizado en aplicaciones web, en plataformas (*Linux/Windows-Apache-MySQL-PHP/Perl/Python*), y por herramientas de seguimiento de errores como *Bugzilla*. Su popularidad como aplicación *web* está muy ligada a *PHP*, que a menudo aparece en combinación con *MySQL*. Es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional *MyISAM* (tecnología de almacenamiento de datos usada por defecto por el sistema administrador de bases de datos relacionales *MySQL*), pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones *web* hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a *MySQL* ideal para este tipo de aplicaciones.

*MySQL* funciona sobre múltiples plataformas. Los usuarios o miembros de la comunidad *MySQL* pueden reportar *bugs* (errores) revisando el manual en línea que contiene las soluciones a problemas encontrados; el historial de cambios; la base de datos *bugs* que contiene *bugs* reportados y solucionados y en las listas de correo *MySQL*.<sup>14</sup>

#### 2.2.2.4 Comparación entre bases de datos

En esta parte del presente trabajo, nos enfocaremos a tomar la decisión de escoger un servidor de base de datos entre las opciones que mencionamos anteriormente. Es muy importante tomar en cuenta que estas opciones son compatibles con lenguaje seleccionado, pero hay que fijarnos la idea de ver si son compatibles con la primera regla de filtrado que ocupamos en la selección del lenguaje de programación.

---

<sup>14</sup> <http://www.pecesama.net/php/bd.php?PHPSESSID=0c39861ebb72800da48edf636080933d>

Por tanto, sabemos que los servidores de *PostGreeSQL* y *MySQL* son compatibles con el sistema operativo *Linux*, pero se tiene que revisar si *SQL Server* lo es, ya que como sabemos Microsoft en la mayoría de sus productos, es incompatible con el software GNU (en este caso *Linux*), por lo que nos dice su página principal la utilería Microsoft *.NET Framework*, permite la interoperabilidad con múltiples plataformas, pero debido a que necesita del mismo software adicional que *ASP* para que funcione en *Linux* llamado *MONO*<sup>15</sup>, agregándole que necesitamos utilizar la plataforma *.Net*, damos por concluida la comparación entre 3 candidatos, para abrir paso solamente a 2, excluyendo a *SQL Server*.

Para continuar las comparaciones es necesario que veamos lo mejor y lo peor de cada uno.

Lo mejor de *PostGreeSQL*:

Las características positivas que posee este gestor según las opiniones más comunes en Internet<sup>16</sup>, son:

1. Posee una gran escalabilidad. Es capaz de ajustarse al número de *CPUs* y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos *benchmarks* [Técnica utilizada para medir el rendimiento de un sistema] se dice que ha llegado a soportar el triple de carga de lo que soporta *MySQL*).
2. Implementa el uso de *rollback's* (operación que vuelve a su estado anterior a la base de datos), subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que *MySQL* no podría.
3. Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser *Oracle*.

Lo peor *PostGreeSQL*:

Los mayores inconvenientes que se pueden encontrar a este gestor son:

1. Consume gran cantidad de recursos.
2. Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
3. Es de 2 a 3 veces más lento que *MySQL*.

Lo mejor de *MySQL*

Es evidente que la gran mayoría de gente usa este gestor en Internet, por lo que encontrar opiniones favorables no ha resultado en absoluto complicado:

1. Sin lugar a duda, lo mejor de *MySQL* es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
2. Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.

<sup>15</sup> <http://www.gxopen.com/commwiki/servlet/hwiki?Proyecto+Mono>

<sup>16</sup> [http://www.netpecos.org/docs/mysql\\_postgres/x108.html#AEN115](http://www.netpecos.org/docs/mysql_postgres/x108.html#AEN115)

3. Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.
4. Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.
5. El conjunto de aplicaciones *Apache-PHP-MySQL* es uno de los más utilizados en Internet en servicios de foro (*Barrapunto.com*) y de buscadores de aplicaciones (*Freshmeat.net*).

### Lo peor de *MySQL*

Debido a esta mayor aceptación en Internet, gran parte de los inconvenientes se exponen a continuación, han sido extraídos de comparativas con otras bases de datos:

1. Carece de soporte para transacciones, *rollback's* y subconsultas.
2. El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.
3. No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

Teniendo en cuenta lo anterior evaluaremos las características de uno y otro, para por fin tomar una decisión.

Podemos mencionar que una de las ventajas de *MySQL* es la rapidez, que es algo que necesitamos a la hora de poder realizar una consulta por la red, pero en contra tenemos que no tiene gran escalabilidad como lo hace *PostGreeSQL*, pero la pregunta aquí es si ¿necesitamos una base de datos de grandes dimensiones? a lo que podríamos contestar que “No”, debido a que no se necesitara una gran relación entre tablas (ver capítulo 3.1), por tal motivo es que esa ventaja que ofrece *PostGreeSQL* no se ajusta mucho a nuestro problema.

Ambos son capaces de adaptarse a los recursos que ofrecen las computadoras, es por ello que puede ser ocupado por cualquier PC o Servidor. Pero si hay que mencionar que por lo mismo que *PostGreeSQL* intenta ser más sofisticado que *MySQL* desperdicia más memoria y sobrecarga al sistema (pueden existir opiniones distintas).

*MySql* incluye dentro de sus ventajas que tiene una probabilidad muy pequeña de corromper datos. Esto nos ayuda, ya que es de suma importancia que ninguno de los datos se corrompa.

*PostGreeSQL* nos ofrece grandes cosas como lo son: *rollback's*, subconsultas y transacciones, pero de nuevo la pregunta ¿Es necesario todo esto?, y la respuesta que tenemos es que no, ya que solamente se harán consultas de datos e inserciones de los mismos, es por tanto que estas características son sobradas.

Para tomar una decisión solo hace falta ver las observaciones anteriores, que en resumen nos dice que *PostGreeSQL* nos da grandes cosas, pero a cambio de ellas sacrificamos

velocidad, que es algo fundamental para este proyecto, y si a esto agregamos que en el lenguaje PHP existen más funciones para comunicarse con *MySQL* que con *PostgreSQL*, tenemos que hay un vencedor para lo que necesitamos y este es *MySQL*, ya que como bien menciona una de sus ventajas es que *Apache-PHP-MySQL*, es algo muy utilizado para el desarrollo web, por lo tanto no es difícil predecir que existe mucha información para lograr un funcionamiento óptimo de este proyecto .

## 2.4 Código numérico utilizado en las simulaciones

Sabemos que las simulaciones numéricas son una poderosa herramienta para el desarrollo y entendimiento de diferentes ramas de la ciencia, tales como la física, la biología, las ingenierías, la medicina, la química, etc. El impresionante crecimiento de la infraestructura computacional, manifestado en supercomputadoras con arquitecturas vectoriales y paralelas, junto con algoritmos numéricos cada vez más sofisticados, nos permiten encontrar soluciones aproximadas a sistemas de ecuaciones diferenciales que describen la evolución de los fenómenos que deseamos estudiar y que no pueden resolverse por métodos analíticos. Estas soluciones aproximadas, a su vez, nos permiten hacer modelos evolutivos bajo diferentes condiciones a la frontera, que suelen denominarse experimentos numéricos. Cuando es imposible hacer experimentación directa, como es el caso de la astronomía, o cuando es peligroso o muy costoso hacer experimentos, como es el caso de la seguridad en reactores o en aviones, los experimentos numéricos son una herramienta valiosísima para estudiar el comportamiento de los sistemas. En el caso particular de la Astronomía, además de que no se pueden medir directamente las propiedades de los objetos celestes, los fenómenos cósmicos que habitan nuestro Universo necesitan de miles o millones de años para desarrollarse. Entonces, será imposible llevar un seguimiento puntual de su evolución, ya que el tiempo de vida del ser humano oscila en promedio entre los 80 y 100 años. Por lo tanto, la simulación numérica será una herramienta fundamental para estudiar y entender los diferentes campos de la astrofísica (cosmología, formación y evolución de estrellas y galaxias, discos protoestelares, formación de planetas, etc.).

### 2.4.1 Código Hidrodinámico Zeus 3D

Las simulaciones numéricas que se presentan en este trabajo se realizaron con el **código hidrodinámico ZEUS-3D** desarrollado en el *National Center Supercomputing Applications at the University of Illinois at Urbana-Champaign*, por Michael Norman, James Stone, y David Clarke. Aún cuando este código fue desarrollado para aplicaciones astrofísicas, puede utilizarse para resolver otros problemas asociados a ciencias físicas y de ingeniería. Astrónomos nacionales e internacionales utilizan ZEUS para describir la magnetohidrodinámica (MHD) de fluidos ideales que gobiernan la evolución de una gran cantidad de sistemas astrofísicos a diferentes escalas, desde estrellas y objetos compactos hasta el medio interestelar o aún más, el medio intergaláctico. El código ZEUS puede realizar cálculos numéricos en 1, 2 y 3 dimensiones, utilizando diferentes geometrías, tales como; cartesiana, cilíndrica y esférica. También puede manejar efectos autogravitatorios o producidos por campos gravitacionales externos. La evolución del fluido, la puede realizar a temperatura constante (proceso isotérmico) o adiabáticamente (flujo de calor constante). El algoritmo o método numérico, que utiliza el código para resolver el sistema de ecuaciones MHD que describen la dinámica del fluido es el de diferencias finitas, explícito en el tiempo. La malla computacional que utiliza ZEUS es euleriana ortogonal,



es decir, las celdas de la malla están fijas en el espacio. El paso de tiempo es controlado por la condición de Courant (si el lector esta interesado en información más detallada del código, recomendamos ampliamente consultar los artículos de Stone & Norman 1992).<sup>17</sup>

---

<sup>17</sup> Stone, J. M., & Norman, M. L. 1992, “*ZEUS-2D: A radiation magnetohydrodynamics code for astrophysical flows in two space dimensions. I - The hydrodynamic algorithms and tests*”. *Astrophysical Journal Supplement Series*, 80, 753.

## CAPÍTULO III IMPLEMENTACIÓN DEL PORTAL DE SIMULACIONES NUMÉRICAS REMOTAS.

### 3.1. Generación de la base de datos.

Para la generación de la base de datos asociadas a este portal como se vio en el capítulo anterior, se utilizó el manejador *MySQL*. Los campos que contienen la base de datos así como su descripción se pueden ver en la siguiente tabla (Tabla 2.1).

Nombre del campo	Variable	Descripción
n0	\$n0	Densidad Numérica (cm <sup>-3</sup> )
t0	\$t0	Temperatura (K)
b0	\$b0	Intensidad del campo magnético (μG)
n	\$n	Densidad (cm <sup>-3</sup> )
r	\$r	Radio (kpc)
v	\$v	Velocidad (km/s)
tar	\$matriz=\$nombre Zhto.\$nombre.tgz	Nombre del archivo comprimido con los archivos generados por la simulación numérica
movie	\$matriz=\$nombre movie.\$nombre.mpg	Nombre de la película, resultado de la simulación numérica.

Tabla 3.1

Después de aplicar la normalización a los datos requeridos para el almacenamiento de los datos de las simulaciones numéricas y llevando a cabo el modelado relacional, las tablas quedaron como siguen (Tablas 3.2 y 3.3).

#### Tabla: Simulación

Id	Entero, no nulo e incremental, llave primaria	Identificador de la simulación
<b>n0</b>	Flotante, no nulo, float (10)	Densidad Numérica
<b>t0</b>	Flotante, no nulo, float (10)	Temperatura
<b>b0</b>	Flotante, no nulo, float (10)	Intensidad del campo magnético
<b>n</b>	Flotante, no nulo, float (10)	Densidad
<b>v</b>	Flotante, no nulo, float (10)	Radio
<b>r</b>	Flotante, no nulo, float (10)	Velocidad

Tabla 3.2

**Tabla: Archivo**

<b>Id</b>	Entero, no nulo e incremental, llave primaria	Identificador de la simulación
<b>tar</b>	Caracter, varchar(30)	Archivo comprimido, con la información de la simulación numérica
<b>movie</b>	Caracter, varchar(30)	Archivo comprimido mpeg, con la información de la simulación numérica

Tabla 3.3

La información contenida en la base se obtendrá a través de la interfaz gráfica asociada a las simulaciones numéricas, más adelante en las secciones 3.2.3.3, 3.2.3.4 y 3.2.3.5., de describe a detalle los tipos de búsquedas que se pueden hacer en el portal, así como su implementación.

### 3.2 Programación del portal

En este desarrollo se ocupan diagramas de flujo para revisar como podrían darse los casos en las búsquedas en el sistema, así como también las necesidades que se tienen dentro del portal y el como se pueden presentar.

Se explica la forma en que se programa cada módulo que se ocupa de una forma detalla y explicita apoyándose en los apéndices que se mencionan ya que ahí se tiene la totalidad de la programación.

Se da a entender el flujo, el dinamismo del portal incluyendo la parte esencial y base de todo lo enfocado al *web*, que es el *html* para dar pie a la validación que se puede tener que es la *Xhtml* por medio de la *W3C*.

Dada esta pequeña introducción abrimos paso a la parte más significativa y clave de este trabajo que es la “**programación del portal**”.

#### 3.2.1 Página principal

Esta parte del programa presenta, a los 2 fragmentos en el que se divide, es decir, al catálogo de simulaciones numéricas y al portal de la realización de estas. Esto es desarrollado con 3 imágenes y con código de html puro, aunque la extensión sea guardada como php. El diseño, esta basado en varios portales dedicados a los eventos astronómicos, y tomando como punto de partido estos (ejemplo <http://vso.nso.edu/>), y agregándole el punto de vista de personas en el Instituto de Astronomía, se opta por una vista sencilla de la página, esto quiere decir que con pocas imágenes y con colores limpios como lo son el blanco, azul y naranja. Comenzando a describir el código *html*, que consta de una pequeña parte de php, y esta hace referencia a una función que lleva por nombre *obvio.php*, que lleva en su contenido 2 funciones, *inicioPag* y *finalPag*. La primera necesita que las variables titulo y estilo sean proporcionadas cuando se llame a ejecución, adicionando que la función lleva elementos básicos de *html* para poder comenzar a insertar lo que se necesita mostrar al cibernauta, tales como son la dirección para hacer valido el código como *Xhtml* ante la organización *W3C*, que se encarga de conducir al *web* a su máxima potencia por medio de estándares fijados desde hace más

de 10 años, estos estándares se consiguen mediante la indicación en el principio de todo *script* de las siguientes líneas:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Y en la segunda función de *obvio.php* (finalPag) solo se cierran los elementos del *html* con `</body>` y `</html>`.

Otro elemento que es incluido en *obvio.php*, es lo primordial que es `<head>`, este lo lleva y se cierra después de haber declarado el `<body>`. Sumándole la expresión `<title>`, que es donde se imprimirá la variable título y se cierra con `</title>`.

Agregamos una forma de que se tomen en cuenta una hoja de estilo, esto por medio de un `@import url` que llevara impreso la variable estilo, y cuando se requiera solo se incorporara la dirección donde se localiza el archivo que necesitamos incorporar para personalizar el portal. Finalmente se concluye la primera función con la apertura de `<body>`.

La segunda función tiene lo que cierra a `<body>` con `</body>`, y al *html* con `</html>`, como su nombre lo indica, simplemente es el final de toda página *Web* (para mayor referencia consultar apéndice B). Se comienza con la siguiente línea: `include("../libs/obvio.php")`; que es lo que informa que se requiere el contenido de dicho archivo ya descrito. En primera instancia se llama a *inicioPag*, a esta función se le da el título de la página, y la url de la hoja de estilo que se necesita como se indica: `inicioPag("Portal de simulaciones numéricas IA", "../libs/estilo.css")`; (*estilo.css*, ver apéndice A). Resultando que el siguiente paso es llenar el espacio después de `body` y comenzamos a hacerlo con dos tablas anidadas, la principal contiene un `borde=0`, un tamaño del 100%, y no hay espacios entre celdas ni del texto a la celda. Dicha tabla esta compuesta por 8 renglones y 3 columnas, que se encuentran distribuidos de la siguiente manera (Tabla 3.4):

# De Renglón	Columna 1	Columna 2	Columna 3
1	Vació con tamaño de 150 píxeles.	Vació con tamaño de 150 píxeles.	Vació con tamaño de 150 píxeles.
2	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.
3	Vació.	Imagen 3d.jpg con altura de 70 píxeles.	Vació.
4	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.
5	Vació con ancho del 10%.	Tabla anidada con ancho del 80%.	Vació con ancho del 10%.
6	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.
7	Vació con tamaño de 5 píxeles.	Letrero centrado.	Vació
8	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.	Vació con tamaño de 5 píxeles y relleno de color naranja.

Tabla 3.4

Es el contenido de la primera tabla, con respecto a la segunda, incluye también un borde=0, ningún espacio entre celdas y de celda al texto, e incluye un tamaño de 100% del 80% asignado en la tabla anterior. Consta de 2 columnas y un renglón, cuyo contenido es el siguiente (Tabla 3.5):

# De Renglón	Columna 1	Columna 2
1	Un tamaño del 50%, un letrero, una imagen con enlace hacia simulación por medio de un mapa de imagen.	Un tamaño del 50%, un letrero, una imagen con enlace hacia el catálogo por medio de un mapa de imagen.

Tabla 3.5

Y para acabar esta programación llamamos a la función *finalPag*, que dará el toque final y exacto a la presentación.

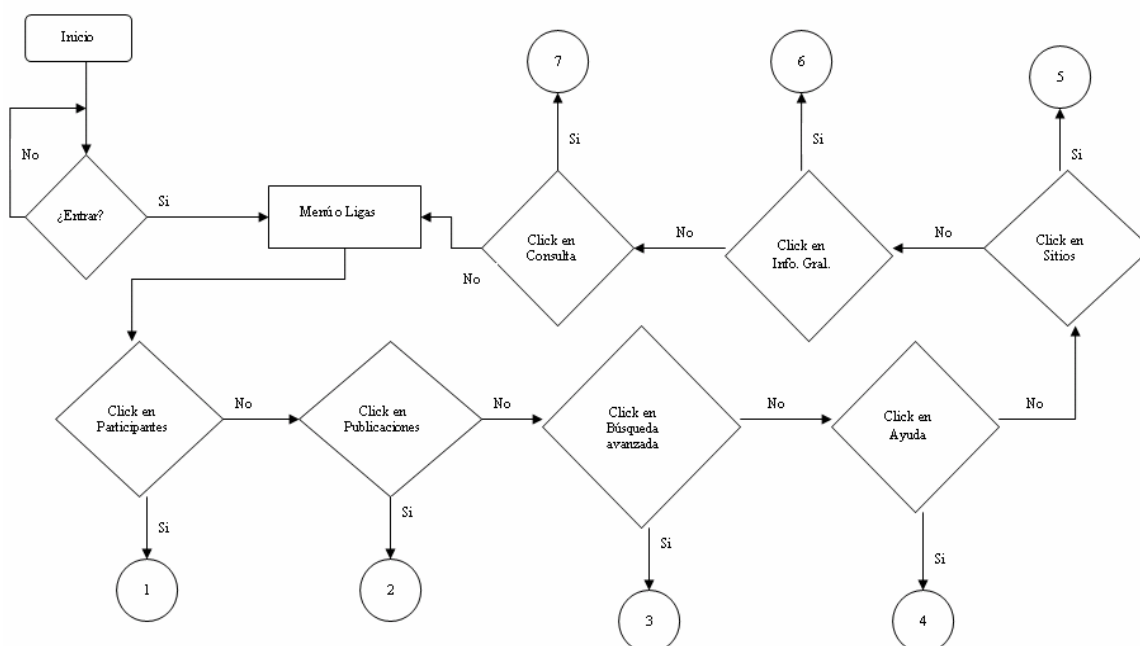
El código fuente de lo antes descrito se puede ver en el apéndice A, para cualquier aclaración o ampliación del panorama.

### 3.2.2 Desarrollo del catálogo de simulaciones numéricas

Es importante tomar en cuenta las herramientas que nos ofrece la programación, y por ello comenzaremos por hacer la planeación de los módulos que consta el catálogo de simulaciones numéricas, y esto lo haré con un simple diagrama de flujo, que será muy útil para tener una mejor descripción de los módulos.

Dicho diagrama se divide en 6 figuras las cuales son (Fig. 3.1):

Fig. 3.1

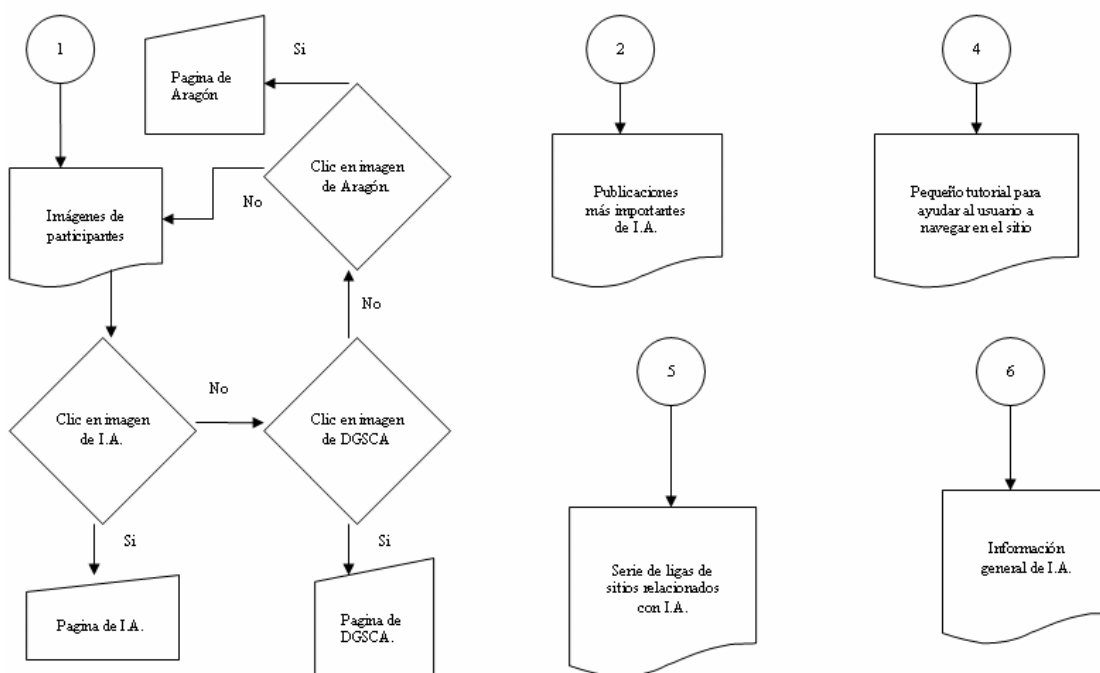


En esta figura se desglosan las secciones que se incluirán en este programa, las cuales son la pantalla de presentación, que en la figura esta desglosado en el inicio, y dicho inicio constara de 2 ligas que son dedicadas a las 2 partes del portal, 1) Catálogo y 2) Realización de simulaciones numéricas.

Continuando con la explicación de la figura, esta entrara en una toma de decisión, que es identificada por un click, si esta decisión no es verdadera, continuará en la misma pantalla, o esperara una señal verdadera para alguna de las dos ligas que se tienen.

Suponiendo que esta señal sea verdadera entrara a la página donde se encuentran localizado el menú de las ligas, el cual constara de 7, cada una con una función explícita. El programa entrara una vez más en alguna toma de decisión del usuario, que se encargara de definir a cual de estas quiere entrar, teniendo como opciones Consulta, Información General, Sitios de Interés, Participantes, Publicaciones, Búsqueda Avanzada. o Ayuda. Una vez tomada la decisión se pasara al número que esta indicado en el caso de un si, llevando acabo la función correspondiente.

Fig. 3.2



En la figura 3.2 se contiene la solución de los casos 1 (Participantes), 2 (Publicaciones), 4 (Ayuda), 5 (Sitios de interés) y 6 (Información general).

En el caso 1, se realizara algo muy sencillo que constara del despliegue de 3 imágenes que son las de los participantes (FES Aragón, DGSCA y el Instituto de Astronomía), aquí se decidirá cual es la elección que se tomara, en dado que sea verdadera la opción de FES Aragón, llevara al cibernauta a navegar en la página *web* de esta institución, o si la elección es DGSCA lo llevara a la sitio de esta y hará lo mismo con respecto al Instituto de Astronomía.

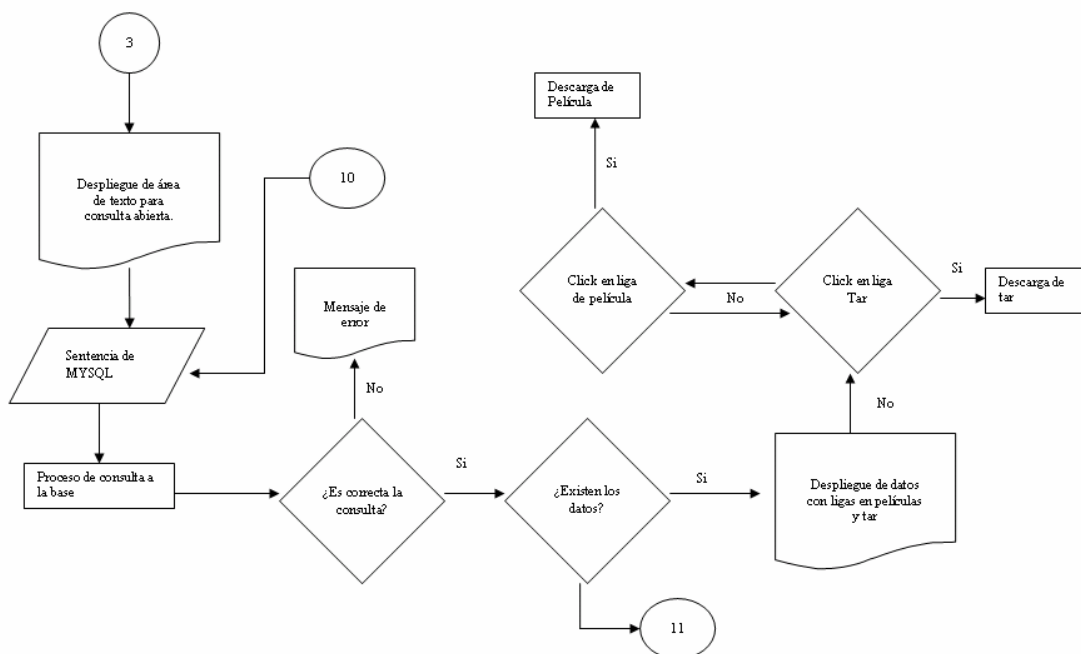
Si dentro del menú de las ligas se da un click en Publicaciones (caso 2), esta desplegara datos que la Institución que tenga el portal considere pertinentes para dicha sección.

Lo mismo ocurre para la elección de la liga Información general (caso 6), que mostrara una serie de información que se considere relevante, además de que podrá incluir (si es que así se decide), datos de las 3 instituciones participantes en este proyecto.

En la liga de ayuda (caso 4), incluirá un pequeño manual para dar a conocer el funcionamiento del sitio, para poder llevar tanto al usuario experto como al inexperto de la mano dentro de este portal.

Pasando finalmente al caso 5 (Sitios de Interés), que se manejara de acuerdo a la institución, ya que se pueden incluir imágenes ligas, información, etc., concernientes a los eventos astronómicos recientes o algunos otros sitios que se considere de importancia.

Fig. 3.3



En la figura 3.3 pasamos a uno de los dos elementos fuertes en el catálogo, que es la búsqueda avanzada, con el simple hecho de que se refiere a avanzado, se requerirá de una menor programación.

Esta liga constara de lo siguiente. Una vez seleccionada esta opción se hará el despliegue de un área de texto, en donde se podrá escribir una consulta abierta en la base de datos, dicha área contendrá una consulta por *default* o inicio que lleva la selección de todos los datos contenidos en la base de datos.

Una vez escrita la sentencia, mediante un click en un botón, dicha petición será enviada, esta se procesa y revisa, apareciendo un mensaje de error en dado caso de que dicha sentencia no sea correcta para el gestor de base de datos (*MySQL*). Si la sentencia es correcta, se verificara que existan los datos. En caso de que los datos no elegidos no

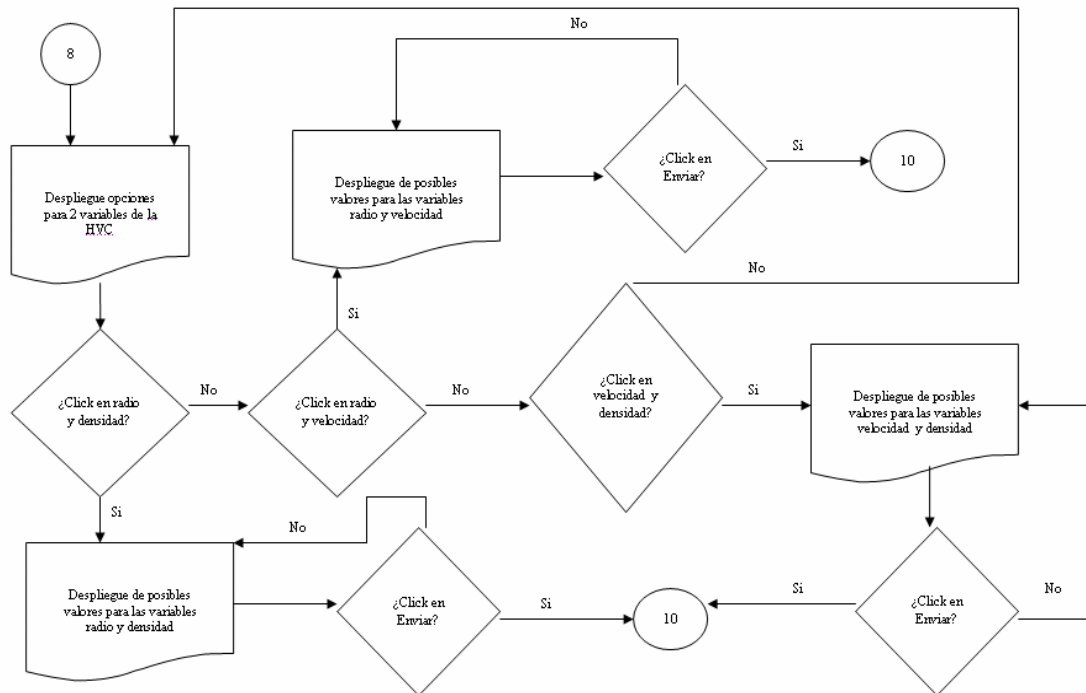




pasara por la sentencia *MySQL*, verificara si los datos existen, y desplegara los datos con sus respectivas ligas para la película o el archivo *.tar.gz*.

En el caso de que sean elegida la opción por 2 variables, nos llevara a la figura 5, y en el caso de “Por todas las variables”, necesitaremos pasar a la figura 6.

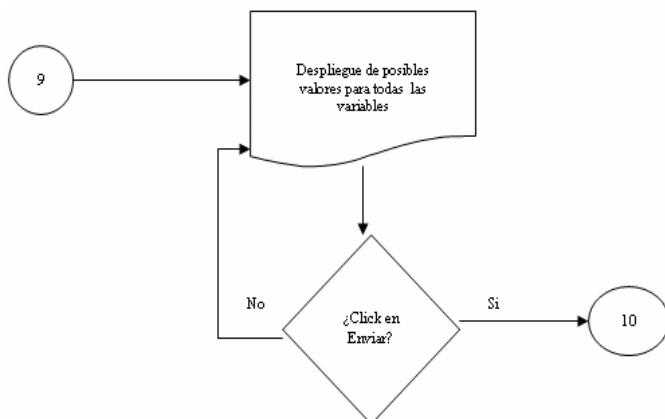
Fig. 3.5



En la figura 3.5, se explica la elección de la opción “Por 2 variables”, que consta de un despliegue de opciones de las diferentes combinaciones que se pueden utilizar, ya sea por radio y densidad, densidad y velocidad o velocidad y radio. En el caso de velocidad y radio, se mostrarán una serie de constantes existentes en la base de datos con las cuales puede ser llenado; aquí se podrá tomar una opción o las 2, y de cada una de ellas se tomara desde una, hasta que no quede ninguna por seleccionar.

Lo mismo pasa para cualquiera de las otras posibilidades, una vez hecho esto pasara al punto que se ha estado describiendo que en el diagrama se asigna como ciclo 10.

Fig. 3.6



Finalmente la figura 3.6 representa lo que se hará en el caso de que sea seleccionado “Por todas las variables”, que consta de la muestra de valores para los datos de velocidad, radio y densidad, en la cual se puede seleccionar de un valor hasta todos los incluidos. Después marcar los deseados, se espera a que se de un click, que llevara a realizar el ciclo 10.

Una vez definidos los módulos y como funcionara cada uno de ellos en el portal de Internet, comenzaremos a la programación de cada uno de ellos, de manera que funcione de la mejor forma.

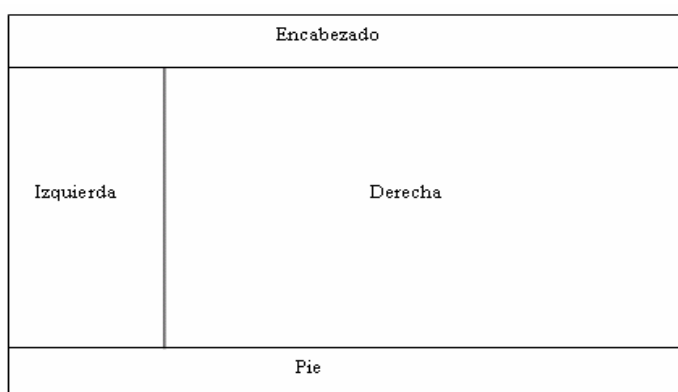
### 3.2.3.1 Presentación

Una vez hecho click en el botón que lleva al catalogo de simulación, aparecerá la página llamada *astros.php*, que contiene en su totalidad código de php, pero por el momento comenzaremos con algo más importante, como lo es la presentación de todas las ligas que incluye este sitio, para mayor referencia, se toma lo que en el punto anterior se hizo, es decir que llevaremos a cabo lo que dice el diagrama de flujo.

La presentación es limpia, y conserva los colores de un inicio (blanco, naranja y azul), contiene 7 ligas y un espacio del lado derecho para mostrar el contenido de cada una.

Pero comenzaremos con la descripción de lo referente a la programación de esta presentación, y lo haré con las funciones que se ocupan para que el correcto funcionamiento. El código incluye 2 archivos con diferentes funciones, el primero que ya fue explicado y que se llama *obvio.php* y el segundo que es un archivo llamado *castros.php*, que contiene el dinamismo del portal, ya que la forma de actuar de este sitio incluye que solo una parte cambie, para poder explicar mejor esto recurriré a un diagrama que nos ayudara a entender como esta dividido, y lo que se pretende hacer, dicha figura hará referencia al monitor y lo dividirá de la siguiente forma (Fig. 3.7):

Fig. 3.7



En donde el encabezado contendrá la imagen, y un letrero, el pie los créditos, izquierda las ligas y cada vez que sea seleccionada una liga se modificara la celda derecha, para que aparezca lo seleccionado ahí.

El archivo *castros.php* esta hecho con pocas líneas de php, y lo demás con *Xhtml*, en sí son dos tablas anidadas, divididas en cuatro funciones (*Encabezado*, *Izquierda*, *Derecha* y *Pie*).

La primera tabla tiene un tamaño del 100%, borde igual a cero, sin espacio entre celdas y lo mismo para el texto y la celda, contiene 7 renglones por 3 columnas, las cuales están conformadas de la forma que se indica abajo (Tabla 3.6):

# de Renglón	Columna 1	Columna 2	Columna 3
1	Vacío, tamaño del 30%, altura de 5 píxeles y fondo de color naranja.	Vacío, tamaño del 40%, altura de 5 píxeles y fondo de color naranja.	Vacío, tamaño del 30%, altura de 5 píxeles y fondo de color naranja.
2	Inserción de imagen 3d.jpg .	Letrero centrado del nombre del portal.	Vacío
3	Vacío con fondo naranja y altura de 5 píxeles.	Vacío con fondo naranja y altura de 5 píxeles.	Vacío con fondo naranja y altura de 5 píxeles.
4	Ocupa 3 columnas y se localiza la tabla anidada.	-----	-----
5	Vacío con fondo naranja y altura de 5 píxeles.	Vacío con fondo naranja y altura de 5 píxeles.	Vacío con fondo naranja y altura de 5 píxeles.
6	Vacío	Letrero centrado con los créditos.	Vacío
7	Vacío con fondo naranja y altura de 5 píxeles.	Vacío con fondo naranja y altura de 5 píxeles.	Vacío con fondo naranja y altura de 5 píxeles.

Tabla 3.6

Con respecto a la tabla anidada, ocupa el 100%, no tiene borde, ni espacios entre celdas ni de celda al texto. Esta compuesta por un renglón y 2 columnas, que contienen (Tabla 3.7):

# de Renglón	Columna 1	Columna 2
1	Ligas en forma de lista.	Código para presentar el contenido de las ligas.

Tabla 3.7

En la tabla 3.7 se encuentra alojado lo más importante del dinamismo del catálogo, ya que en la columna 2 se encuentra lo que lo hace cambiar, y esto es la función *Derecha* cuyo código es:

```
function Derecha($pagina="inicio") //declaración por default del contenido de la
//variable pagina=inicio-->
{
?>
<td >
<?php
$path="./paginas/".$pagina.".php"; //definición del path de las paginas
include($path);
?>
</td>
```

```

        </tr>
    </table><!--fin de la tabla anidada-->
</td>
</tr>
<?php
}

```

La primera línea es la declaración de la función, la cual se llama *Derecha*, y esta siempre tendrá como valor de inicio de la variable *\$pagina* “inicio”, esto quiere decir que cuando se ejecute por primera vez, presentara a *inicio.php*, que se encuentra localizada dentro de la carpeta “paginas” en este sitio. Siguiendo con la explicación se abre la llave de la función, y se cierra *php*, esto con la finalidad de programar con *Xhtml*. Se coloca una columna en el renglón con `<td>`, y posteriormente se vuelve a abrir el *php*, con las líneas:

```

    $path="./paginas/" . $pagina . ".php"; //definicion del path de las paginas
    include($path);

```

En donde la variable *\$path* incluye el *path* donde se localizara el archivo *.php* que se solicite que es en el directorio “paginas” y se llama el archivo *\$pagina.php*, el valor de *\$pagina* será dado por la función *Izquierda* (ver un poco más abajo), y termina con la línea de *include(\$path)*, que quiere decir que se incluirá el directorio. Finalmente se cierra el *php* y se termina cerrando los valores de la columna, el renglón, la tabla anidada, columna y renglón de la tabla principal, en ese orden, para posteriormente cerrar la función con la llave `}`.

La función *Izquierda* es la encargada de enviarle el nombre a la variable *\$pagina*, lo hace con algo como esto:

```

<li><a
                                href="<?php
                                echo
                                $HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=info">Informaci&oacute;n General<br />
</a><br />
</li>

```

En donde `<li>` es la forma de indicar que se trata de un elemento de lista. `<a`, hace que sea un *link*. `href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=info">` quiere decir que hará referencia a la impresión de la variable global *\$HTTP\_SERVER\_VARS* (que es una variable definida por el servidor Web ó directamente relacionadas con el entorno en donde el *script* se esta ejecutando), esta variable en su elemento [*PHP\_SELF*] (esta es el nombre de archivo del *script* ejecutándose actualmente, relativo a la raíz de documentos.), que juntas asignaran el valor a la variable *\$pagina* con `>?pagina=info">`, el valor asignado es *info*, y que ya pulsando un click a esa liga mostrara el contenido de *info.php*. Lo siguiente es *Informaci&oacute;n General*, en donde se despliega el nombre que se verá en este caso *Información General*, (aquí se puede apreciar un *&oacute;*, que quiere decir que se le pondrá acento a la o), después aparece el cierre del *link*, un salto de línea y el cierre del elemento de lista.

**Nota: Todas las ligas contienen el mismo mecanismo, solo que se llaman diferente, para mayor referencia ver el apéndice B.**



```

<tr>
  <td width="33%">
    <br /><br />
    <a href="http://www.astroscu.unam.mx" target="_blank">
      </a><br /><br /><br /> <br /><br /><br />
    </td>
  <td width="33%">
    <a href="http://informatica.aragon.unam.mx" target="_blank">
      <br /><br /> </a>
      <br /><br /><br /> <br /><br /><br /> </td>
  <td width="33%" height="200">
    <a href="http://www.dgsca.unam.mx" target="_blank">
      <br /><br />
      </a>
      <br /><br /><br /><br /><br /><br /> </td>
</tr>
</table>

```

Como se puede apreciar, la primera línea solo se trata de la personalización del tipo y color de letra que se ocupara para mostrar el mensaje de la línea 2 (“PARTICIPANTES PARA EL DESARROLLO DEL SITIO”), en la misma línea uno se encuentran la impresión de espacios (&nbsp;), esto solo por la razón de ajustar el título. En la segunda línea se finaliza con el cierre de las características de fuente que se requirieron para el letrero, y con saltos de línea (<br>), después se identifica una tabla con un tamaño del 100%, sin espacios entre celdas y del texto a la misma, seguidos de el inicio de un renglón (<tr>), y una columna (<td>) con un tamaño del 33% , en donde se hace la creación de un hipervínculo que llevara a la página del Instituto de Astronomía, en una nueva ventana del navegador, esto sin cerrar la propia del sitio (target="\_blank"), y seguido por la imagen que nos llevara hasta la *url* del I.A., que se localiza en el directorio de *img* bajo el nombre de *cuastros.jpg*, en esa parte solo se hacer referencia a que no se quiere un borde que ilumine el contorno de la imagen al ser mostrada en pantalla, posteriormente se cierra el hipervínculo (</a>), y se sigue con una serie de saltos de línea. Lo que se acaba de explicar es lo mismo para las dos imágenes más, salvo con pequeñas diferencias las cuales son la *url* y el nombre del archivo que contiene a la imagen; para el caso de la FES Aragón la *url* es la siguiente: <http://informatica.aragon.unam.mx>, y el nombre de su archivo es *aragon.jpg*, y para DGSCA la *url* es <http://www.dgsca.unam.mx>, con el nombre de archivo de *dgsca2.jpg*.

Como se alcanza a apreciar esto no tiene mayor complejidad, y esta página esta abierta a modificaciones que se requiera por el que haga uso del sitio, ya que se pueden transformar las imágenes y poner algunas más estéticas, cambiar las *url's* o simplemente cambiarlo todo por algo con mejor diseño.

### Sitios de Interés

De acuerdo a lo que se vaya necesitando esta página será llenada, ya que el que ocupe este sitio solo necesitara crear *links* a lugares de Internet que considere pertinentes, esto se logra de una forma muy sencilla, ya que en el directorio *paginas* encontrara un archivo llamado *sitios.php*, el cual puede ser modifica de acuerdo a las necesidades que se tenga, esto para ser llenado con ligas, imágenes, texto, o lo que quiera poner en ese archivo, solo se requiere que se mantenga el mismo nombre en el archivo y que se localice en el directorio *paginas*, o si se quiere cambiar el nombre del archivo solo hay

que modificar en la clase *castros.php*, la liga que hace referencia a sitios de interés, de la siguiente forma:

Localizar la línea:

```
<li > <a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=sitios">Sitios de
inter&eacute;s <br /></a><br />
```

Y cambiar, por lo siguiente:

```
<li > <a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=nuevo nombre">Sitios
de inter&eacute;s <br /></a><br />
```

Guardar los cambios, y almacenar el archivo con el nuevo nombre asignado y con la extensión .php, en el directorio de *paginas*.

### **Información General**

Esta sección, se deja vacía en espera del texto que se quiera incluir, ya sea con imágenes, hipervínculos, o alguna otra cosa que se quiera adicionar.

No se incluye nada aquí debido a que se da la opción de que se personalice el sitio, solo se incluye el archivo *info.php*, con la finalidad de que se vea que funciona esta liga. En ese archivo se almacenaran todos los cambios que se requieran, tan solo con simple programación *html*, o con algún lenguaje de programación.

Este sigue las mismas reglas que la liga de Sitios de Interés en cuestión de almacenaje del archivo y modificación en el nombre.

### **Publicaciones**

Exactamente igual a los anteriores, ya que también se deja abierto para que se personalice, esto porque este espacio esta pensado para ser actualizado constantemente por publicaciones que se hayan hecho en el tema astronómico, o alguna otra cosa que considere importante el *webmaster* para ser colocada ahí.

Las mismas reglas de cambios para las modificaciones que en las anteriores ligas, y el nombre del archivo es *publicaciones.php* alojado en el mismo directorio *paginas*.

### **Ayuda**

Esta es una liga que contiene un tutorial de todo el sitio de Internet, esta desarrollado en 5 fases, cada una de las cuales contiene una sección del portal.

Comenzando por el principal que se llama *ayuda.php* y que su código se encuentra en el directorio *paginas/ayuda.php*. Este archivo esta compuesto por un letrero que indica la bienvenida al manual del catálogo de simulaciones numéricas, y como mencionaba anteriormente 5 secciones o 5 ligas que lo dividen para la explicación de todos los componentes del sitio. Dichas ligas están programadas con el mismo dinamismo mencionado con anterioridad. El nombre de cada una de ellas es Introducción, Páginas de Información, Consultas Básicas, Consultas Avanzadas y Contacto.

Al dar click con el *mouse* en alguna de ellas, nos modificara el lado derecho del usuario en la pantalla apareciendo así el contenido del *link* según se haya seleccionado (para mayor referencia consultar el código fuente *ayuda.php* en el apéndice A).

Siguiendo con la explicación del *link* ayuda comenzaremos diciendo que sus ligas llevan un orden, y por dicho orden se explicara cada uno de estos.

#### Link Introducción

Este lleva por contenido un texto introductorio al manual, además de que tiene 3 ligas más, que llevan por nombres *Inicio*, *Siguiente* y *Atrás*, estas como sus nombres lo indican, se encargaran de llevar al cibernauta por el orden que tiene este tutorial, en este caso *Inicio* llevara a *Ayuda*, *Siguiente* ira a *Páginas de Información* y *Atrás* lo hará también a *Ayuda*.

El código comprende una tabla con 2 renglones y 3 columnas, que almacenan las siguientes características (Tabla 3.8):

# de Renglón	Columna 1	Columna 2	Columna 3
1	Vacío con tamaño del 15%.	Un letrero como titulo y un tamaño del 70% en esa celda.	Vacío con tamaño del 15%.
2	Vacío.	Texto con todo lo introductorio referente al sitio.	Vacío.

Tabla 3.8

Todo esto explica al cibernauta el funcionamiento del sitio y este código se almacena en el archivo *ayuda\_a.php* localizado en el directorio *paginas/ayuda\_a.php* (Para consultar este código ver apéndice A).

#### Link Páginas de Información

Se encarga de explicar las ligas que tiene dedicadas a la información el portal, es decir lleva al usuario al conocimiento de las ligas principales Participantes, Sitios de Interés, Información General, Publicaciones y Ayuda, explicando para que son cada una de estas.

La programación que se utiliza aquí es la misma que la liga Introducción, solo con la modificaciones apropiadas, como el letrero y el contenido del texto, cabe añadir que están incluidas la ligas *Anterior*, *Inicio* y *Siguiente*, que en esta página conducen de la siguiente forma: *Inicio* a *Ayuda*, *Atrás* a *Introducción* y *Siguiente* a *Consultas Básicas*, dicho código esta contenido en el archivo *ayuda\_b.php* que se localiza en *paginas/ayuda\_b.php* (para ver de que se trata consultar apéndice A)

#### Link Consultas Básicas

En esta se trata de explicar al usuario la liga de consulta, y el funcionamiento de la misma, este es el punto en el que hay que tener más cuidado, ya que esta es una parte importante del portal, por que se trata auxiliar en el funcionamiento del mismo, aunque el portal es muy claro y será fácil en esta parte.



El código sigue siendo el mismo que los anteriores ya que se utilizó como machote el código de *Introducción*, es decir que aquí solo vuelve a cambiar el texto del título, y el contenido de datos que explican esta parte. Las ligas *Anterior*, *Siguiente* e *Inicio* solo cambian en referencia ya que *Anterior* lleva a *Páginas de Información*, *Siguiente* a *Consultas avanzadas* e *Inicio* es fija y sigue llevando a *Ayuda*, el código esta contenido en el archivo *ayuda\_c.php* en el mismo directorio *paginas/ayuda\_c.php* (para mayor referencia ver apéndice A).

#### Link Consultas Básicas

Esto explica como debe usar el portal cibernautas avanzados, ya que hace referencia a personas que tengan práctica en consultas de bases de datos *MySQL*. Se considera de suma importancia lo que se explica en esta parte, porque el portal no acepta otra sentencia que no sea un *Select*, además de especificar que no se tiene que mover el contenido que aparece por *default* en el área de texto de la liga *Búsqueda Avanzada*, ya que solo se buscaran los datos deseados y esto puede ser solucionado con un *Where*, y sus acompañantes como puede ser un *And*, un *OR* o la inclusión de ambas, por mencionar algunas formas.

Seguimos con el mismo código manejado en toda esta sección de ayuda y que en este caso modifica el título y lleva diferente texto. La liga *Siguiente* conduce a *contacto*, *Atrás* a *Consultas Básicas*, e *Inicio* a *Ayuda*. El código esta almacenado en el archivo *ayuda\_d.php* que esta en el directorio *paginas/ayuda\_d.php* (ver apéndice A).

#### Link Contacto

Esto solamente es una liga que especifica direcciones de correo para un posible contacto con los *webmasters*, lleva el mismo código, con la diferencia de que este cambia en el título y en el contenido, ya que como especificaba, lleva el nombre de los *webmasters*, en este caso particular será la dirección de correo de la encargada del Instituto de Astronomía la M. en I. Liliana Hernández, y la de un servidor C. Erick D. Valle.

La programación esta contenida en el archivo *ayuda\_e.php* que se localiza en el directorio *paginas/ayuda\_e.php* (ver apéndice A), y cabe señalar que esta es la única página que solo contiene 2 ligas, *Inicio* y *Atrás*, a diferencia de las anteriores que contenían además de estas a *Siguiente*. La liga *Inicio* lleva a *Ayuda*, y la que dice *Atrás* lo hace a *Consultas Avanzadas*.

Hay que hacer referencia también a la parte de inicio del sitio, ya que como mencionaba anteriormente es la primera página que abrirá el catálogo, por lo cual es muy importante dar al usuario una prueba de lo que puede obtener de este catálogo, por ello, el principio consta de una película en formato *mpg* que se desplegara en pantalla, y se reproducirá de forma infinita, esto se hace porque la duración del video es de un segundo, por lo que no se alcanza a presentar de forma clara la proyección de la simulación.

La programación de esto es solo con líneas de texto que se desplegaran arriba y abajo del video, y para lograr que este se proyecte solo se utilizan unas líneas de *html* como es `<embed> </embed>`, con sus atributos que son *autostart*, *loop*, *width*, etc<sup>1</sup>.

---

<sup>1</sup> Manual de HTML, Héctor Leal, Centro de Investigación Cibernética, 2005.

Esto logra una buena presentación ante los ojos del cibernauta, causándole así un interés por consultar más videos de las distintas simulaciones existentes en la base de datos. El código lo encuentran en el archivo *inicio.php* alojado dentro del directorio *paginas/inicio.php* (consultar apéndice A).

### 3.2.3.3 Búsqueda Avanzada

Esta sección esta pensada para los usuarios que tienen mayor experiencia en el manejo de consultas de bases de datos, para ser más específico en *MySQL*, ya que consta de un área de texto para que sea colocada la sentencia de consulta.

El código esta basado en el diagrama de flujo que se localiza en la figura 3, y que comienza con el despliegue del área de texto, respetando el mismo dinamismo que se lleva en el sitio, es decir que solo cambia el lado derecho del portal. Esta programación solo consta de simple *html*, que se encuentra alojado en el archivo *consulta.php*, localizado en el directorio *paginas/consulta.php*. El archivo contiene las siguientes líneas de programación:

```
<br /><br />
<form method="post" action="/paginas/busqueda.php" > <font color="darkblue"> <!--inicio del formulario-->
Escriba su búsqueda: </font><br /><br /> <!--tamaño del area de texto--> <textarea cols="30" rows="15" name="b">
<!--consulta general dada por default-->
SELECT c.id ,n0 ,t0 ,b0 ,n ,r ,v ,tar ,movie FROM archivo d, simulacion c where c.id=d.id
</textarea> <br><br><br>
<input type="submit" value="Enviar"> <!--boton de enviar-->
</form>
<br /><br /><br /><br />
```

El primer renglón solo lleva 2 saltos de línea, para abrir paso al segundo que contiene la especificación de un formulario que mandara los datos por medio del método POST (se encarga de enviar los datos haciendo el primer contacto con el CGI y este recibirá los datos del formulario, para mayor información consultar manual de *html*<sup>2</sup>), y todos los caracteres escritos serán enviados al CGI *busqueda.php*, quien se encargara de procesarlos, después solo se indica el color de la fuente.

La tercera línea es un comentario que permite la mejor interpretación del código. Pasando a la cuarta línea, aparece el texto donde especifica que debe de escribir la sentencia *MySQL* para consultar los datos que el usuario necesita como los ejemplos que se mencionan en el manual de ayuda (Apéndice A *ayuda\_d.php*), esto vienen seguido por el cierre del tipo de fuente, un par de saltos de línea y un comentario, para abrir paso a la quinta línea donde se especifica el tamaño del área de texto que lleva por atributos el número de columnas (*cols="30"*) igual a 30, cantidad de renglones (*rows="15"*) que es igual a 15 y el nombre de la variable (*name="b"*) que es b, llevándonos así al siguiente renglón que es un comentario, para después escribir una consulta por *default* que es “*SELECT c. id ,n0 ,t0 ,b0 ,n ,r ,v ,tar ,movie FROM archivo d, simulación c*”, en donde se puede entender como: Selecciona (*SELECT*) el

<sup>2</sup> <http://es.tldp.org/Manuales-LuCAS/doc-curso-html/doc-curso-html/>

identificador (*id*), la densidad numérica (*n0*), la temperatura (*t0*), intensidad del campo magnético (*b0*), la densidad (*n*), el radio (*r*), la velocidad (*v*), el nombre del archivo *tar.tgz* (*tar*) y el nombre de la película (*movie*) de las tablas simulación y archivo, cuando (*where*) *c.id=d.id* que quiere decir que mostrara cuando sean iguales los campos *id* de las tablas .

Cerrando después el área de texto, junto con 3 saltos de línea, colocando inmediatamente después el botón para enviar la información. Para finalizar cerrando el formulario y colocando 3 saltos de línea más.

Como se puede apreciar esto no es difícil y el diseño fue sencillo, lo complicado comienza con el CGI *busqueda.php*, que es en si el que realiza todo el trabajo.

El contenido de ese CGI esta almacenado en el archivo *busqueda.php*, localizado en el directorio *paginas/busqueda.php*, este contiene casi lo mismo que el motor de búsqueda del sitio, pero entra en diferencia por la intervención de un programa que es ocupado para realizar un *query* en la sección de búsquedas a base de clicks (ver capítulo 3.2.3.4), pero aclararemos de cómo esta compuesto el motor que se encarga de buscar en esta sección.

```
<?php
b=$_POST['b'];
if ($b==""){ //en caso de que la variable del area de texto este vacia
echo "<font color='red' size='+3'>El formulario no puede estar vacío </font>";
exit;
}
$cadena2 = strtolower($b);//convierte a b en minusculas
$divide=explode("select", $cadena2);//divide en arrays a cadena2
if($divide[1]!=""){//si existe cadena2 en su elemento [1] pasa a la busqueda, esto para asegurarnos que no
metan insert o cualquier cosa a la base
$conn = mysql_connect("localhost", "root", ""); //conexion a mysql
mysql_select_db("zeus2");//conexion a la base de datos zeus
$peticion=mysql_query($cadena2); //quersolicitado
if($peticion==""){//si la busqueda no existe
echo "<font color='red' size='+3'>La búsqueda no es valida </font>";
exit;
}
if ($row = mysql_fetch_array($peticion)){ //si existe manda en arreglos la busqueda
echo "<table border = '1'> \n"; //imprime tabla con un borde de un pixel
echo "<center> \n";
echo "<tr> \n";
mysql_field_seek($peticion,0);//devuelve el offset del resultado

while ($field = mysql_fetch_field($peticion) ){ //extrae la informacion de una columna y devuelve
como objeto
echo "<td><b><center>$field->name</center></b></td> \n";
echo "</tr> \n";
do { //repetición hasta que termine con los resultados de la busqueda
echo "<tr> \n";
echo "<td><center>".$row["id"]."</center></td> \n"; //imprime el resultado que es
nombrado como id
echo "<td><center>".$row["n0"]."</center></td> \n";//imprime el resultado de la busqueda
nombrado como n0
echo "<td><center>".$row["t0"]."</center></td> \n";//imprime el resultado de la busqueda
nombrado como b0
echo "<td><center>".$row["b0"]."</center></td> \n";//imprime el resultado de la busqueda
nombrado como t0
```

```

        echo "<td><center>".$row["n"]."</center></td> \n";//imprime el resultado de la busqueda
nombrado como n
        echo "<td><center>".$row["r"]."</center></td> \n";//imprime el resultado de la busqueda
nombrado como v
        echo "<td><center>".$row["v"]."</center></td> \n";//imprime el resultado de la busqueda
nombrado como r
        echo "<td><center><a href='descargartar.php?id=$row[id]'>".$row["tar"]."</center></a></td>
\n"; //imprime el resultado de la busqueda nombrado como tar en una liga
        echo "<td><center><a href='descargarm.php?id=$row[id]'>".$row["movie"]."</center></a></td>
\n"; //imprime el resultado de la busqueda nombrado como movie en una liga
    } while ($row = mysql_fetch_array($peticion) ); //hasta que en la variable row ya no haya ningun
dato de matriz
    echo "</table> \n";
    }
else{
    header("Location: ../astros.php?pagina=regresa");
}
mysql_close($conn); //cierra la base de datos
}
else{//en caso de que no exista cadena[1] quiere decir que no se incorporo un insert
echo "<font color='red' size='+3'>Solo se permite utilizar Select </font>";
}
?>

```

Este código comienza con la apertura del php, después se rescata la variable que fue enviada por medio del formulario, esto se hace por medio de la variable global `$_POST` haciéndolo por medio de lo siguiente `$b=$_POST['b'];` que quiere decir que se le asignara todo lo enviado a través del formulario a la variable `$b`, y comenzamos con la sentencia de control:

```

if ($b==""){
echo "<font color='red' size='+3'>El formulario no puede estar vacío </font>";
exit;
}

```

Esto se hace para prevenir que el usuario envíe en blanco el formulario, es decir si la variable `$b` esta vacía, imprimirá la leyenda “El formulario esta vacío”, y saldrá del programa sin importar las líneas posteriores de código, esto lo hace por medio de la instrucción `exit`.

Si el formulario no esta vacío se saltara las líneas anteriores y pasara directamente al séptimo renglón en donde encontraremos: `$cadena2 = strtolower($b);` que significa que convertiremos todo lo que se recibió del formulario a minúsculas, esto para anticiparnos a cualquier mayúscula que pueda escribir el cibernauta en su consulta, ya que si alguno de los parámetros de la base de datos como por ejemplo n0 es escrito N0, será una consulta no valida y por lo tanto mandara un mensaje de error; es por eso que se tiene esta prevención. Una vez transformada la información enviada, esta será asignada a la variable `$cadena2`.

Como medida de seguridad esta que no se puedan insertar, eliminar y/o cambiar datos en la base, por lo que las siguientes líneas de código lo hacen:

`$divide=explode("select", $cadena2);` esto significa que será asignado a la variable `$divide` elementos de matriz, esto por medio de `explode`, que se encarga de dividir una cadena a partir de un cierto tipo de grupo de caracteres; por lo cual dividimos a partir del grupo de caracteres “`select`”, de la variable `$cadena2`, es decir cada vez que se localiza un `select` en `$cadena2` se cortara la cadena, haciendo que cada búsqueda valida

tenga por lo menos 2 elementos en la variable *\$divide*, es decir que existirá *\$divide[0]* y *\$divide[1]*.

Con base a lo anterior iniciamos un *if else*, que consta de lo siguiente:

```
if($divide[1]!=""){
más código
}else{ echo "<font color='red' size='+3'>Solo se permite utilizar Select </font>";
}
?>
```

Lo anterior se puede interpretar como: La variable *\$divide* en su elemento *[1]*, es diferente a vacío, ingresara a la parte de “más código”, pero si no, desplegara un mensaje de error en color rojo que indicara que solo se puede utilizar *select*.

Una vez terminada esta medida de seguridad, pasaremos a las 2 siguientes líneas que son:

```
$conn = mysql_connect("localhost", "root", "");
mysql_select_db("zeus2");
```

Lo que indica es que en la variable *\$conn* hacemos la conexión al servidor *MySQL*, con los atributos de servidor “*localhost*”, usuario “*root*” y el *password* “”, haciendo después la selección de la base de datos que en este caso es “*zeus2*”.

Una vez hecho lo anterior comenzaremos por enviar la sentencia *MySQL* con lo siguiente: *\$peticion=mysql\_query(\$cadena2)*; esto dice que la variable *\$petición* contendrá el *query* de la sentencia modificada a minúsculas, lo que abrirá el camino para verificar que sea algo valido, y esto lo haremos por medio de un *if*, de esta manera:

```
if($peticion==""){
echo "<font color='red' size='+3'>La búsqueda no es valida </font>";
exit;
}
```

Se interpreta diciendo que si la variable *\$peticion* es idéntica a vacío desplegara un mensaje de error en color rojo diciendo “la búsqueda no es valida”, para posteriormente salir del programa sin importar lo que este en los renglones consecuentes (*exit*);).

Pero si *\$peticion* no es idéntico a vacío pasaremos a las siguientes líneas en donde se desplegaran todos los datos, comenzando con un *if (\$row = mysql\_fetch\_array(\$peticion))*{, que se encarga de verificar si la variable *\$row* existe, cuyo contenido es la instrucción de extraer los datos de la consulta que se localiza en la variable *\$peticion* en una matriz asociativa, y si estos existen imprime una tabla con borde igual a 1, de forma centrada e incluyendo el primer renglón , además de devolver con la instrucción *mysql\_field\_seek(\$peticion,0)*; el puntero al *offset* del resultado, es decir al inicio de *\$peticion*.

Se incluye un ciclo *while* evaluando la variable *\$field* que lleva adentro la palabra reservada: *mysql\_fetch\_field(\$peticion)* que tiene por tarea extraer la información de una columna y devolverla como objeto, es decir este ciclo concluirá hasta que la última columna de *\$petición* sea devuelta. Para completar este ciclo se añade la impresión de una columna (<td>), poniendo el texto centrado y en negrita del objeto *\$field->name*, que en este caso es el nombre de las columnas, por dar un ejemplo, colocara *id* en una columna, *n0* en otra y así con todos los elementos seleccionados en la búsqueda.

Posteriormente imprimirá otro renglón que será acompañado por un ciclo *do while*, que se hace para imprimir un renglón y una columna nuevos por cada sección consultada, es decir llevara a pantalla todos los datos obtenidos por medio del *query* contenido en la variable *\$peticion* de forma centrada y en su respectiva columna cada uno. Se concretara a hacer *link's* a todos los datos pertenecientes a *tar* y *movie*, para poder descargarlos haciendo click (ver 3.2.3.5).

Terminado de imprimir los datos en que se tenían se proseguirá cerrando la tabla y la conexión de la base de datos.

Pero si la variable *\$row* no existe o esta vacía, se ingresara a un *else*, el cual se encargara de redirigir el *script* a *regresa.php*, con las siguientes líneas:

```
else{
    header("Location: ../astros.php?pagina=regresa");
}
```

En donde la palabra reservada *header* es la encargada de redirigir el *script* y *location* especifica la dirección del *script* al cual queremos redirigir. Con la parte *../astros.php?pagina=regresa* indicamos que se redirigirá al *script* *astros.php* que se encuentra en el directorio raíz, y que además le asignara el valor *regresa* a la variable *\$pagina*, es decir, que debido al dinamismo del portal, solo cambiara la parte derecha y ejecutara el *script* *regresa.php* (para mayor referencia ver *regresa.php* en apéndice A).

El código de *regresa.php* no contiene más que un oración que dice: “**la simulación no ha sido realizada, si la desea realizar dar click aquí**”, que es realizada en código *html*, con formato de la fuente y un *<a>*, para hacer liga la palabra *aquí*, la cuál nos llevara al **Portal de Realización de Simulaciones Numéricas**.

Es así como funciona la liga de búsqueda avanzada, encontrando como resultado la impresión de todos los datos requeridos en una tabla con sus respectivos links para descargar los archivos disponibles de las simulaciones numéricas que ya se encuentren realizadas.

### 3.2.3.4 Búsqueda Personalizada

Esta sección esta dedicada a todos los usuarios, en especial a aquellos que no conocen las sentencias para consultar en una base de datos, es por eso que se presenta esta alternativa, para que con el simple hecho de conocer los datos que busca, el usuario pueda por medio de clicks encontrar los archivos de la simulación requerida.

Esta es la parte más fácil de usar para el visitante, pero no quiere decir que lo sea para el programador, ya que esto requiere de mayor atención y precisión a la hora de comenzar a codificar el sitio. Por eso se requiere de diferentes funciones, para poder incluir los datos más comunes en cada variable y estos puedan ser vistos con tan solo dar un click. Cabe señalar que estará dividido como se menciona en la figura 3.4 del capítulo 3.2.3, es decir que se presentara en 2 partes, una que consta de elegir el medio ISM, cuyos datos son la temperatura, la intensidad del campo magnético, y la densidad numérica, esto lo harán escogiendo una constante por cada variable, para posteriormente entrar a la segunda parte, que es escoger la HVC, en lo cual hay varias opciones de búsqueda que puede ser por velocidad, por radio, por densidad, por 2 variables (esto significa que es la combinatoria de las 3 variables correspondientes a la HVC) o por todas las variables. Posteriormente se incorporaran los datos a un programa que se encargara de generar el

*query* correspondiente a la elección de los datos y filtrar este al motor de búsqueda que desplegara los resultados correspondientes.

Para poder empezar a explicar las funciones, es necesario comenzar por el código que llama a todas estas, ya que es el que controla el manejo de cada una de estas. Es por eso que todo el buen funcionamiento depende del archivo llamado *cons.php* (ver apéndice A) alojado en el directorio raíz *paginas/cons.php*. Dicho archivo contiene la programación de la siguiente forma:

Las primeras líneas que incluyen la palabra reservada *include*, se refieren a las funciones que a lo largo del *script* ocuparemos y que llamaremos en el momento que se necesite su funcionamiento. Terminando de definir las, están seguidas por la asignación de las variables recuperadas por el método *POST* que proporciona *html*, dichas variables se distinguen por el signo \$, estas variables son necesarias para cada función ya que en cada fragmento de la consulta se envían datos al servidor, los cuales tienen que ser recuperados, para posteriormente ser enviados e interpretados por otra función, por lo que es necesario que sean recuperados. Dicho lo anterior comenzamos con sentencias de control que es lo que maneja las entradas de cualquiera de los formularios. El primero en ser evaluado es si la respuesta del botón que se encuentra localizado en la selección de por 2 variables es vacía, para aclarar mejor esto cada función despliega un menú, que será elegido de acuerdo como el usuario lo vaya requiriendo, es decir, esta hecho por medio de bucles *if else if*, de manera que dependa del botón enviar, ya que este lleva un contenido y para cada caso el valor es igual, lo único que cambia es el nombre de la variable a evaluar. El primer ciclo es para evaluar la opción por 2 variables, si el contenido de la variable es igual a enviar, entrara a este, llevando dentro del bucle una sentencia de control como lo es *switch*, y juzgara el valor según sea el caso. Si la variable no contiene ese valor, pasara al *elseif* siguiente el cual evalúa el menú de selección de la sección para la elección de cómo quiere buscar si por r, por v, por n, por 2 o por todas las variables, en cuyo caso sí entrara a este ciclo, iniciaremos de nuevo un *switch* que ira a la función dependiendo sea el caso de lo elegido, pero si el contenido es diferente a enviar pasara al siguiente *elseif* que se encargara de desplegar el menú que se menciono anteriormente, y si este es vacío desplegara la primera pantalla que es a selección del medio ISM.

Comenzare por explicar todas las funciones que hacen posible el despliegue de los datos, así como el diseño del sitio.

**Nota: Dentro de los bucles encontrara nombres que pertenecen a cada una de las funciones que se detallaran en los párrafos siguientes.**

### **Funciones Necesarias**

Se puede deducir que toda esta parte del sitio necesita de algunas funciones que se encargan de realizar la búsqueda a base de clicks, es por eso que aquí explicare cada una de ellas, de forma que se pueda captar su respectiva funcionalidad. Para esto nos basaremos en el apéndice A, en la sección del código fuente *cons.php*, que como se explico anteriormente maneja a todas las funciones, por lo cual en el intervalo de las líneas 3 a 12 vienen los nombres de las funciones que se necesitan, y por ese orden comenzare.

## Función ism

Se localiza en el directorio clases bajo el nombre de *ism.php*, y esta encargada de mostrar en pantalla los datos correspondientes a la selección de las constantes de cada variable del medio ISM como lo son la densidad numérica, la intensidad del campo magnético y la temperatura; esto se hace de una forma simple ya que solo incluye unas cuantas líneas del lenguaje php, que son indispensables para especificar que es una función, las líneas restantes es *html*, que esta basado en una serie de tablas anidadas, esto se hace para que pueda quedar ajustado el contenido de cada celda. Contiene también un formulario el cual enviara todo lo seleccionado a la misma página *cons.php* que se encargara de procesar dicha información. Contendrá también tres columnas que son pertenecientes a cada una de las variables, es decir, que la variable densidad numérica será una columna, temperatura otra y la intensidad del campo será otra. Cada una poseerá una serie de constantes que se pueden elegir, esto se hará de un valor numérico por variable, dichos valores están contenidos en un formulario de tipo *radio* (para mayor información ver manual de *html*<sup>3</sup>), ya que este es el que permite que se seleccione una opción solamente en cada variable, además esta dotado de un botón que tiene de nombre *enviar*, y que como su nombre lo indica se encargara de enviar lo seleccionado (para mayor información sobre el código *ism.php* ver apéndice B).

## Función opcion

Esta alojada en el directorio clases bajo el nombre de *opciones.php* y esta encargada de mostrar las opciones que tiene el usuario para elegir una búsqueda de la HVC.

A esta función le tienen que ser enviadas las variables *\$n0*, *\$t0* y *\$b0* que son las correspondientes al ISM<sup>4</sup> y que son ocupadas para desplegar un recordatorio en color rojo de lo seleccionado anteriormente.

Esta basado en *html*, por lo que contiene 3 mensajes, que son el recordatorio del ISM seleccionado en color rojo, un letrero que especifica que se elegirán las condiciones de la HVC<sup>5</sup> y uno que dice “búsqueda por”. Después contiene una tabla con un renglón y 3 columnas, de las cuales solo se ocupa la del centro, en donde se comienza con un formulario a base de botones radio que pide la selección de cómo desea buscar, si por radio, densidad, velocidad, por 2 variables o por todas las variables. Para finalizar se completa e formulario con un botón de enviar, que esta dedicado a conducir la opción que se tomo hacia *cons.php*, que se encarga de procesarla e imprimir la siguiente pantalla de acuerdo a lo que se indico (para consultar el código fuente ver *opciones.php* en el apéndice B).

## Función porN

Se encuentra en el directorio clases bajo el nombre de *porN.php*, y es utilizada en el caso de que sea seleccionada la opción densidad en el menú que aparece en el tipo de búsqueda de la HVC. Esta encargada del despliegue de las posibles constantes para la variable n perteneciente a la HVC, esto llevara al usuario a poder consultar las simulaciones hechas con los valores elegidos de ISM y de n, sin importar la velocidad y el radio de la HVC.

<sup>3</sup> <http://es.tldp.org/Manuales-LuCAS/doc-curso-html/doc-curso-html/>

<sup>4</sup> Ver apartado 4.2.1 del presente trabajo

<sup>5</sup> Ver apartado 4.2 del presente trabajo



Es indispensable para esta función que se le envié el contenido de las variables en el ISM, por lo que serán impresas en forma escondida (ver tipo *hidden* en formularios del manual *html*<sup>6</sup>), para que se respete el contenido de la selección de cada usuario si es que este en algún momento llega a chocar en espacio y tiempo con otros cibernautas que están utilizando este sitio (así se hará para todas las funciones de aquí para adelante).

Esta función contiene código *php* solo para especificar que es una función, y lo demás son líneas *html* que son una par de letreros que especifican que se hará la búsqueda por densidad, y que elija la(s) constante(s). Esto estará desplegado en una tabla con 4 renglones con tres columnas cada uno, en el cual se encuentran especificados los valores que están en las simulaciones contenidos en un formulario del tipo *checkbox* (ver formularios en manual *html*) que permiten que sea seleccionado de uno a todos los valores que están en pantalla, finalizando con un botón de “enviar” que llevara la información a *result.php* (ver motor de búsqueda) y este se encargara de procesarlo (ver *porN.php* en el apéndice B).

### **Función porR**

Similar a la anterior función, se encuentra hospedada en el archivo *porR.php* en el directorio *clases/porR.php*. Esta es utilizada en el caso de que sea seleccionada la opción Radio en el menú que se despliega para el tipo de búsqueda HVC. Esta dotada de las líneas de *php* necesarias para decir que es una función y necesita que le sea enviado el valor de las variables del medio ISM que están colocadas en tipo *hidden* y estas serán arrastradas por la función *opcion* (*opciones.php*).

Contiene *html*, que consiste en desplegar en pantalla un par de letreros que especifican que se hará la búsqueda por medio de la variable radio, y que seleccione una o más de de los valores que se encuentran para esta variable.

Consiste de una tabla con 4 renglones de 3 columnas cada uno, en donde son desplegados de manera uniforme los posibles valores para la variable, los cuales son extendidos gracias a un formulario de tipo *checkbox*.

Cierra esta parte el botón de “enviar” que se encarga de llevar lo seleccionado a *result.php*, que hará el proceso para desplegar los resultados de la búsqueda (ver *porR.php* en el apéndice B).

### **Función porV**

Muy similar a las 2 anteriores funciones, localizada en el directorio *clases/porV.php*, es utilizada cuando en el menú de selección de búsqueda de la HVC es elegida la opción por velocidad. Necesita también a las variables *n0*, *t0* y *b0* del medio ISM.

En el código *html* es exactamente igual a las otras 2 funciones, solo cambiando los valores de las constantes, ya que también cuenta con 2 letreros que especifican el tipo de búsqueda y la selección de la o las constantes.

Cuenta con una tabla de 4 renglones de 3 columnas y un formulario de tipo *checkbox* que permite la selección, para enviarla posteriormente a *result.php*, en donde este hará el despliegue del resultado, cerrando esto con el respectivo botón de “enviar” que es el encargado de llevar los resultados a su destino (ver *porV.php* en apéndice B).

---

<sup>6</sup> <http://es.tldp.org/Manuales-LuCAS/doc-curso-html/doc-curso-html/>

## Función por2

Guardada en el archivo con el nombre *por2.php* que se localiza en el directorio *clases/por2.php*. Es desplegada cuando en el menú de selección de búsquedas de la HVC es elegida la opción por 2 variables, y está encargada de desplegar un nuevo menú de selecciones, que son las posibles combinatorias de 2 de las 3 variables correspondientes a la HVC, es decir, que podrán buscar eligiendo una o más constantes para cada variable de las 2 que saldrán, por lo que se despliegan las opciones de buscar por: densidad y velocidad, densidad y radio o radio y velocidad.

El código consta de lo esencial en *php* para especificar que es una función, y del elemento en el formulario que siga permitiendo el mismo dinamismo que se a manejado en todo el sitio, dicho sea, que le enviará lo seleccionado al código que maneja a todas las funciones en esta sección (*cons.php*), para que este lo procese como sea correspondiente.

Las líneas adicionales de programación son de *html* que consta de un letrero que invita a escoger una de las opciones, y una tabla con 1 renglón y 3 columnas, siendo la de en medio la única ocupada para imprimir las opciones que se mencionaron arriba, que son puestas en un formulario de tipo *radio*, para que solo se pueda ocupar una de las 3 posibles combinaciones, agregando solo un botón que sirve para “enviar” lo elegido por el cibernauta al *script cons.php*.

Cabe mencionar que es necesario para esta función que le sean enviados los valores del medio ISM, ya que este también los heredara a las funciones que depende de ella (para mayor información de esta función ver *por2.php* en el apéndice B).

## Función todas las opciones

Almacenada bajo el nombre de *todas.php* en el directorio *clases/todas.php*, y que es utilizada solo en el caso de que en el menú de opciones para seleccionar el tipo de búsqueda de la HVC sea elegida la opción “por todas las variables”, la cual tiene como función el despliegue de los valores más comunes para las tres variables, y que el usuario pueda seleccionar de uno a n valores por cada variable. Consta en sus primeras líneas las características que marca el lenguaje *php* para indicar que se trata de una función, además de que exige que le sean enviados los valores del medio ISM, ya que los imprimirá como *hidden* y heredara estos al motor de búsqueda *result.php*. Las demás líneas de código consisten en *html* que son el despliegue de 2 letreros en donde especifica como se hará la búsqueda, y la invitación a seleccionar las constantes para las variables. Contiene una tabla de 16 renglones con 3 columnas para cada uno, en donde se distingue bien de que variable se trata debido a una especificación del lado izquierdo de la pantalla. Contiene también un formulario tipo *checkbox* para que pueda elegir de una hasta todas las constantes mostradas en cada variable, y no puede faltar su botón “*enviar*” que hará que todo llegue al motor de búsqueda (ver *todas.php* en el apéndice B para mayor referencia de esta función).

## Función porNyR

Con el nombre de archivo *porNyR.php* y almacenada en el directorio *clases/porNyR.php*. Esta función es llamada en el caso de que sea seleccionada la opción “densidad y radio” en el menú desplegado por la función “por2”, y está encargada de desplegar los posibles valores de selección de estas 2 variables. Consta de líneas que especifica que es una función, y le son heredados los valores del medio ISM

que fueron adquiridos también por la función “por2”, y que esta función le asignara al motor de búsqueda, por lo cual es necesario que le sean enviados.

Lo demás es como todo, ya que consta de un par de letreros que especifican que tipo de búsqueda es, y la invitación a seleccionar los valores; agregándole una tabla con 10 renglones y 3 columnas por cada uno de estos en donde aparecerán las constantes que se pueden seleccionar de las variables densidad y radio, cada una dividida por un letrero en el lado izquierdo de la pantalla. Las posibles selecciones están colocadas en un formulario *checkbox* para que pueda ser elegida una o todas las opciones por cada variable, mencionando también que en el formulario aparecerán escondidos los valores del ISM para poder heredarlos al motor de búsqueda, y con su respectivo botón que llevara las opciones seleccionadas a *result.php* para que este se encargue de mostrar los resultados (ver *porNyR.php* en el apéndice B para mayor información).

### **Función porNyV**

El nombre del archivo es *porNyV.php* que esta en el directorio *clases/porNyV*. Está solo será utilizada si en el menú de la función “por2” es elegida la opción por “densidad y velocidad”, ya que se encargara de mostrar las opciones de estas 2 variables. Esta función es muy similar a la anterior ya que cuenta con el mismo formato de letreros, formularios y opciones escondidas, solo que cambia en los renglones de la tabla ya que este cuenta con 12, debido a que en la variable velocidad contiene más constantes, pero con el mismo número de columnas por renglón, y cambiando solo los valores de las constantes de la variable radio por las de velocidad, pero la finalidad es la misma, ya que también hereda valores al motor de búsqueda, para que este haga su trabajo y traiga los resultados a pantalla (para mayor referencia ver *porNyV.php* en el apéndice B).

### **Función porVyR**

Almacenada en el archivo *porVyR* correspondiente al directorio *clases/porVyR.php*. Esta opción es desplegada en el caso de que sea seleccionada la opción “velocidad y radio” del menú que esta en la función “por2”. Esta encargada de despegar una serie de posibles constantes para las variables ya mencionadas, y enviar estos junto con los datos heredados del medio ISM al motor de búsqueda. Consta de lo mismo que las anteriores funciones, un par de letreros que especifica el tipo de búsqueda, y la invitación a elegir los valores de las variables, además cuenta con una tabla de 11 renglones y 3 columnas, el formulario *checkbox* para elegir de una a n opciones, el botón “*enviar*”, y el correspondiente letrero del lado izquierdo para identificar a que variable pertenecen dichos números, así como también las variables escondidas del medio ISM (ver *porVyR.php* en el apéndice B).

#### **3.2.3.5 Motor de búsqueda**

Como se puede ver en las funciones anteriores, son enviados los datos escondidos en los formularios del medio ISM, así como también los de la HVC, estos últimos que son elegidos por medio de esas funciones al motor de búsqueda que esta en el archivo *result.php* dentro del directorio *paginas/result.php*.

Este como su nombre lo indica, es el encargado de dar al usuario el resultado final, y esto lo hará a través de una conexión a la base de datos, seguida de una sentencia

*MySQL (query)*, y el despliegue de resultados por medio de una tabla con un encabezado por columna del dato correspondiente a la variable y unas ligas para la descarga de la película y el archivo *tar.tgz*, que corresponderán a los datos elegidos por el usuario.

Dicho código es el mismo que el desplegado en la sección de búsqueda avanzada, pero con una gran diferencia ya que este en sus primeras líneas, recibe toda la información de las variables que asigno el usuario por medio de los formularios radio del medio ISM, y del *checkbox* de la HVC, para posteriormente ser transmitidas a una librería que será la encargada de generar un *query* con los parámetros requeridos, que será el mensaje para el servidor, y se pueda obtener una respuesta de la base de datos, que logra ser variante de acuerdo a la existencia de simulaciones con esos datos. En la tercera línea de código se especifica como se llama dicha librería, y que será incluida gracias a la palabra reservada *include*. En los siguientes 14 renglones, es donde se rescatan las variables que pueden ser enviadas, en el caso del medio ISM son 3, esto porque solo puede ser elegido una constante por cada variable, pero en el caso de la HVC, son 11, ya que pueden ser elegidos 3 valores para la variable densidad, 3 para radio y 4 para velocidad (estas pueden ser aumentadas de acuerdo a las necesidades que se tengan); en el renglón siguiente se asigna a la variable *\$b* el contenido que generara la librería del *query*, es decir, se llama a la función (*query.php*) para que haga una sentencia *MySQL* con los datos rescatados de lo enviado por el formulario y lo que se tenga como resultado lo contendrá *\$b*. Acto seguido por la conexión a la base de datos, esto como *localhost* con el usuario *root* y sin *password* alguno (esto puede variar de acuerdo a como se tenga configurado *MySQL*), y la selección de la base que lleva el nombre de “zeus2”.

Una vez hecho lo anterior nombramos una nueva variable que tiene el nombre de *\$peticion*, que llevara en su interior el resultado del *query* hecho con la sentencia de la librería, es por eso que utilizamos lo siguiente: *\$peticion=mysql\_query(\$b)*; ya que *\$b* es la que lleva la sentencia y la palabra reservada *mysql\_query* es la encargada de llevarla hasta la base.

Con lo anterior se asigna una nueva variable llamada *\$existe* que lleva la palabra reservada *mysql\_num\_rows*, que se encargara de devolver el número de filas que genero la variable *\$peticion*, esto se hace para que en lo consiguiente el motor entre en un ciclo *if*, que evaluara si *\$existe* es igual a vacío, y si (*if*) lo es, desplegara un resultado que informara que la búsqueda no es valida y saldrá del programa debido a la instrucción *exit*; pero si no es igual, continuara con el motor y un ciclo *if* más, con la intención de comenzar a desplegar los resultados. Comienza preguntando si (*if*), una nueva variable llamado *\$row* que lleva en sus adentros la palabra reservada *mysql\_fetch\_array(\$peticion)*, que se encarga de hacer que el *query* se convierta en arreglos, por lo que el ciclo pregunta si existe *\$row* (¿esta el *query* en arreglos?), si la respuesta es que no, debido a que no existen los datos que se seleccionaron en la base, entrara a un *else* que ejecutara un *header*, redireccionando el *script* a *regresa.php*. Pero si la respuesta de *mysql\_fetch\_array(\$peticion)* es positiva entrara al ciclo y automáticamente se convertirá la información en arreglos (gracias a *mysql\_fetch\_array*), para posteriormente mandar código *html*, que hará que se despliegue una tabla con tamaño de borde igual a 1, instrucción de centrado y abrir un renglón en dicha tabla. Dentro del ciclo también encontraremos la palabra reservada *mysql\_field\_seek(\$peticion,0)*, que devuelve el puntero al inicio del *query* (para más

referencia de esta instrucción consultar manual de php<sup>7</sup>), dando pie al inicio de un ciclo *while* que se dará a la tarea de evaluar la variable *\$field* que posee la palabra reservada: *mysql\_fetch\_field(\$peticion)* que llevara a cabo la extracción de la información de cada columna y la devolverla como un objeto, por lo cual se ocupa la línea de código *echo "<td><b><center>\$field->name</center></b></td> \n";* que significa: imprime (*echo*) código *html* de una columna(*<td>*) en letra negrita y centrada(*<b><center>*) la variable en su objeto *name* (*\$field->name*) y cierra en *html* el centrado, la negrita y la columna (*</center></b></td>*) aplicando un salto de línea (*\n*), y termina con una línea más que es el cierre del renglón(*</tr>*), este ciclo se repetirá hasta que termine la última columna de la variable petición.

Hecho lo anterior entramos a un nuevo ciclo que será el encargado de desplegar todos los datos de las variables, y se trata de un *do while*, que significa (hazlo, mientras). Este comienza con el *do*, y seguido de la impresión de código *html* que indica un nuevo renglón, después indica que se hará una columna que llevara a la variable *\$row* en su elemento *id*, otra columna con *\$row* en su elemento *n0*, otra columna más en el elemento *n0*, una con el elemento *b0*, con *t0*, *n*, *r*, *v*, el nombre columna más que lleva un *link* de descarga en el elemento *tar* de *\$row*, y otro *link* con el elemento *movie*, estos 2 últimos elementos harán referencia a *descargartar* y *descragarm.php* respectivamente, agregando que enviara el *id* con la finalidad de encontrar el archivo correspondiente (se especificara el funcionamiento de estos *links* en la sección descargas de archivos *tar* y películas), terminando con un *while* (*\$row = mysql\_fetch\_array(\$peticion)*), que quiere decir hasta que terminen los arreglos de la variable *\$peticion* contenidos en la variable *\$row*; esto ocasionara que todos los resultados sean impresos en la pantalla. Una vez que todos los elementos hayan sido desplegados, se cerrara el ciclo *do while* y se hará cerrara el *html* de la tabla con *</table>*, para que enseguida se cierre la base de datos "zeus2" (para mayor referencia motor de búsqueda [*result.php*] en el apéndice A).

### Programa generador de un *query* con n número de variables

Como se menciona en el tema anterior, es necesario que se envíe una sentencia *MySQL* para poder realizar la búsqueda de los datos seleccionados, y para ello se tuvo que realizar un programa que pudiera interpretar los datos enviados por medio de los formularios, y convertirlos a una instrucción que interpretara el servidor de la base de datos.

Comenzamos con los requerimientos que este debe de tener, el cual es que pueda ser actualizable de una forma sencilla, y que no sea tan difícil de manejar para el *webmaster* que se haga cargo del sitio.

Para comenzar a explicar el código hay que partir de que este se usara como una función que mandara a llamar el motor de búsqueda, es por ello, que se incluye en la segunda línea de este. Por tal motivo esta almacenado en el directorio de */paginas*, bajo el nombre de *query.php*.

El programa comienza con la declaración de que se trata de una función, así como también especificar las variables que necesitan ser incorporadas cuando se mande a llamar a esta función, que son las del medio ISM (*\$n0*, *\$t0* y *\$b0*), y las que se

<sup>7</sup> <http://php.net/manual/en/>

incorporarán a través de la HVC que son para la densidad ( $\$n$ ,  $\$n1$ ,  $\$n2$  y  $\$n3$ ), para el radio ( $\$r$ ,  $\$r1$ ,  $\$r2$ ) y para la velocidad ( $\$v$ ,  $\$v1$ ,  $\$v2$ ,  $\$v3$  y  $\$v4$ ), y con esto se termina de definir que se trata de una función. Seguido de esto se abre una llave para comenzar a precisar todo lo que se utilizara, iniciando con la definición de variables, esto con el fin de mantener lo que siempre se ocupará, como son las 2 que se definen a continuación:

```
$inicio="SELECT c.id,n0,t0,b0,n,r,v,tar,movie FROM archivo, simulacion WHERE ";
$medio="c.id=d.id and n0='$n0' and t0='$t0' and b0='$b0' and ";
```

Como se puede observar solo le falta la distinción del valor de cada elemento, y es aquí cuando comenzamos la programación que lo hará.

El método que propongo es por medio de arreglos (esto no quiere decir que sea la única forma de hacerlo) que almacenaran a los datos seleccionados en las variables, por lo que se definen 3, una para cada variable de la HVC que contendrá a todos los datos que haya seleccionado el usuario. Las variables del medio ISM, no se tocan ya que estas solo pasaran un resultado por cada una de estas. Posteriormente se declaran tres variables más que llevan por contenido el conteo de elementos por matriz, es decir, una para contar los elementos de r, otra para v y una más para n, esto con la finalidad de utilizarlos en ciclos, que se explicaran más abajo. Enseguida de eso se declaran 3 arreglos vacíos, que serán llenados dentro de los ciclos. Se comienza con un ciclo *for* para revisar el contenido de los elementos de la variable  $\$n$  de la HVC en la matriz  $\$m$ , auxiliado por el conteo de estos elementos, es decir que este ciclo se repetirá hasta que el incremento de uno por pasada alcance el mismo valor que lo contado. Una vez declarado este, seguirá de forma inmediata un bucle *if* (si), que evaluara sí el arreglo en el elemento *i* es igual a vacío, y si lo es, destruirá ese elemento, pero si no (*else*), lo insertara en una nueva matriz, hasta que el ciclo iguale al número que resulto de contar lo que compone a la matriz de la densidad ( $\$m$ ). Se hace el mismo tipo de ciclo para las variables de velocidad y el radio de la HVC, solo que si el resultado es diferente a vacío se almacenan en matrices distintas. Una vez finalizado estos ciclos, sabremos con certeza que no habrá elementos vacíos en las matrices, por lo que contaremos a las nuevas matrices ingresando los valores resultantes en una respectiva variable por cada una. Se declaran 3 arreglos vacíos nuevamente, procediendo a una nueva sentencia de control para evaluar que es un *if*, que verifica si los nuevos conteos del arreglo  $\$mv$  (matriz velocidad) es idéntico a 0, si lo es, entrara a otro *if* que preguntara si el conteo de la matriz  $\$mn$  (lleva los elementos de la densidad) es idéntica a 0, y si esta lo es, entrara a un nuevo ciclo, que ahora preguntara si el conteo del arreglo  $\$mr$  es idéntico a 0, y si este lo es, habrá una variable que recibe el nombre de  $\$final$ , la cual llevara un mensaje de que el formulario es vacío. La variable  $\$final$  será ocupada para regresar el valor al motor de búsqueda, es por ello que el contenido puede variar, pero no el nombre de la variable.

Siguiendo con el bucle anterior, si el conteo de la matriz  $\$mr$  no es idéntico (*else*) entrara a un *if* anidado, que preguntara entonces (*else*) si (*if*) el conteo de  $\$mr$  es idéntico a 1, y si lo es, la variable  $\$final$  estará compuesta por la concatenación de las variables  $\$inicio$ ,  $\$medio$  y  $r=mr[0]$ , que quiere decir que r será igual al elemento 0 del arreglo  $\$mr$ , pero si no (*else*) es idéntico a 1, entrara a un ciclo *for*, donde tendrá que alcanzar la variable de este ciclo, al conteo de los elementos de  $\$mr$ . Estos en el proceso de alcance guardaran en un nuevo arreglo que se llamara  $\$primera$ , la concatenación de las variables  $\$medio$  y  $r = \$mr[\$n-1]$  (que significa el dato almacenado en el arreglo  $\$n-1$ ), y así sucesivamente hasta que  $\$n$  alcance el valor de lo contado. Ya finalizado este ciclo se saca el primer elemento de la matriz  $\$primera$  y se deposita en la variable

*\$ultima* y posteriormente se cuenta los elementos de *\$primera* que se depositara en la variable *\$mas*. Hecho lo anterior se pasa a un nuevo ciclo *for* que parara hasta que la variable que se incrementa en este caso *\$i* que inicia con un valor de uno, iguale en su valor a *\$mas* y en cada pasada se almacenara en una nueva matriz llamada *\$segunda* la concatenación de *\$mr[\$i-1]* con el texto *or*, hasta que finalice este ciclo. Después de eso se almacena en la matriz *\$segunda* la variable *\$ultima* (contiene el primer elemento de la matriz *\$primera*), para que inmediatamente se haga una cadena separado por un espacio, todo lo que conforma la matriz *\$segunda*, a través de la instrucción *implode* de *php* (ver manual de *php*<sup>8</sup>) y será almacenado en una variable que se llama *\$salida*, para finalizar este bucle se concatenan las variables *\$inicio* y *\$salida* para formar a la variable *\$final*.

Pero en el caso de que los elementos de la matriz *\$mn* (elementos de la densidad), no es idéntico a 0 (*else*), entrara a un nuevo *if*, que esta encargado de evaluar si el conteo de los elementos de la matriz radio (*\$mr*) son idénticos a 0. Si este es el caso, pasara a un nuevo *if* donde preguntara si el conteo de la matriz *\$mr* es idéntica a 1, y si cae en este caso se hace una concatenación de las variables *\$inicio*, *\$medio* y " *n=\$mn[0]*" (significa que *n* será igual al elemento 0 de la matriz *\$mn*), todo esto para formar a la variable *\$final*. Pero si el conteo de *\$mn* no es idéntico a 1, entrara a un ciclo *for* que iniciara a la variable *\$i* en 1 y la incrementara por cada pasada que de, haciendo por cada una de estas una inserción de la concatenación de las variables *\$medio* y lo siguiente: " *n=\$mn[\$i-1]*" (que significa *n* igual a el elemento *\$i-1* del arreglo *\$mn*) a la matriz *\$primera*, para posteriormente sacar el primer elemento de *\$primera* y guardarlo en la variable *\$ultima*, seguido de un conteo de *\$primera*. Una vez teniendo estas 2 variables se comienza un nuevo ciclo *for* que iniciara a la variable *\$i* en 1 nuevamente e incrementara a esta hasta que alcance el número que contó en *\$primera* haciendo en cada pasada una inserción de la matriz *\$mn* en su elemento [*\$i-1*] concatenado con el texto "or" en la matriz vacía *\$segunda*. Ya finalizado este proceso se ingresa lo contenido en la variable *\$ultima* a *\$segunda*, para posteriormente juntar los elementos de esta matriz y convertirlos en una cadena de texto separados por un espacio y depositarlo en la variable *\$salida* que se concatenara con la variable *\$inicio* y juntas formaran a *\$final*, la cual será el *query*.

Pero en el caso de que el conteo de *\$mr* no sea idéntico a 0 (*else*) iniciara un ciclo de *for* anidados, en el que el primer *for* iniciara a la variable *\$i* en 1 y la incrementara por cada pasada que dé, haciendo en el inter de cada una de estas la inserción de la concatenación de la variable *\$medio* con " *n=\$mn[\$i-1]*." and " (que significa *n* igual a el elemento *\$i-1* de la matriz *\$mn*) en la matriz *\$primera* y después de esto entrara en el siguiente *for* que inicializa a la variable *\$j* en 1 y la incrementa en 1 por cada vez que pasa haciendo en el inter la inserción de la concatenación de la matriz *\$mn* en su elemento *\$i-1* con " *r=\$mr[\$j-1]*, hasta que *\$j* sea igual al conteo de los elementos de *\$mr*. Una vez que se acabe este *for* hará una nueva pasada del primer *for* y así sucesivamente hasta que el primer *for* sea igual al conteo de *\$mn*. Ya finalizados estos bucles, se extraerá el primer elemento de *\$segunda* y se depositara en *\$ultima*, procediendo a un nuevo conteo de elementos que componen a *\$segunda*. Después se hará un nuevo ciclo *for* que iniciara a la variable *\$i* en uno y la incrementara por cada pasada en una unidad hasta que iguale al numero resultante del conteo de *\$segunda*, haciendo en cada una de las pasadas la inserción de *\$segunda[\$i-1]* (segunda en su elemento *\$i-1*) concatenado con

---

<sup>8</sup> <http://php.net/manual/en/>

la palabra "or " y será almacenado en la matriz  $\$tercera$ , para después añadirle a esta misma lo contenido en  $\$ultima$ , y así proceder a la conversión de una cadena a la matriz  $\$tercera$ , separando los elementos por un espacio y formar la variable  $\$salida$  que concatenada con  $\$inicio$  formaran a la variable  $\$final$ .

Con esta parte finalizamos la posible combinatoria de que en el conteo tengan o no elementos las matrices  $\$mn$  y  $\$mr$  cuando el conteo de  $\$mv$  es idéntico a 0. En caso de que  $\$mv$  no sea idéntica a 0 pasa lo siguiente.

Si el conteo de  $\$mv$  no es idéntico a 0 (*else*). Pregunta si (*if*) el conteo de  $\$mn$  y  $\$mr$  son idénticos a 0, y solo si ambos son 0 pasara a un siguiente *if* que preguntara si el conteo de  $\$mv$  es idéntico a 1, y si lo es se concatenaran las variables  $\$inicio$ ,  $\$medio$  y el texto "  $v=\$.mv[0]$  (que significa que  $v$  es igual al elemento 0 de la matriz  $\$mv$  ) para formar la variable  $\$final$ , pero si no es idéntico a 1 (*else*) iniciara un ciclo *for* que asignara el valor de 1 a la variable  $\$i$  y la incrementara de uno a uno por pasada hasta que iguale al número que resultado del conteo de  $\$mv$ , insertando la concatenación de la variable  $\$medio$  y el texto "  $v=\$.mv[\$i-1]$  ( $v$  igual al elemento  $\$i-1$  de la matriz  $\$mv$ ) en la matriz vacía  $\$primera$  por cada pasada, para inmediatamente después extraer el primer elemento de esta y guardarlo en  $\$ultima$ , y así contar ahora los elementos de  $\$primera$ , para después iniciar un nuevo ciclo *for* que inicializara como siempre en 1 la variable  $\$i$  y hará el incremento igual que siempre, guardando en la matriz  $\$segunda$  la concatenación de  $\$primera[\$i-1]$  (primera en su elemento  $\$i-1$ ) con la palabra "or" y así sucesivamente hasta que  $\$i$  alcance el valor numérico del conteo de  $\$mv$ , posteriormente se almacena el contenido de  $\$ultima$  en  $\$segunda$  y se procede a convertir en una cadena separando cada elemento por un espacio a la matriz  $\$segunda$  y el resultado lo almacena en  $\$salida$ , para que concatenada con  $\$inicio$  formen a la variable  $\$final$ .

Pero si el conteo de  $\$mn$  y  $\$mr$  no son idénticos a 0 (*else*), preguntara si (*if*) el conteo de  $\$mn$  es idéntico a 0 y  $\$mr$  es diferente de 0, solo si se cumplen las 2 anteriores condiciones evaluara lo siguiente:

Entrara a un ciclo *for* donde iniciara en 1 a la variable  $\$i$  y la incrementara en uno por pasada hasta que iguale al conteo de  $\$mv$ . En cada pasada le insertara a  $\$primera$  la concatenación de  $\$medio$  y  $v=\$.mv[\$i-1].$  and " (el elemento  $\$i-1$  de  $\$mv$  concatenado con la palabra *and*), y estará adjunto un ciclo *for* más, que se encargara de inicializar la variable  $\$j$  en 1 y la incrementara en uno por cada pasada (**Esto se hará para todos los ciclos for**), y en cada vez que se incrementa se insertara en la matriz  $\$segunda$ :  $\$primera[\$i-1]$  concatenada con  $r=\$.mr[\$j-1]$  ( $r$  igual a el elemento  $\$j-1$  de  $\$mr$ ) hasta que alcance el valor numérico contado de  $\$mn$ . Hecho lo anterior se sacara el primer elemento de  $\$segunda$  y se almacenara en  $\$ultima$  procediendo después al conteo de  $\$segunda$ , para iniciar un *for* que almacenara la concatenación de  $\$segunda[\$i-1]$  con la palabra "or", en la matriz  $\$tercera$ , hasta que sea igual la variable inicializada a el conteo de  $\$segunda$ . Una vez hecho esto se guardara el contenido de  $\$ultima$  en  $\$tercera$ , para enseguida convertir a esta última en una cadena y almacenarla en  $\$salida$  que concatenada con  $\$inicio$  formaran a  $\$final$ .

Si no se cumple que el conteo  $\$mn$  sea diferente de 0 y  $\$mr$  idéntico a 0, evaluara el caso contrario, es decir, que  $\$mn$  idéntico a 0 y  $\$mr$  diferente de 0 (*else if*), si este es el



caso, hará lo mismo que el anterior caso solo que en lugar de evaluar a  $\$mn$  lo hará para  $\$mr$ , esto implica que en donde aparezca la letra  $n$ , se cambiara por una  $r$ .

Pero si ninguno de los casos anteriores se cumple para cuando el conteo de  $\$mv$  es diferente de 0 (*else*) se iniciara un ciclo *for* en donde el número a igualar por la variable inicializada es el conteo de  $\$mv$  y en cada pasada se almacenara en  $\$primera$  la concatenación de la variable  $\$medio$  con "  $v=\$mv[\$i-1].$ " and " (v igual al elemento  $\$i-1$  de la matriz  $\$mv$  junto con la palabra "and"). Dentro de este *for* aparece otro ciclo igual que guardara en  $\$segunda$  la concatenación de  $\$primera[\$i-1]$  con  $n=\$mn[\$j-1].$ " and " (n igual al elemento  $\$j-1$  de la matriz  $\$mn$  y la palabra "and") hasta que la variable inicializada  $\$j$  iguale el conteo de  $\$mn$ . Una vez terminados estos ciclos anidados se contara a la matriz  $\$segunda$ , para posteriormente abrir un nuevo *for* que se inicia para vaciar los elementos de  $\$segunda$  y que terminara cuando la variable inicializada  $\$i$  sea igual al número que resultado del conteo de  $\$segunda$ . Para vaciar los elementos se ayuda de otro *for* que se encuentra anidado y terminara hasta que su variable  $\$j$  alcance el conteo de  $\$mr$  por lo que en cada pasada almacenara en  $\$tercera$  la concatenación de  $\$segunda[\$i-1]$  ( $\$segunda$  en su elemento  $\$i-1$ ) con "  $r=\$mr[\$j-1]$ " (r igual a el elemento  $\$j-1$  de la matriz  $\$mr$ ). Ya terminado esto se extraerá el primer elemento de  $\$tercera$  y se vaciara en  $\$ultima$  seguido de contar los elementos de  $\$tercera$ , esto abrirá paso a otro *for* que se encargara de recolectar en  $\$cuarta$  la concatenación de  $\$tercera[\$i-1]$  ( $\$tercera$  en su elemento  $\$i-1$ ) con la palabra "or" que se repetirá hasta que la variable inicializada  $\$i$  alcance el numero contenido en el conteo de  $\$tercera$ . Ya realizado lo anterior, se almacenara lo contenido en  $\$ultima$  en la matriz  $\$cuarta$ , que será convertida en una cadena con un espacio de separación por elemento y llevando a la variable  $\$salida$  que concatenada con  $\$inicio$  formaran a la variable  $\$final$ .

Ya que se tenga la variable  $\$final$  (en cualquier caso de los anteriormente mencionados) será la que se retorne (para ver el código consultar Apéndice B en *query.php*) como sentencia de *query* para que el motor de búsqueda lo ejecute.

## Descargas de archivos tar y películas

Con todo lo anterior se consigue un despliegue de los datos elegidos por el usuario, pero falta explicar como se consigue la descarga de los archivos *.tar* y la película, que es la finalidad de cada simulación. Por ello, nos apoyamos en el motor de búsqueda, y es que recopilando un poco de la explicación del ciclo *do while* que se hace para mostrar los datos, existen un par de ligas que señalan a la información de los archivos comprimidos (*tar*) y otra a la película. En este se exporta el identificador que tiene en la base, que como se sabe es la llave maestra de toda la base. Este dato conduce a un *script* más de *php* que lleva el nombre de *descargartar.php* o *descargam.php* según sea el caso en donde demos click.

Para poder explicar el código que contienen estos 2 *scripts* solo basta explicar uno, ya que en el otro solo se cambiara el *query*. Para esto tomare el archivo *descargartar*, que se encuentra localizado en el directorio */paginas/descargartar.php* (en el mismo se encuentra *decargam.php*), y que tiene en su interior la conexión a la base de datos como *localhost* y seleccionando a la base de datos "zeus2", seguido por una variable que recibe el nombre de  $\$qry$ , que contiene el *query* para buscar en la base de datos, la cual lleva el siguiente texto: "SELECT tar FROM archivo WHERE id= $\$id$ " que se puede

interpretar como: Selecciona la columna *tar* de la tabla datos cuando el identificador sea igual al identificador enviado desde el motor de búsqueda.

Después se le manda dicho *query* al servidor de *MySQL* cuya respuesta será almacenada en la variable *\$res*, que es utilizada para traer el elemento 0 del campo *tar*, que se almacenara en una variable llamada *\$nombre*. Enseguida se le asigna la ruta *../descargas\_t/\$nombre* a la variable *\$path*, esto se hace con la finalidad de revisar si existe el archivo en el directorio *descargas\_t* que es donde se almacenaran todos los archivos *tar* (en el caso de películas el directorio se llama *descargas\_m*) y esta ruta se concatena con la variable *\$nombre*, ya que es el resultado de la búsqueda en la base, por tanto el archivo debe de llevar el mismo nombre.

Ya hecho esto, se revisara si el archivo existe en el *path* y esto se hace con la instrucción de *php file\_exists* (ver manual de *php*<sup>9</sup>) que será un dato verdadero o falso según sea el caso de si existe o no el archivo, el resultado de lo anterior se depositara en una variable llamada *\$fr* que se utiliza en un operador condicional *if*, que preguntara si dicha variable es igual a falso, y si lo es, desplegara en pantalla “el archivo no existe o no se encuentra disponible”, pero si no es igual a falso, enviara unas cabeceras (*header*; ver manual de *php*), en la que la primera especificara el tipo de archivo que es, en el caso de los *tar* será lo siguiente:

*header("Content-type: application/x-zip-compressed")*, que quiere decir que el tipo de contenido es una aplicación *zip* o comprimida (en el caso de las películas será *video/mpeg* que significa un video con formato *mpeg*), la segunda cabecera es para decir el tamaño del archivo y para esto se utiliza lo siguiente:

*header ("Content-Length: ".filesize(\$path))* que dice, “el tamaño del contenido es “, y se concatena con una instrucción de *php filesize*, (ver manual de *php*<sup>10</sup>) que extrae el tamaño del archivo en la dirección *\$path*, que como bien sabemos contiene la ruta del archivo. Y la tercera cabecera hace referencia a la disposición del archivo, haciéndolo de esta forma:

*header("Content-Disposition: attachment; filename=".\$nombre);*

Que dice que la disposición del archivo es como *attachment* (significa que va a pasar el archivo como cuando es enviado con un mensaje por correo) de nombre, y se concatena la variable *\$nombre*, para finalizar con una lectura del directorio o de la variable *\$path*. Con esto se puede disponer de cualquiera de los 2 archivos que se generan ejecutándose las simulaciones numéricas (para mayor detalle ver apéndice B *descargartar* y *descargam.php*) y realizando la edición de la película resultante de los archivos generados con base en la simulación.

---

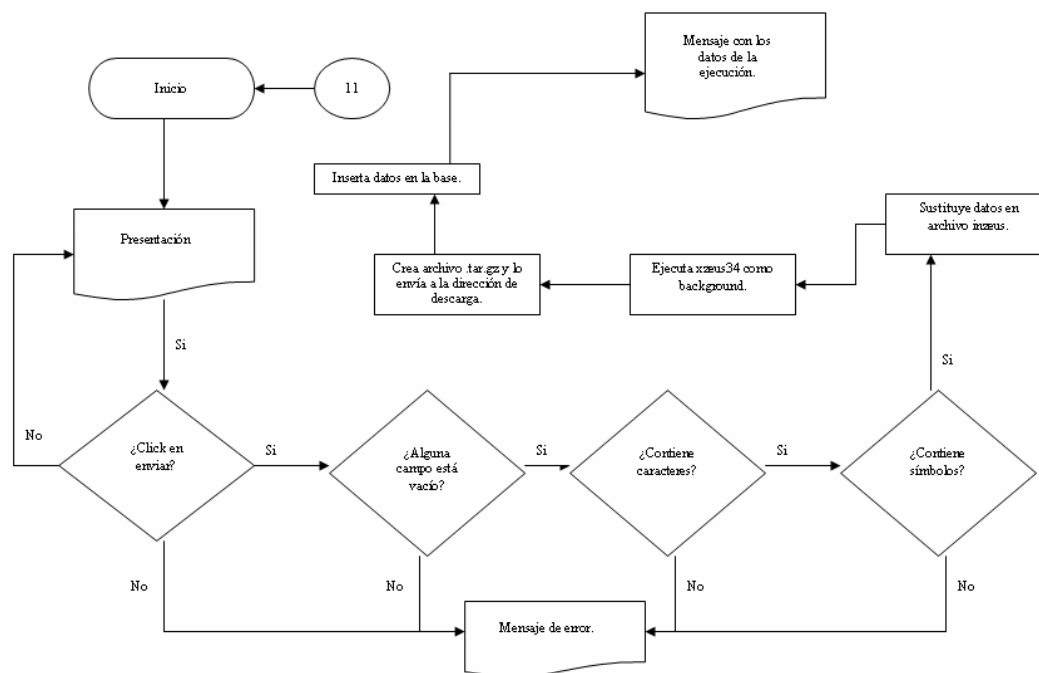
<sup>9</sup> <http://php.net/manual/en/>

<sup>10</sup> <http://php.net/manual/en/>

### 3.2.3 Desarrollo del portal para la realización de simulaciones numéricas

A partir de esta sección se comienza con la segunda parte del sitio de Internet, que consta de las simulaciones numéricas, que nos sirve para que el usuario que no encuentre los archivos con los datos que necesite, pueda hacer su propia simulación. Para realizar esto se tomara como base el siguiente diagrama (Fig. 3.7) que nos explicará a grandes rasgos cual será el algoritmo a seguir.

Fig. 3.7



Como se alcanza a apreciar en la figura 3.7, se comienza con un inicio, o con la llegada del proceso 11 localizado en el catálogo de Simulaciones Numéricas, los cuales nos llevarán a la impresión de pantalla de la presentación del sitio, en el cual se encontrara un formulario donde se pedirán las constantes para las variables, que podrán ser enviadas tan solo con un click en un botón que tendrá el nombre de “enviar”, por ello que se encuentra en un símbolo de decisión, ya que si no es presionado seguirá en la misma pantalla, pero si es pulsado pasara a la fase de depuradores, en donde se preguntara primero si algún campo esta vacío, y si este es el caso mandara un mensaje de error, pero en caso contrario preguntara ahora si algún campo contiene caracteres, esto debido a que solo se aceptaran números y puntos decimales, y si se localizan caracteres en alguna variable, se mandara el mensaje “existen caracteres”, pero si no, pasa a un tercer depurador que se encargara de evaluar si contiene algún símbolo que no sea un punto, mandando un mensaje respectivo si lo logra a localizar, si logra pasar la anterior condicionante, se comenzara a sustituir los valores asignados en el archivo *inzeus*, que es el que contiene todas las instrucciones para que se realice una correcta simulación para posteriormente ejecutar el archivo *xzeus34* (que debe estar dado de alta en el sistema operativo con el alias *.xzeus34*), que se encargara de ejecutar la simulación de una forma *background* o “cortina trasera” (significa que el usuario no verá el proceso), y una vez que finalice el proceso se comprimirán los archivos generados para realizar un fichero *.tar* para que inmediatamente después sea copiado al

directorio *descargas\_t* con el nombre de los datos que selecciono el navegante. Y ya que fue copiado, se insertara en la base de datos, para dar fin al sitio con un mensaje de que esta en ejecución la simulación con el nombre correspondiente.

### 3.2.3.1 Presentación

La presentación del sitio requiere de una serie de funciones que estarán depositadas en un archivo llamado *emula.php* que se encuentra alojado en el directorio *clase/emula.php*, el cual esta encargado de presentar a el formulario que será llenado por el usuario con las constantes en las variables requeridas, dicho archivo contiene a 3 funciones las cuales son *Encabezado*, *muestravariabes* y *Pie*, en las cuales esta dividido el código *html* para desplegar gráficos en la pantalla.

La variable *\$Encabezado* contiene los datos de una tabla que se explican enseguida:

Primero se define que se trata de una tabla, y al hacerlo se le indica que no lleve borde, y que la distancia de la celda a su contenido sea de 0, así como también la distancia entre celdas (esta tabla no se cierra hasta la función *Pie*). En esta instancia la tabla se compone de 3 renglones y 3 columnas que contienen (Tabla 3.9):

# De Renglón	Columna 1	Columna 2	Columna 3
1	Vació con una altura de 5 píxeles relleno de color naranja y un ancho de 30%.	Vació con una altura de 5 píxeles relleno de color naranja y un ancho de 40%.	Vació con una altura de 5 píxeles relleno de color naranja y un ancho de 30%.
2	Imagen 3d.jpg.	Mensaje con tamaño de fuente +2 y en color azul oscuro.	Vació.
3	Vació con una altura de 5 píxeles relleno de color naranja	Vació con una altura de 5 píxeles relleno de color naranja.	Vació con una altura de 5 píxeles relleno de color naranja

Tabla 3.9

Continuando con la tabla pasamos ahora a la función *muestravariabes* la cual esta compuesta de un renglón y una columna, esta última ocupara el tamaño de 3, y dentro de ese renglón y columna aparecerá una tabla anidada que lleva un tamaño del 100% sin borde y sin espacios entre celdas y del contenido a la misma. Esta tabla tiene un contenido de 6 renglones y una variedad de 1 a 4 columnas cuyo contenido es el siguiente (Tabla 3.10):

# De Renglón	Columna 1	Columna 2	Columna 3	Columna 4
1	Mensaje que invita al ingreso de las constantes en las variables con un tamaño de fuente de +1 y un color azul oscuro.			
2	Mensaje que da a entender que en la parte de abajo se elegirán los valores para el medio ISM con tamaño de fuente +1 y de color azul oscuro.		Mensaje que da a entender que en la parte de abajo se elegirán los valores de la HVC con tamaño de fuente +1 y de color azul oscuro.	
3	Inicio de formulario con el método post y que será enviado así mismo, junto con el letrero de densidad numérica, todo alineado a la derecha.	Pequeña área de texto para ingresar la densidad numérica junto con el tipo de unidades todo alineado a la izquierda.	Mensaje con la palabra radio que indica que a su lado debe de ir esta constante todo alineado a la derecha.	Pequeña área de texto para ingresar el radio junto con el tipo de unidades todo alineado a la izquierda.
4	Mensaje con la palabra temperatura que indica que a su lado debe de ir esta constante todo alineado a la derecha.	Pequeña área de texto para ingresar la temperatura junto con el tipo de unidades todo alineado a la izquierda.	Mensaje con la palabra densidad de la HVC que indica que a su lado debe de ir esta constante todo alineado a la derecha.	Pequeña área de texto para ingresar la densidad de la HVC junto con el tipo de unidades todo alineado a la izquierda.
5	Mensaje con la palabra Intensidad el campo magnético que indica que a su lado debe de ir esta constante todo alineado a la derecha.	Pequeña área de texto para ingresar la intensidad del campo magnético junto con el tipo de unidades todo alineado a la izquierda.	Mensaje con la palabra velocidad que indica que a su lado debe de ir esta constante todo alineado a la derecha.	Pequeña área de texto para ingresar la velocidad numérica junto con el tipo de unidades todo alineado a la izquierda.
6	Botón de enviar los datos del formulario.			

Tabla 3.10

Cerrando de esta forma la tabla, junto con la columna y renglón correspondientes a la tabla principal, cabe aclarar que cuando se presiona el botón “enviar”, se manda a sí mismo el contenido de las variables (en cada una de las áreas de texto se le asignan nombres a las variables por ejemplo: velocidad se llama “v”), por lo que son recibidas por el *script* donde son incluidas estas funciones que se llama *simulación.php* que esta ubicado en el directorio raíz, pero este archivo se explicara más adelante.

Se continúa con la función *Pie*, que como su nombre lo indica es el pie de página, y contiene 3 renglones con 3 columnas por cada renglón, distribuidos de la siguiente forma (Tabla 3.11):

# De Renglón	Columna 1	Columna 2	Columna 3
1	Vació con una altura de 5 píxeles relleno de color naranja.	Vació con una altura de 5 píxeles relleno de color naranja.	Vació con una altura de 5 píxeles relleno de color naranja.
2	Vació	Mensaje de Derechos e información del programador.	Vació
3	Vació con una altura de 5 píxeles relleno de color naranja.	Vació con una altura de 5 píxeles relleno de color naranja.	Vació con una altura de 5 píxeles relleno de color naranja.

Tabla 3.11

Cerrando así la tabla principal., y la presentación del ingreso de los datos en pantalla (para mayor detalle ver *emula.php* en el Apéndice B).

Como se menciono arriba, estas funciones son manejadas por *simulación.php*, pero esta cumple 2 funciones, que pueden ser divididas en antes y después de pulsar el botón “enviar”; por lo pronto solo explicare la parte “antes” del click. Dicha sección se encuentra desde el principio del archivo hasta cuando se encuentra el primer *if* y se vuelve a comenzar cuando se cierra la llave del mencionado *if* y comienza un *else*, hasta el final del archivo. En estas partes encontraremos que se incluyen varias librerías como son *obvio*, *emula*, *simbol*, *ceros*, *archivo*, *inter*, y *db*, todas estas con extensión *php*, las cuales serán explicadas en los próximos temas a excepción de *emula* que ya fue explicado. Posteriormente será llamada e impresa la función *InicioPag* (incluida en *obvio.php* que se verá abajo), con el contenido que sus variables de rigurosa necesidad debe llevar, en este caso es el título de la página y el archivo de la hoja de estilo. Pasando al siguiente renglón de código se llama a la función de *Encabezado*, que se desplegara, y si no es oprimido el botón “enviar”, se caerá en un *else*, el cual mostrara en pantalla las funciones *muestravariabes*, *Pie* y *finalPag* (esta última contenida en el archivo *obvio.php*) en este orden, con lo que se consigue que se muestre la solicitud de ingresar datos en las variables.

### 3.2.3.2 Descripción de elementos que siempre son requeridos

Como fue en el caso del catálogo, siempre se requerirán de elementos primordiales en el *html*, como son las declaraciones de inicio, por lo que se volverá a ocupar el código de *obvio.php*, en donde como habíamos mencionado en el tema 3.2.1, se incluye 2 funciones las cuales son *inicioPag* y *finalPag* (ver tema 3.2.1 para consultar su contenido), en la que la primera función es llenada con el título de la página el cual es "Bienvenido al portal de Zeus 3-D", y el *path* donde se encuentra el archivo de la hoja de estilo el cual es el mismo que en el catálogo ("*./libs/estilo.css*"). Con esto aprovechamos funciones ya hechas, que son indispensables para el correcto funcionamiento del sitio.

### 3.2.3.3 Desarrollo de funciones

Como bien se marcaba en la anterior figura 7, esta sección necesita de funciones que están contenidas en librerías para poder obtener los datos sin basura, modificar el archivo que contiene los datos para poder efectuar la simulación, mandar a ejecución las instrucciones necesarias para que se haga dicha simulación y poder insertar los datos en la base, todo esto sin que el usuario se de cuenta de que esta pasando, por ello se

explicaran el contenido de esas funciones y su estructura que son controladas por el código principal (*simulacion.php*).

### 3.2.3.3.1 Depuradores

Como hacíamos mención en el tema anterior hay necesidad de hacer depuradores para obtener los datos sin basura, ya que esta nos generaría un gasto innecesario de espacio en la base de datos, además de crear conflictos a la hora de ejecutar el programa que realizara la simulación e incluso generar simulaciones que ya fueron hechas. Para esto se han creado los siguientes depuradores, que en el caso de las letras o caracteres insertados, esta contenido dentro de la estructura del código principal, y otros más en librerías.

Comenzare explicando el depurador de caracteres y determinación de si alguna variable esta vacía, que esta en la estructura del código principal. Este simplemente radica en un operador condicional *if*, que se puede dar cuenta de donde se localiza por que ya había mencionado que dicho código se dividía en 2 uno desde el inicio hasta el primer *if*, y continuaba hasta donde se cerraba la llave de este y comenzaba un *else* hasta el final. Partiendo de lo anterior nos encontramos con esta parte de código “*if(\$enviar)*” (que significa que si se le da click en el botón de enviar) y enseguida de eso nos toparemos con una llave que abre esta instrucción e inmediatamente después otro *if* anidado que no hace más que preguntar si la variable *\$b0* (intensidad del campo magnético) esta vacía (*empty(\$b0)*) o (*or*) la misma contiene caracteres (*ctype\_alpha(\$b0)*) (“esta instrucción verifica si la variable contiene por lo menos un carácter y envía cierto en caso de encontrar. Ver manual de php<sup>11</sup>”) y agregando un *or* por cada variable a verificar, que en este caso son 6, es decir, que si por lo menos alguna contiene caracteres o esta vacía entrara a este ciclo, en donde se asignara un mensaje de error a la variable *\$muestra*, que será desplegada en la pantalla. Con esto evitamos que se introduzcan caracteres que estorbarían a la hora de la simulación y de falta de datos en esta. Con lo anterior podremos pasar a las librerías que se encargan de corregir los errores o datos innecesarios que puede haber en el llenado de las variables.

### Librería Simbol.php

Ubicada en el directorio *clase/simbol.php*, contiene a la función *símbolos*, la cual se encarga de revisar que no se ingresen símbolos en las variables, ni tampoco más de un punto decimal. Esto comienza con la declaración de que se trata de una función, en este caso *símbolos*, a la que se le debe de proporcionar la variable *\$cadenas*, que no es otra cosa que la cadena que se vaya a evaluar (la variable (*\$b0*, *\$t0*, *\$n0*, *n*, *r* o *v*)). Una vez proporcionada esa variable, comenzara por cortarla cada vez que encuentre un punto, y se almacenara en la variable *\$corta*, por lo que podremos determinar si contiene errores, esto por conducto de una serie de estructuras *if*, que comenzara por preguntar si (*if*) la variable *\$corta[0]* (*\$corta* en su arreglo 0) y (*and*) *\$corta[1]* (*\$corta* en su arreglo 1) están vacías, si es así, regresara el mensaje de que están vacías, debido que por lo menos una de ellas debe existir, pero si no están vacías las 2 (*else*), preguntara si (*if*) *\$corta[0]* es igual a vacía y *\$corta[1]* es diferente de vacío, y si lo es preguntara si (*if*) *\$corta[1]* contiene únicamente números, para esto ocupamos la función de *php ctype\_digit* (ver

---

<sup>11</sup> <http://php.net/manual/en/>

manual de php) y si lo es, regresara una respuesta verdadera, pero si no (*else*), retornara el mensaje “contiene símbolos o contiene más de un punto”.

Pero en el caso de que la condición *\$corta[0]* igual a vacío y *\$corta[1]* diferente de vacío no se cumpla, entonces evaluara (*else*) si (*if*) *\$corta[0]* es diferente de vacío, y (*and*) *\$corta[1]* es igual a vacío, si cae en este caso verificara si (*if*) *\$corta[0]* es de puro contenido numérico, y si lo es regresara un verdadero pero si no (*else*), mandara mensaje de error. Para continuar solo nos queda el caso de que los 2 arreglos de *\$corta* sean diferentes de vacío (*else*), si este es el caso (*if*), preguntara si (*if*) son de contenido puramente numérico, y si lo es retornara un verdadero pero si no (*else*), desplegara mensaje de error, haciendo por último que si no cae en ninguno de estos casos (*else*), que muestre un mensaje de error (para mayor detalle ver *simbol.php* en el Apéndice B).

### Librería Ceros.php

Contenida en el directorio *clase/ceros.php*, y que tiene la finalidad de eliminar los ceros que son puestos sin que haya la necesidad, como puede ser el caso de 001 o 1.200. Como se puede ver no hay necesidad de ingresar tal cantidad de ceros, por lo que hay que eliminarlos. Esta librería contiene funciones que ella misma ocupa para su función principal llamada *ceros*, por lo que comienza declarando la función *derecha* que requiere de una variable (*\$var1*). En esta se declaran dos matrices vacías *\$ma2* y *\$mat2* que se ocuparan durante esta función, posteriormente inicializamos la variable *\$g* en 1, la cual se ocupara durante un siguiente ciclo *for* que inicializa a *\$h* en 0 y hará lo siguiente hasta que sea igual a *\$g*, y en cada pasada incrementara a *\$h*, lo que hará durante este proceso será preguntar si (*if*) la variable *\$var1[\$h]* (*\$var1* en su elemento *\$h*) es diferente de vacío, y si lo es, insertara en la matriz *\$ma2* el elemento *\$var1[\$h]*, e incrementara a la variable *\$g* y así seguirá hasta que sea igual a vacío *\$var1[\$h]*, y una vez que caiga en este caso (*else*) se suspenderá el ciclo *for*. Después contara los elementos en *\$ma2* asignándole dicho número a la variable *\$cuenta4*, para proseguir con otro *for*, en donde se inicia con la variable *\$p=0* y parara hasta que sea igual a *\$cuenta4*, incrementando a *\$p* por cada pasada, y en cada incursión se hará una resta de la siguiente forma *\$resta=\$cuenta4-\$p*, en donde como se ve, el valor quedara en la variable *\$resta*, para posteriormente preguntar si (*if*) *\$ma2[\$resta]* es igual a 0 o (*or*) *\$ma2[\$resta]* es igual a vacío, y si cae en este caso, suprimirá la porción de la matriz *\$ma2[\$p]*, esto con la función *array\_splice*(ver manual de php<sup>12</sup>), pero si no es este caso (*else*) hará un nuevo *for* en donde la variable inicializada es *\$f* y lo hace en 0, que hará que se repita el ciclo hasta que alcance el valor de *resta*, incrementándola en uno por pasada, y en cada ocasión que se ejecute, insertara en la matriz *\$mat2* el elemento *\$ma2[\$f]*, una vez concluido este ciclo se dará salida al anterior *for* con un *break* (ver manual de php) y se proseguirá a convertir en cadena la matriz *\$mat2* y se le asignara este contenido a la variable *\$derecha*, la cual será retornada con la palabra reservada *return* (ver manual de php). Esta función se encarga de la parte decimal del número, por ejemplo, .10000.

Después de haber concluido la función *derecha*, comienza la función *izquierda*, que comienza declarándose como tal y especificando la variable que necesita para funcionar (*\$var*). En los adentros de esta función se declara 2 nuevas matrices vacías que serán ocupadas a lo largo de la programación de este modulo, las cuales son *\$matr* y *\$mas*,

<sup>12</sup> <http://php.net/manual/en/>



después sigue la inicialización de la variable  $\$k$  en 1, para abrir paso a un ciclo *for*, que lleva a la variable  $\$j$  a iniciar en 0, y a parar el ciclo hasta que sea igual a  $\$k$  en donde se incrementara a  $\$j$  en cada pasada, y en cada vez que se incremente  $\$j$  se preguntara si  $\$var[\$j]$  es diferente de vacío, y si así fuera, ingresara en la matriz  $\$matr$  el elemento  $\$var[\$j]$ , después incrementara a  $\$k$ , pero si no es diferente a vacío (*else*), suspenderá el ciclo *for*, dando camino a que se cuenten los elementos de  $\$matr$  y guardando el resultado en  $\$cuenta3$ , para que posteriormente sea abierto un nuevo *for* que inicializa a  $\$l$  en 0 y la incrementara en cada pasada hasta que sea igual a  $\$cuenta3$ ; y en cada vez que se pase preguntara si (*if*)  $\$matr[\$l]$  es igual a 0, en cuyo caso se suprime la porción de la matriz  $\$matr[\$l]$ , pero si no es igual a 0 (*else*), iniciara otro *for* con la variable  $\$q=0$  y parara hasta que sea igual a  $\$cuenta3$ , haciendo en cada pasada una inserción en la matriz  $\$mas$  del elemento  $\$matr[\$l+\$q]$ , y una vez que termine se papara el *for* anterior. Ya hecho esto se convertirá a  $\$mas$  en una cadena con la palabra reservada *implode* (ver manual de php<sup>13</sup>) guardándola en  $\$izquierda$ , para que finalmente esta última sea retornada. Esta función se ocupa para los ceros adicionales antes del punto decimal como por ejemplo 001.

Teniendo lo anterior se da comienzo a la función ceros en donde se inicia con la declaración de esta y de la variable que necesita para funcionar ( $\$cad$ ), para que después se corte dicha variable en donde encuentre un punto, esto lo hace con la palabra reservada *explode* (ver manual de php), que guardara los elementos en la variable  $\$corte$ , después se le asignara a la variable  $\$var$  el contenido de  $\$corte[0]$  (el contenido de  $\$cad$  hasta que encontró el punto) y a  $\$var1$  el contenido de  $\$corte[1]$  (lo que hay después del punto), para posteriormente hacer una nueva separación cuando aparezca un 0 en las variables  $\$var$  y  $\$var1$ , guardando resultados en  $\$corte2$  y  $\$corte3$  respectivamente. Ya teniendo lo anterior se cuentan los elementos de  $\$var$  y  $\$var1$  siendo depositado el resultado en  $\$cuenta$  y  $\$cuent$  respectivamente, después preguntara si (*if*) existe  $\$corte[2]$ , y si existe significa que tiene más de un punto, por lo cual regresara un valor vacío, pero si no (*else*), se declararan 2 matrices vacías ( $\$ver$  y  $\$ves$ ) que serán ocupadas a lo largo del módulo, abriendo inmediatamente un *for* con la variable  $\$i=0$  y que parara hasta que sea igual a  $\$cuenta$  incrementando en uno por pasada mientras que en cada una de estas se preguntara si (*if*)  $\$corte2[\$i]$  es diferente de vacío, en cuyo caso se insertara en la matriz  $\$ver$  el elemento  $\$corte2[\$i]$ , una vez terminado este ciclo, se cuenta los elementos de  $\$ver$  y su resultado será almacenado en  $\$cuenta2$ , después preguntara si (*if*)  $\$cuenta2$  es igual a 0, y si es así se mandara a llamar a la función *derecha* dándole como variable inicial a  $\$var1$ , y su resultado será guardado en la variable  $\$dere$ , pero si no es igual a 0  $\$cuenta2$  (*else*), se hará un *for* con la variable  $\$i$  iniciando en 0 y concluyendo el ciclo hasta que sea igual a  $\$cuent$ , incrementando a  $\$i$  en cada pasada y en cada una de estas, se preguntara si (*if*)  $\$corte3[\$i]$  es diferente a vacío, y si lo es se insertara en la matriz  $\$ves$  el elemento  $\$corte3[\$i]$ . Una vez terminado el anterior *for* se cuentan los elementos de  $\$ver$  y se guarda el resultado en  $\$cuentab$  para poder preguntar si (*if*)  $\$cuentab$  es igual a 0, en cuyo caso se mandara a llamar a la función *izquierda* dándole de variable inicial a  $\$var$  y almacenando el resultado de esta función es  $\$izq$ , pero si no es igual a 0 (*else*) se manda a llamar a *derecha* asignado  $\$var1$  como variable inicial y guardando el resultado en  $\$dere$ , e inmediatamente después se llama a *izquierda* asignándole a  $\$var$  como variable de inicio y guardando el resultado de esta función en  $\$izq$ . Ya con lo anterior se pregunta si (*if*)  $\$izq$  es igual a vacío, y si cae en este caso se le asignara a la variable

<sup>13</sup> <http://php.net/manual/en/>

*\$final* el contenido de la concatenación del texto “0.” con la variable *\$dere*, pero si no cae en este caso preguntara si (*else if*) *\$dere* es igual a vacío, en este caso se le asignara a la variable *\$final* el valor de la variable *\$izq*, pero si no cae en ninguno de estos dos casos (*else*) *\$final* será igual a la concatenación de *\$izq* con el texto “.” y la variable *\$dere*. Para finalizar esta función retornara la variable *\$final*. Es así como se impide que sean integrados ceros inútiles dentro de la designación de constantes a las variables (para más detalle ver *ceros.php* en el Apéndice B).

### 3.2.3.3.2 Modificación al archivo *inzeus*

Una vez que nos aseguramos que los datos están limpios de cualquier basura que pudiera venir anexa, se podrá comenzar a modificar los datos del archivo *inzeus*, en el cual deben de ir todos los datos participantes en la simulación. Para esto se hace uso de una librería llamada *archivo.php*, que será incluida en el manejo del código principal y esta alojada en el directorio *clase/archivo.php*.

Esta librería contiene a la función *inzeus* la cual tiene la tarea de realizar la modificación. En esta, es necesario que se tenga el archivo origen en el directorio raíz, y ya que se tenga ahí, se ocupara para ser modificado y copiado en un nuevo directorio que será creado para esa simulación. Teniendo en cuenta el detalle anterior se procede a explicar el contenido de la librería que comienza con la declaración de la función *inzeus*, la cual necesita que le pasen las variables *\$directorio*, *\$n0*, *\$t0*, *\$b0*, *\$n*, *\$r*, *\$v*, que serán ocupadas a lo largo de este *script*. Después de eso se abre un archivo en la dirección *./\$directorio/inzeus*, que quiere decir que será abierto el archivo *inzeus* que se encuentra en el directorio con el nombre que contenga la variable *\$directorio*, para ello ocupamos la instrucción *fopen* con el argumento anterior y con la propiedad “a+”, que nos da la oportunidad de abrir el fichero para lectura y escritura y crearlo en caso de que no exista (ver manual de php<sup>14</sup>), almacenando esto en la variable *\$fp*, es así como creamos el archivo en el directorio señalado. Ya hecho lo anterior asignamos a la variable *\$archivo* el contenido de “inzeus”, para pasar después a leer el archivo que se encuentra en la variable *\$archivo* y devolver el contenido de este en arreglos con la instrucción *file* de *php*, y el resultado lo guardamos en una variable llamada *\$lineas*. Declaramos una matriz vacía (*\$matriz*) y abrimos un ciclo *for* que inicia a la variable *\$i* en 0 y la incrementara en uno hasta que alcance el valor de 35 (numero total de líneas en el archivo *inzeus* antes de declarar los valores de las variables), realizando en cada pasada la inserción de *\$lineas[\$i]* a *\$matriz*. Ya finalizado el *for* se procede a declara 3 cadenas como se muestra enseguida:

```
$cadena1="      teta_hvc=270.0, dn_hvc=$n, v0_hvc=$v., r_hvc=$r,\n";
$cadena2="      dn0=$n0, b10=$b0.e-6 , b20=1.e-6, t0 = $t0.          $\n";
```

En donde estas son las 2 últimas líneas que lleva el archivo *inzeus* y en las cuales se asigna los valores de las variables (ya pasadas por los depuradores) dadas por el usuario, y que serán insertadas en *\$matriz*, para dar lugar a contar los elementos de esta y que su resultado se guarde en *\$cuent*, con la finalidad de abrir un nuevo *for* que iniciara a la variable *\$k* en 0 y la aumentara en uno hasta que alcance el valor de *\$cuent*, haciendo en cada pasada la escritura de *\$matriz[\$k]* en *\$fp* (con la ayuda de *fwrite*, ver manual de php). Y así finaliza la sustitución de los nuevos valores de las variables en el archivo *inzeus* (para más detalle ver *archivo.php* en el apéndice B).

<sup>14</sup> <http://php.net/manual/en/>

### 3.2.3.3.3 Instrucciones a ejecutar

En esta parte se ven las instrucciones que se va a realizar para poder mandar a llamar al ejecutable *xzeus34*, y para esto hay que preparar el terreno con lo que se utilizara, por lo cual hacemos uso de instrucciones en la estructura del código principal y de una librería. Las instrucciones en el código principal las dejaremos pendiente por el momento ya que se explicaran en el tema (3.2.3.4), y se vera en este la librería que se ocupa para mandar las instrucciones.

Dicha librería recibe el nombre de *inter.php* y se localiza en *clase/inter.php*. Esa aloja a una función llamada *crea*, que se encargara de crear un archivo con el nombre de *back.php* en el directorio raíz, que será el *script* enviado a ejecución en cortina trasera (*background*). Para poder hacer esto se comienza a desarrollar en *inter.php* la declaración de la función *crea* que necesitara de las variables *\$nombre* y *\$directorio*, posteriormente se le asigna a la variable *\$archivo* el contenido de “./back.php”, para después abrirlo con el mismo método ocupado para *inzeus* y guardando el resultado en *\$fp*, enseguida se hace una variable con el nombre de *\$contenido* que lleva lo siguiente:

```
$contenido="<?php
chdir('./".$directorio."");
exec('./xzeus34');
exec('tar -zcvf zhto."$nombre.".tgz zhto*');
exec('cp zhto."$nombre.".tgz ../descargas_t');
exec('rm zhto0* -f');
?>";
```

En la primera línea se asigna a la variable *\$contenido* la cadena “<?php, que indica que se trata de un *script* de *php*, en la segunda línea se ve la instrucción *chdir* que sirve para cambiar el directorio actual a el directorio *./\$directorio*, en la tercera se manda a llama al ejecutable *xzeus34* (alias *./xzeus34* en el sistema operativo) contenido en el nuevo directorio, para después ejecutar la instrucción del sistema para realizar un archivo *tar* con los ficheros generados por la simulación con la extensión *.zhto* y que recibirá el nombre de *\$nombre.tgz*, todo esto con la instrucción de *php* “exec“. En la siguiente línea se manda a ejecutar la instrucción del sistema *cp* que sirve para copiar el archivo *tar* creado al directorio *../descargas\_t*, y la siguiente línea se encarga de remover (*rm*) los archivos que ya fueron incluidos en el tar (todos los que contengan *zhto0*), y finaliza cerrando el *script php* (ver *inter.php* en el apéndice B).

### 3.2.3.3.4 Inserción a la base de datos

Después de haber corrido las instrucciones para la simulación es necesario almacenar los datos que fueron empleados por la misma en la base, y para ello ocuparemos una librería más que recibe el nombre de *bd.php*. Dentro de esta librería que se encuentra alojada en *clase/bd.php*, existe una función llamada *base* que declara las variables que necesita para su correcto funcionamiento (*\$nombre*, *\$n0*, *\$t0*, *\$b0*, *\$n*, *\$v*, *\$r*), para posteriormente abrir una conexión como *localhost* y un usuario *root* a la base de datos “zeus2”, y luego declarar la inserción de los datos en un par de cadenas que están contenidas en las variables *\$peticion* y *\$peticion2*, para que al final se manden con la instrucción *mysql\_query(\$peticion)* y *mysql\_query(\$peticion2)* la inserción de los datos correspondientes a cada campo de la base (Ver *bd.php* en el apéndice B).

### 3.2.3.4 Estructura del código principal

Como habíamos mencionado en temas anteriores este código se almacena en el archivo *simulacion.php* ubicado en raíz y es el que controla a todas las librerías anteriores en esta sección de la realización de simulaciones numéricas. Anteriormente también se había explicado el contenido del principio de este código, por lo que retomaremos en la parte que se dejó que fue el primer *if*. Así que comenzamos en esa parte y es que se pregunta si (*if*) la variable *\$enviar* es cierta, es decir, si fue pulsado el botón “enviar”, si este fuera el caso, preguntara ahora si (*if*) alguna de las variables contiene caracteres o esta vacía, y si así fuera, mandara el mensaje de error correspondiente, pero si no (*else*), se mandara a ejecutar la función *símbolos* por cada una de las variables y se almacenara en otra cómo por ejemplo:

```
$checan0=simbolos($n0);
```

El resultado que dé la función *símbolos* que evaluara a la variable *\$n0* se almacenara en *\$checan0*, y esto se hará para las 5 variables restantes (*t0*, *b0*, *n*, *v* y *r*). Concluido lo anterior preguntara si (*if*) alguno de los resultados de la función *símbolos* es diferente de vacío, y si cae en este caso, se mandara a pantalla el error correspondiente, pero si no (*else*), se llamara a la función *ceros* de la siguiente forma:

```
$sincerosn0=ceros($n0);
```

Que quiere decir que se le enviara la variable *\$n0* como elemento a la función *ceros*, y el resultado que se obtenga se guardara en *\$sincerosn0*, haciendo lo mismo por cada variable faltante (*t0*, *b0*, *n*, *v* y *r*). Con el resultado de la función *ceros* para cada variable se evalúa si (*if*) alguno es igual a vacío, y si lo es, se almacenara el mensaje de error correspondiente en la variable *\$muestra*, pero si no hará la siguiente asignación a la variable *\$matriz*:

```
$matriz=$sincerosn0."_".$sincerosb0."_".$sincerosn0."_".$sincerosr."_".$sincerosv;
```

Para posteriormente concatenar el texto “*RUN\_*” con la variable *\$matriz* y que sea guardado en *\$nombre*, y este ultimo ayudara a formar a la variable *\$path* junto con la cadena “*./*” de esta forma “*\$path=“./”.\$nombre;*” . Ya formado lo anterior se preguntara si (*if*) *\$path* existe, y si existe, se puede decir con certeza que la simulación ya fue hecha, por lo que este mensaje se almacenara en la variable *\$muestra*, pero si no (*else*), se creara con la instrucción *mkdir* (ver manual de php<sup>15</sup>), el directorio *\$path* con permisos de lectura y escritura 0776 (*chmod* de linux) y será almacenado en la variable *\$crear*. Se preguntara si (*if*) *\$crear* tuvo éxito, y en este caso se proseguirá llamando a la función *inzeus* de la siguiente forma:

```
“inzeus($nombre,$sincerosn0,$sincerosb0,$sincerosn0,$sincerosr,$sincerosv);”
```

Asignándole las variables que necesita para poder realizar su objetivo, y después se llamara a ejecución las instrucciones que habían quedado pendientes de explicar, como son las de consola que se hace de esta forma:

---

<sup>15</sup> <http://php.net/manual/en/>

```
system('cp xzeus34 ./.$nombre./xzeus34');
```

En donde *system* es la palabra reservada de *php* que manda a ejecución en consola el comando *cp*, que se encargara de copiar el ejecutable *xzeus34* en el directorio *./\$nombre/*, dando nombre de *xzeus34* al archivo. Se llamara después a la función *crea* con las variables *\$matriz* y *\$nombre*, que le son indispensables para ser ejecutada, para que después se llame a la función *base* y esta ingresara los datos en la base de datos, se llamara con las variables necesarias (*\$matriz,\$sincerosn0,\$sinceros0,\$sincerosb0,\$sincerosn,\$sincerosv,\$sincerosr*) para que se obtenga el resultado deseado con esta función y ya teniendo lo anterior se manda la instrucción clave de la realización que es :

```
exec("/usr/bin/php ./back.php > /dev/null 2>&1 &");
```

La cual indica que se ejecutara como cortina trasera el *script back.php* localizado en raíz y que será el proceso hijo, el cual será esperado tardando la ejecución del proceso padre que en este caso es *simulación.php*, esto debido a la instrucción *exec* de *php*, por lo cual dirigimos la salida del proceso hijo a */dev/null* y las corrientes de entradas y salidas (*stdin* y *stdout*) con *2>&i &*, y con esto conseguimos bifurcar o converger el proceso hijo sin obstaculizar al padre<sup>16</sup>. Con lo anterior, se prosigue a darle una cadena a la variable *\$muestra* que indique que la simulación esta en proceso. En el caso de que la creación del directorio no tenga éxito (*else*), la variable *\$muestra* llevara un mensaje de error. Se finaliza el caso de que sea cumplido al apretar el botón “enviar”, desplegando en pantalla la variable *\$muestra*, seguida de las funciones *Pie* y *finalPag* (ver *simulación.php* en el apéndice A).

---

<sup>16</sup> <http://www.welldonesoft.com/technology/articles/php/forking/>

## CAPÍTULO IV APLICACIÓN DEL PORTAL UNIVERSITARIOS A UN PROBLEMA ASTROFÍSICO.

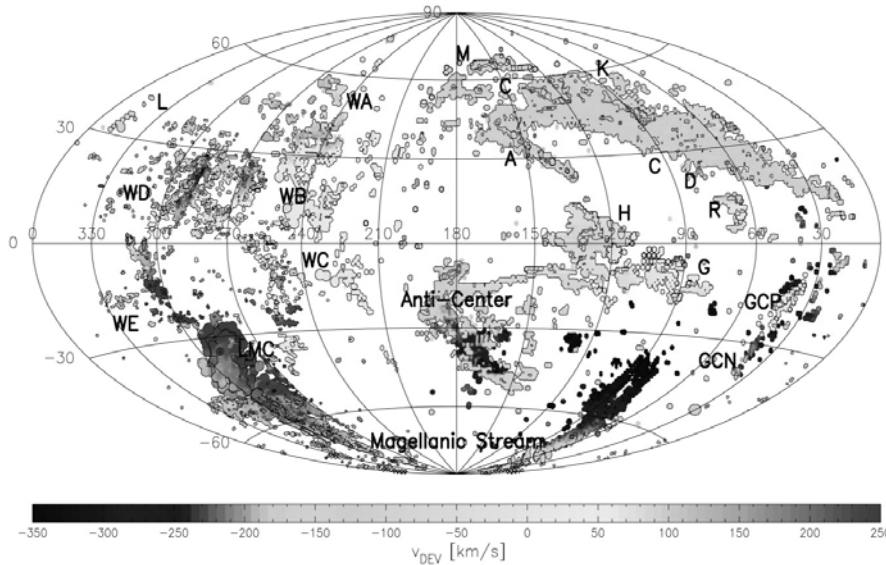
### Introducción.

Como vimos en el capítulo 2 las simulaciones numéricas son indispensables para el desarrollo de proyectos de investigación en áreas científicas y de ingeniería. Hasta la fecha, muchas personas que trabajaban con modelos numéricos realizan sus simulaciones y una vez obtenidos los resultados de su interés desechaban los cálculos. Con la herramienta que aquí presentamos tenemos la intención de reformar esa manera de trabajar, para que todos los cálculos realizados se almacenen y puedan ser utilizados por grupos de investigación, nacionales o extranjeros, independientemente del lugar de dónde se encuentren, necesitando solamente una conexión a Internet. En el caso particular de este trabajo tomaremos un caso astrofísico como ejemplo; la evolución de flujos de hidrógeno atómico en la Vía Láctea.

### 4.1 Nubes de Alta Velocidad o Flujos Supersónicos de Hidrógeno Neutro.

Las nubes de alta velocidad (HVC, por sus siglas en inglés) fueron descubiertas por los astrónomos Muller, Oort y Raimond en 1963 y se definen como enormes flujos de gas, principalmente compuestos de hidrógeno atómico neutro, que presentan velocidades radiales que no pueden ser representadas por modelos de rotación diferencial aplicados a nuestra Galaxia. La mayoría de las HVCs que se han observado muestran velocidades radiales preferentemente negativas, es decir, que sus movimientos están dirigidos hacia el observador. Estos flujos de gas se localizan en las partes más altas de nuestra Galaxia y viajan a velocidades mayores a los 90 km/s (ver figura 1). Hasta el momento, estos objetos son un gran misterio para astrónomos que se dedican a su estudio, ya que no han podido determinar con precisión su origen. Algunos grupos de investigación creen que es gas eyectado del disco galáctico por vientos estelares y múltiples supernovas y que ahora están cayendo desde el halo; otros piensan que son fragmentos de brazos espirales lejanos, en el disco galáctico fuertemente alabeado; también se cree que es material que ha sido robado a galaxias vecinas por interacción gravitacional; como es el caso de la Vía Láctea quitándole material a una de sus galaxias satélite, la Nube Mayor de Magallanes dando origen a la conocida *corriente de Magallanes* (ver figura 1). Por otro lado, el tener una descripción detallada de su evolución y su interacción con el disco de la Vía Láctea, por medio de modelos numéricos, nos pueden dar información sobre la formación de grandes estructuras en el plano de nuestra Galaxia (Santillán, et al., 1999, 2004), además de poder ser un excelente mecanismo para explicar la turbulencia, o agitación del gas, en las partes externas del disco de las galaxias espirales. (Sánchez-Salcedo, F. J.; Santillán, A.; Franco, J., 2007).

Fig 4.1



La figura 4.1 muestra la distribución de las HVCs en nuestra Galaxia. Allí podemos observar que estos objetos presentan velocidades de cientos de kilómetros por segundo. La barra de colores que se encuentra en la parte inferior representa la escala de velocidades de las nubes que van de -350.0 a +250.0 km/s (Wakker et al 2003).

#### 4.2 Portal Universitario de Simulaciones Numéricas: Nubes de Alta Velocidad.

Recordemos que una de las metas de este trabajo es proporcionar un *Catálogo de simulaciones numéricas vinculadas a las HVCs* con el propósito de que la comunidad científica astronómica internacional, que observa estos objetos, pueda tener a la mano un modelo magnetohidrodinámico que les ayude a describir de forma general la evolución de las nubes en el Medio Interestelar de una galaxia. El Portal que hemos descrito en capítulos anteriores es completamente transparente para el usuario; y no solo eso sino que cualquier investigador que utilice este Portal, no necesitará tener conocimiento de códigos numéricos complejos, indispensables para realizar cálculos numéricos, sino que únicamente tendrá que llenar una serie de campos (asociados a las propiedades físicas de las HVCs y del medio donde evolucionaran) y dar un click para obtener una simulación del problema de su interés. A continuación daremos una descripción de la información física que contiene el Portal.

Cuando el investigador accede al Portal, éste tiene dos opciones:

1. Realizar un Cálculo Numérico Remotamente con determinadas condiciones iniciales o
2. Entrar directamente al Catálogo de Simulaciones Numéricas.

##### 4.2.1 Simulaciones Numéricas Remotas.

En la figura 4.2 se muestran los campos que deberá llenar el usuario y que están relacionados con las condiciones iniciales tanto del Medio Interestelar (ISM, por su

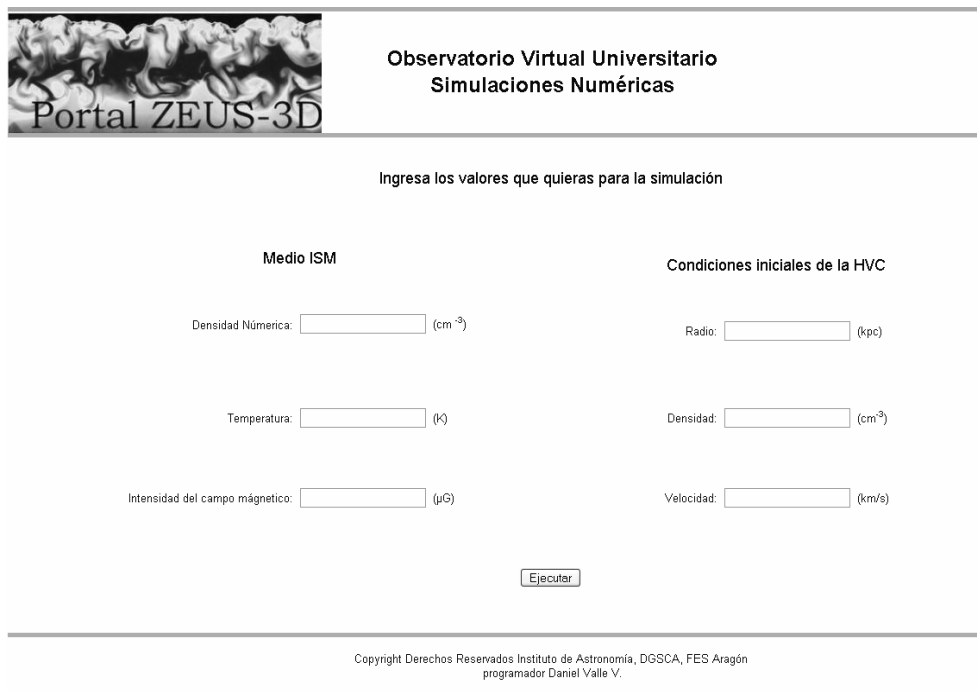
siglas en inglés) donde evolucionaran las nubes, como de las HVCs que quieran simular.

<b>Medio Interestelar.</b>		
<b>Cantidad Física</b>	<b>Valores Típicos</b>	<b>Unidades</b>
Densidad Numérica	1	Partículas/cm <sup>3</sup>
Temperatura	10,000.	Grados Kelvin
Intensidad del Campo Magnético	3	10 <sup>-6</sup> Gauss

Valores típicos del Medio Interestelar se muestran en la tabla 4.1.

Cuando el usuario da un click en el botón *Ejecutar*, el sistema analizará las condiciones iniciales, si estas existen, mostrara el mensaje: “Esta simulación ya fue hecha. Si desea los archivos de clic *aquí*”, siendo la palabra *aquí* un hipervínculo hacia la página principal del catálogo en donde se podrán extraer los archivos de la simulación para ponerlos a disposición del investigador. Este proceso permitirá liberar tiempo de procesador, ya que no se realizan cálculos numéricos, y por lo tanto, podrá utilizarse para una realizar una simulación que no existe en el catálogo. En caso de que no existan las condiciones iniciales se ejecutará el código ZEUS.

Fig. 4.2



**Observatorio Virtual Universitario  
Simulaciones Numéricas**

Ingresa los valores que quieras para la simulación

<p><b>Medio ISM</b></p> <p>Densidad Numérica: <input type="text"/> (cm<sup>-3</sup>)</p> <p>Temperatura: <input type="text"/> (K)</p> <p>Intensidad del campo magnético: <input type="text"/> (µG)</p>	<p><b>Condiciones iniciales de la HVC</b></p> <p>Radio: <input type="text"/> (kpc)</p> <p>Densidad: <input type="text"/> (cm<sup>-2</sup>)</p> <p>Velocidad: <input type="text"/> (km/s)</p>
--	--

Copyright Derechos Reservados Instituto de Astronomía, DGSCA, FES Aragón  
programador Daniel Valle V.

Figura 4.2 Muestra los campos que determinan las condiciones iniciales del Medio Interestelar y de las Nubes de Alta Velocidad.



### 4.2.2 Catálogo de Simulaciones Numéricas.

Una de las tareas innovadoras de este trabajo es la de liberar tiempo de procesador para que la computadora que hospede el Portal no se sature y pueda realizar su función de forma eficiente. Esto se logra almacenando las simulaciones en una base de datos que aquí llamaremos **Catálogo de Simulaciones Numéricas**. Éste traerá como consecuencia una opción extra para el usuario ya que le permitirá acceder a la información almacenada en el Catálogo, es decir, tendrá a su disposición todas las simulaciones que ya se hayan elaborado y almacenado hasta el momento, y podrá disponer de ellas por medio de consultas básicas o avanzadas. El Catálogo está compuesto básicamente de una serie de archivos binarios en formato *HDF* (*Hierarchical Data Format*<sup>1</sup>), formato especial para almacenar datos científicos. Para cada simulación se tienen 10 ficheros almacenados en un solo archivo de nombre *zhtoXXX.tar.tgz* allí se encuentra toda la hidrodinámica del desenvolvimiento de las HVCs en el ISM. Cada fichero representa un tiempo *t* de la evolución de las nubes y contiene todas las variables físicas que son de interés para el investigador: *campo de velocidades, campo magnético, densidad y energía interna*. En el mismo Catálogo el usuario tendrá a su disposición una animación en formato MPEG que le permitirá al usuario ver toda la evolución del fenómeno astrofísico en unos cuantos segundos, esta la podrá descargar con un solo click.

### 4.3 Consultas Básicas.

#### 4.3.1 Características del Medio Interestelar.

Si el usuario no está interesado en realizar una simulación en particular sino que simple y sencillamente quiere tener acceso al Catálogo de Simulaciones, este lo puede hacer sin ningún problema, de hecho, puede consultar desde una hasta todas la simulaciones de la base de datos y lo puede hacer escogiendo únicamente las que le interesan con ciertas propiedades, por medio de búsquedas avanzadas.

Primero tiene que elegir el medio donde van a evolucionar la HVC. Las opciones se muestran en la figura 3, los rangos de valores de la densidad, temperatura e intensidad del campo magnético se pueden consultar en la Tabla 4.2.

Figura 4.3 Elección que puede hacer el usuario del medio Interestelar donde evolucionaran las HVC's.

	Densidad (partículas/cm <sup>3</sup> )	Temperatura (K)	Intensidad del Campo-B (10 <sup>-6</sup> Gauss)
Rango	0.01 - 10.	10. - 10,000.	1. - 6.

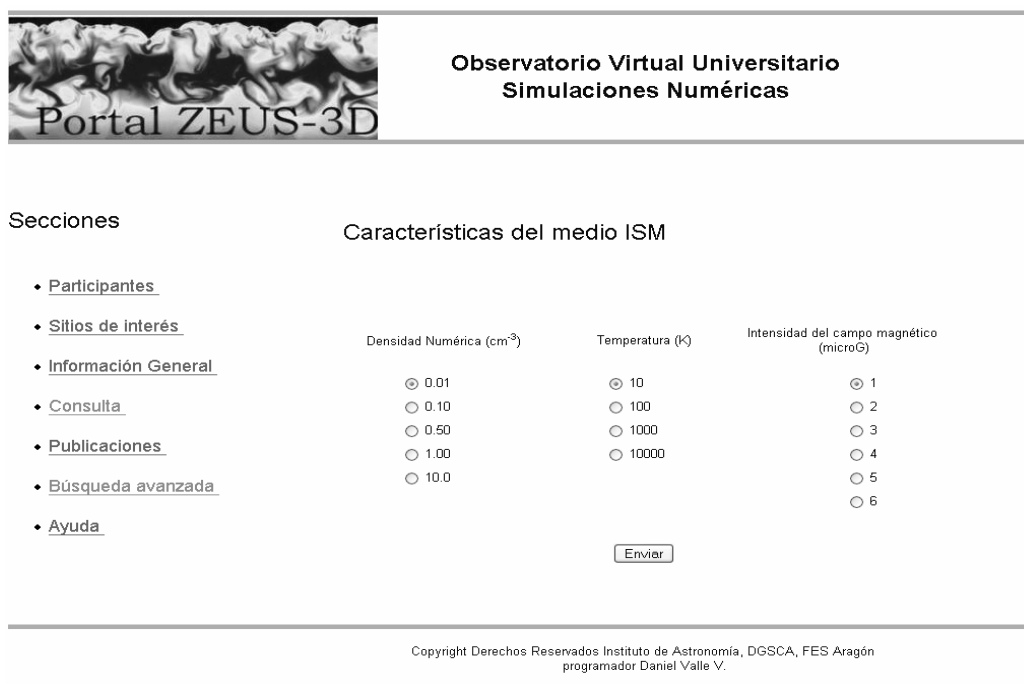
Tabla 4.2. Rango de valores del Medio Interestelar.

Una vez elegido el medio donde evolucionarán las nubes, el usuario debe de dar click en el botón *Enviar* para procesar la información. Enseguida cambiara la pantalla central mostrando con letras rojas el medio que se eligió y poniendo debajo de estas **la elección de la búsqueda que podrá realizarse, de acuerdo a las propiedades físicas iniciales**

<sup>1</sup> <http://www.hdfgroup.org/>

de las HVC's, podrá ser por cualquiera de sus 3 variables (radio, densidad y velocidad), o por 2 de ellas o por todas, tal y como se muestra en la figura 4.4.

Fig. 4.3



**Portal ZEUS-3D**

**Observatorio Virtual Universitario  
Simulaciones Numéricas**

Secciones

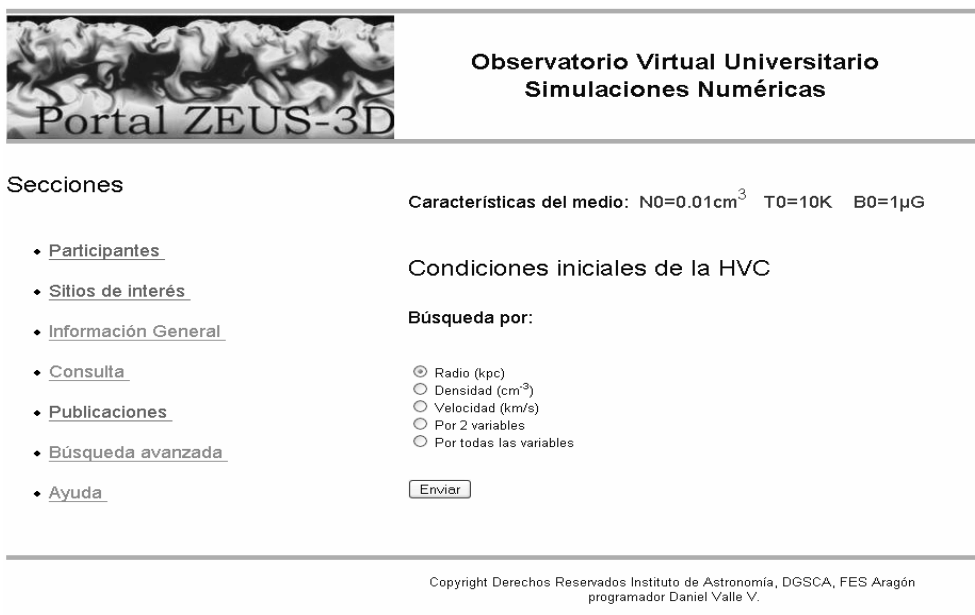
- [Participantes](#)
- [Sitios de interés](#)
- [Información General](#)
- [Consulta](#)
- [Publicaciones](#)
- [Búsqueda avanzada](#)
- [Ayuda](#)

**Características del medio ISM**

Densidad Numérica (cm <sup>3</sup> )	Temperatura (K)	Intensidad del campo magnético (microG)
<input checked="" type="radio"/> 0.01	<input checked="" type="radio"/> 10	<input checked="" type="radio"/> 1
<input type="radio"/> 0.10	<input type="radio"/> 100	<input type="radio"/> 2
<input type="radio"/> 0.50	<input type="radio"/> 1000	<input type="radio"/> 3
<input type="radio"/> 1.00	<input type="radio"/> 10000	<input type="radio"/> 4
<input type="radio"/> 10.0		<input type="radio"/> 5
		<input type="radio"/> 6

Copyright Derechos Reservados Instituto de Astronomía, DGSCA, FES Aragón  
programador Daniel Valle V.

Fig. 4.4



**Portal ZEUS-3D**

**Observatorio Virtual Universitario  
Simulaciones Numéricas**

Secciones

- [Participantes](#)
- [Sitios de interés](#)
- [Información General](#)
- [Consulta](#)
- [Publicaciones](#)
- [Búsqueda avanzada](#)
- [Ayuda](#)

**Características del medio: N0=0.01cm<sup>3</sup> T0=10K B0=1μG**

**Condiciones iniciales de la HVC**

**Búsqueda por:**

- Radio (kpc)
- Densidad (cm<sup>3</sup>)
- Velocidad (km/s)
- Por 2 variables
- Por todas las variables

Copyright Derechos Reservados Instituto de Astronomía, DGSCA, FES Aragón  
programador Daniel Valle V.

Figura 4.4 Muestra el medio interestelar que se ha elegido, y pone a disposición del usuario la elección de la forma en que se van a buscar las variables de la HVC.

Una vez que el usuario eligió como lo va a buscar, se pueden presentar 5 diferentes tipos de escenarios.

1. Que el usuario elija la opción por **Radio** y se presente una siguiente pantalla con la posibilidad de elegir el valor de la variable radio sin importar la velocidad, ni la densidad. Los valores del radio pueden ser de 0.05, 0.1 ó 0.15 kpc ( $1 \text{ kpc} = 3.0857 \times 10^{21} \text{ cm}$ ) y el usuario puede seleccionar desde uno hasta todos los valores que aparecen en pantalla.
2. Si selecciona el usuario la opción de **Densidad**, presentara en pantalla los valores que puede tener esta, que son 0.01, 0.1, 0.5 y 1 partículas/cm<sup>3</sup>. Puede ser elegido desde uno hasta todos los valores antes mencionados, usando esta opción no importara el valor que contenga la velocidad o el radio ya que solo será de importancia el medio interestelar que se selecciono y las constantes para la densidad.
3. Si se elige la opción de **Velocidad** se seguirá el mismo procedimiento que en los 2 casos anteriores, ya que no importaran los valores del radio y la densidad. El usuario puede elegir uno o más valores, dichos valores constan de 50, 100, 150, 200 y 300 km/s.
4. Si se opta por 2 variables, el usuario tendrá que elegir cualquiera de las combinaciones de 2 de 3 variables, ya sea por **radio y velocidad, densidad y velocidad o radio y densidad**. Una vez hecha la selección el usuario presionara el botón *Enviar*, que mostrará el contenido de los valores antes mencionados para las variables de la combinación seleccionada.
5. Que se elija por todas las variables, en donde se desplegaran los valores mencionados de las 3 variables.

Ya teniendo seleccionado la forma de búsqueda solo basta con presionar el botón *Consultar* para que enseguida se muestren los resultados de la búsqueda dando como ligas a la descarga del archivo **zhtoXXX.tar.tgz** y la película con extensión *mpg*.

Si los datos no existen, el usuario vera una pantalla que le dirá que no existe la simulación y que si desea realizar dicha simulación, deberá dar un click en una liga llamada *aquí*, la cual llevara directo a la pantalla de la simulación numérica remota.

#### 4.4 Consultas Avanzadas

El usuario puede dar click en la liga de búsqueda avanzada, permitiendo así poder elegir los datos sin tanto click ya que bastara con solo añadir la instrucción *where* e igualar las variables a la constante deseada para obtener el resultado que de igual manera mostrara las ligas a la descargas del *zhtoxxx.tar.tgz* y de la película, o en el caso de que no exista llevara a la pantalla de información de que no existe y pulsando una liga conducirá al camino para realizar la simulación deseada.

Finalmente, mostramos un ejemplo del tipo de las cosas que se pueden obtener a partir de graficar algunos de los ficheros contenidos en el archivo **zhtoXXX.tgz**. La graficación se realizó en *IDL*<sup>2</sup> (*Interactive Data Language*). Las condiciones iniciales del ISG y de la HVC se presentan en la tabla 4.3.

---

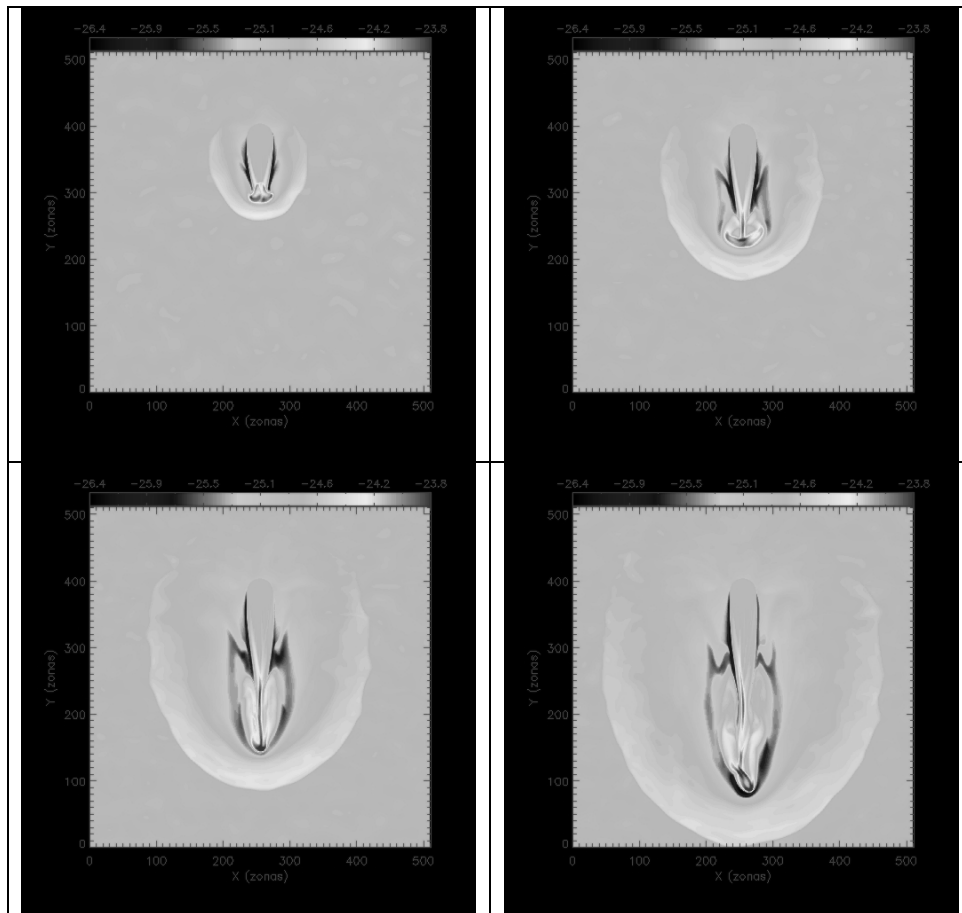
<sup>2</sup> <http://www.itvis.com/idl/>

	Medio Interestelar	HVC
Densidad (Part./cm <sup>3</sup> )	1.0	0.1
Temperatura (K)	100.	100.
Intensidad de B (10 <sup>-6</sup> Gauss)	1.	-
Velocidad (km/s)	0.	200.
Radio (kpc)	-	0.15

Tabla 4.3. Condiciones iniciales del Medio Interestelar y de una Nube de Alta Velocidad para la evolución que se muestran en la figura 4.5.

En la figura 4.5 se puede ver como una nube de alta velocidad interacciona con un medio que tiene un campo magnético aleatorio. En la evolución se producen una serie de efectos físicos que pueden ser de interés para el usuario que esta accediendo al Portal.

Fig.4.5



## CAPÍTULO V CONCLUSIONES Y TRABAJO A FUTURO

Al término del presente trabajo podemos decir que se han cumplido con los objetivos inicialmente planteados, toda vez que se ha implementado una solución completa y eficiente para el desarrollo del portal universitario de simulaciones numéricas.

Como se menciona la principal característica de este portal era el contar con una interfaz gráfica intuitiva que permitiera que usuarios de diferentes partes del mundo tuvieran acceso a esta herramienta.

Una de las tareas innovadoras de este trabajo es la de liberar tiempo de procesador para que la computadora que hospede el Portal no se sature y pueda realizar su función de forma eficiente. Esto se logró almacenando las simulaciones en una base a la que se le llamo **Catálogo de Simulaciones Numéricas**. Esto trajo como consecuencia proporcionarle una opción extra al usuario de tal manera que antes de hacer una simulación pueda llevar a cabo diferentes tipos de búsquedas desde las más simples a las más complicadas e incluso personalizadas, así como un completo acceso a toda la información almacenada de las simulaciones previamente hechas.

Como se puede ver en el capítulo IV, en este trabajo solo se utilizó el código hidrodinámico Zeus 3D para un problema astrofísico en particular que es el estudio de nubes de alta velocidad o flujos supersónicos de hidrógeno neutro, sin embargo otra de las aportaciones es que el diseño empleado en el mismo puede ser utilizado para ejecutar remotamente otro código numérico o incluso resolver otro tipo de problemas astrofísicos.

Como parte de los objetivos particulares estaban la investigación de diferentes aplicaciones computacionales, por ello durante el desarrollo de este trabajo se han aprendido a utilizar distintas herramientas para poder llevar a cabo la programación dinámica del portal, entre las cuales se pueden distinguir las bases de datos, herramientas de programación, y sobre todo la forma de poder combinar ambas de manera eficaz.

Del mismo modo se utilizaron diversas herramientas de apoyo como los diagramas de flujo y diferentes algoritmos para el diseño del sistema los cuales que fueron fundamentales al momento de encontrar salida a los diferentes problemas que se presentaron.

Se diseñó también un mecanismo de integridad de información, con el objetivo de evitar que al momento de ingresar las condiciones iniciales en el portal, se introdujeran valores no válidos o fuera de rango, con ello se consiguió que solamente se ejecuten en el servidor simulaciones que cumplieran con estas características.

En cuanto a la interfaz gráfica el sitio fue diseñado de tal forma que cumple con los estándares de programación W3C, además de utilizar hojas de estilo (archivos CSS) que facilitarán en el futuro el mantenimiento de este portal además de ser sencilla y fácil de usar.

## TRABAJO A FUTURO

Este portal tiene mucho trabajo que se puede realizar para mejorarlo, ya que hasta el momento solo contempla la posibilidad de llevar a cabo las simulaciones numéricas de un problema astrofísico en particular, pero lo deseable es que este se convierta en una herramienta útil para quiénes se dedican a estudiar este tipo de fenómenos astrofísicos.

Por lo tanto algunas de las características que se pueden agregar a este portal con la finalidad de mejorarlo sustancialmente son:

1. Agregar la posibilidad de que el usuario pueda ingresar su correo electrónico, y así poder programar que el sistema le envíe un correo cuando se termine de llevar a cabo la ejecución del proceso asociado a la simulación numérica, con una liga directa a una página *Web* en que pueda descargar los archivos generados por este proceso.
2. Implementar una función que permita al usuario que llevo a cabo la simulación numérica, generar a través del *Web* la animación o imágenes en base a los archivos de salida, ya que hasta el momento las animaciones se generan después de que se hizo la simulación numérica utilizando un software externo.
3. Diseñar un mecanismo interactivo que le permita al usuario bajar varios archivos al mismo tiempo asociados a diferentes simulaciones numéricas con alguna característica en común
4. Implementar un sistema para encolar trabajos de tal forma que cada vez que se ejecute una simulación esta pueda aprovechar al máximo todos los recursos computacionales.

Como se puede ver este trabajo solo es la base para el desarrollo de una herramienta mucho más completa.

## BIBLIOGRAFÍA

1. Date, C.J., “*Introducción a los sistemas de bases de datos*”, Traducc. Jaime Malpica y Américo Vargas, México, Sistemas Técnicos de Edición, 1986, p. 5-20.
2. Adoración de Miguel, Piattini, Mario, et. al., “*Diseño de bases de datos relacionales*”, México, Alfaomega, 2000, p 147-160.
3. Minera, Francisco, “*XML. La guía total del programador*”, Argentina, MP Ediciones, 2006, p.14-30.
4. www.LaLibreriaDigital.com, “*Programación de aplicaciones para Internet con ASP 3*”, España, Grupo Eidos, 2000, p. 11-43.
5. Cobo, Ángel, Gómez, Patricia, et. al, “*PHP y MySQL. Tecnologías para el desarrollo web*”, España, 2005, p. 504.
6. Sánchez-Salcedo, F. J.; Santillán, A.; Franco, J., 2007, “*Cloudy intergalactic accretion flows in the outer discs of galaxies*”, New Astronomy Reviews, Volume 51, Issue 1-2, p. 104-107.
7. Santillán, A.; Franco, J.; Kim, J., 2004, “*Interaction of High Velocity Clouds with Magnetized Disks: Three-Dimensional Numerical Simulations*” Journal of the Korean Astronomical Society, vol. 37, no. 4, pp. 233-235.
8. Santillán, A.; Franco, J.; Martos, M.; Kim, J., 1999, “*The Collisions of High-Velocity Clouds with a Magnetized Gaseous Galactic Disk*”, The Astrophysical Journal, Volume 515, Issue 2, pp. 657-668.
9. Stone, J. M., & Norman, M. L. 1992, “ZEUS-2D: A radiation magnetohydrodynamics code for astrophysical flows in two space dimensions. I - The hydrodynamic algorithms and tests”. Astrophysical Journal Supplement Series, 80, 753.
10. Wakker, B. P., et al., 2003, “*The Far Ultraviolet Spectroscopic Explorer Survey of O VI Absorption in and near the Galaxy*”, The Astrophysical Journal Supplement Series, Volume 146, Issue 1, pp. 1-123.
11. Leal, Héctor, “*Manual de HTML*”, México, Centro de Investigación Cibernética, 2005, p. 76.
12. Muller, C.A., Oort J.H. & Raimond, E., 1963, C.R. Acad. Sci. Paris. 257, 1661.
13. Date, C.J., “*Date on Database: Writtings 2000-2006*”, E.U.A., Apress, 2006, p. 319-327.
14. Powell, Gavin, “*Database Desing*”, E.U.A., Wiley Publishing Inc, 2006, p. 73-170.
15. Pavón, Jacobo, “*Creación de un Portal con PHP y MySQL*”, México, Alfaomega, 2005, p. 203.

16. Pérez Cesar, “*MySQL para Windows y Linux*”, México, Alfaomega, 2004, p. 454.
17. Costagnetto, Jesús, Harish, Rowat, et. al., “*Professional PHP Programming*”, EUA, Wrox Press Ltd, 2000, p. 85-164.
18. Benavides, J., Olaizola, J.M., et. al., “*SQL para usuarios y programadores*”, 2da. Edición, España, Parainfo, 1992, p. 17-153.
19. Ullman, Chris, dykes, Lucinda, “*Beginning Ajax*”, EUA, Wiley Publishing, 2007, p. 1-20.
20. Brown, Simon, Burdick, Robert, et. al., “*Professional JSP*”, 2da. Edición, EUA, Wrox Press, 2001, p. 1-30.
21. Worsley, Jhon C., Drake, Joshua D., “*Practical PosGreSQL*”, EUA, O’Reilly Media, 2002, p. 3-32.



## REFERENCIAS

1. [http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001\\_MAMoraga.pdf](http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_MAMoraga.pdf)
2. [http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001\\_Igarcia.pdf](http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_Igarcia.pdf)
3. <http://www.fismat.umich.mx/~elizalde/tesis/node15.html>
4. <http://alarcos.inf-cr.uclm.es/doc/bbddavanzadas/Funcionalidad%201.pdf>
5. <http://www.jorgesanchez.net/bd/docs2006/sghd.pdf>
6. <http://www.geocities.com/ymarte/trab/infalgprg.html>
7. [http://www.elprisma.com/apuntes/administracion\\_de\\_empresas/quesonlosdiagramasdeflujo/](http://www.elprisma.com/apuntes/administracion_de_empresas/quesonlosdiagramasdeflujo/)
8. <http://www.sistemas.unam.mx/software.html>
9. <http://es.wikipedia.org/wiki/Programaci%C3%B3n>
10. [http://platea.pntic.mec.es/jmas/manual/html/estilo\\_con\\_css.html](http://platea.pntic.mec.es/jmas/manual/html/estilo_con_css.html)
11. [http://es.wikipedia.org/wiki/Dise%C3%B1o\\_de\\_p%C3%A1ginas\\_web](http://es.wikipedia.org/wiki/Dise%C3%B1o_de_p%C3%A1ginas_web)
12. [http://www.inclusiondigital.net/albergue/dweb/leng\\_marcas.html](http://www.inclusiondigital.net/albergue/dweb/leng_marcas.html)
13. <http://www.osmosislatina.com/xml/basico.htm>
14. <http://www.aulambra.com/ver2.asp?id=20&tipo=>
15. <http://www.uca.edu.sv/investigacion/bdweb/tecnolog.html>
16. <http://es.tldp.org/Manuales-LuCAS/doc-curso-html/doc-curso-html/>
17. <http://www.wmlclub.com/articulos/jsp.htm>
18. <http://www.ajaxya.com.ar/temarios/descripcion.php?cod=8&punto=1>
19. [http://www.programacion.net/java/tutorial/servlets\\_jsp/12/](http://www.programacion.net/java/tutorial/servlets_jsp/12/)
20. <http://www.maestrosdelweb.com/editorial/phpintro/>
21. <http://www.gxopen.com/commwiki/servlet/hwiki?Proyecto+Mono>
22. <http://www.sobl.org/traduccion/practical-postgres/node12.html>
23. <http://www.microsoft.com/latam/sql/2005/productinfo/>
24. <http://www.programacionweb.net/articulos/articulo/?num=184>
25. <http://www.pecesama.net/php/bd.php?PHPSESSID=0c39861ebb72800da48edf636080933d>
26. [http://www.netpecos.org/docs/mysql\\_postgres/x108.html#AEN115](http://www.netpecos.org/docs/mysql_postgres/x108.html#AEN115)
27. <http://php.net/manual/en>

## APÉNDICE A

### *estilo.css*

```
body, td{
    font-size:10pt;
    font-family:arial;
}
```

### *index.php*

```
<?php
include("../libs/obvio.php");
inicioPag("Portal de simulaciones numéricas IA", "../libs/estilo.css");
?>
<table border="0" width="100%" cellpadding="0" cellspacing="0" >
  <tr> <!-- renglon de inicio-->
    <td height="150"> </td>
    <td height="150"> </td>
    <td height="150"> </td>
  </tr>
  <tr> <!-- renglon del borde superior de la cabecera-->
    <td height="5" > </td>
    <td height="5" bgcolor="orange"> </td>
    <td height="5"> </td>
  </tr>
  <tr > <!-- renglon de cabecera-->
    <td > </td>
    <td height="70">
      <center>
      </center>
    </td>
    <td > </td>
  </tr>
  <tr> <!-- renglon del borde inferior de la cabecera-->
    <td height="5" > </td>
    <td height="5" bgcolor="orange"> </td>
    <td height="5"> </td>
  </tr>
  <tr > <!-- renglon del centro-->
    <td width="10%">
      </td>
    <td width="80%" ><br /><br /> <!-- tabla para distribuir la imagen del centro-->
    <table border="0" width="100%" cellpadding="0" cellspacing="0">
      <tr >
        <td width="50%" >
          <br /><br /><br />
          <font size="+3" color="darkblue"><center>Realizaci&oacute;n de simulaciones
N&uacute;mericas<br /><br />
          </center> </font>
          <map name="boton1">
            <area shape="circ" coords="36,36,67" href="/simulacion.php" />
          </map>
          <br /><br /><br /> <br />
          <center>
           <br /><br />
        </td>
      </tr>
    </table>
  </td>
  <td width="50%" >
```

```

<br /><br /><br />
<font size="+3" color="darkblue"><center>Catalogo de simulaciones N&uacute;mericas<br />
  <br />
</center> </font>
  <map name="boton">
    <area shape="circ" coords="36,36,67" href="/astros.php" />
  </map>
<br /><br /><br /> <br />
<center>
 <br /><br />
</td>
</tr>
</table>
<br /><br />
</td>
<td width="10%">
</td>
</tr>
<tr> <!-- renglon del borde superior del pie-->
  <td height="5" > </td>
  <td height="5" bgcolor="orange"> </td>
  <td height="5"> </td>
</tr>
<tr>
  <td height="5">
  </td>
  <td > <br />
  <font color="darkblue" size="+1">
  <center>UNAM<br />
  "Por mi raza hablara el esp&iacute;ritu" </center>
  </font>
  </td>
  <td>
  </td>
</tr>
<tr> <!-- renglon del borde inferior del pie-->
  <td height="5" > </td>
  <td height="5" bgcolor="orange"> </td>
  <td height="5"> </td>
</tr>
</table>
<?php
finalPag();
?>

```

### ***astros.php***

```

<?php
include("../libs/obvio.php");
include("../clases/castros.php");
inicioPag("Bienvenido al portal de Zeus 3-D", "../libs/estilo.css");
Encabezado();
Izquierda();
if(empty($pagina)){
  Derecha();
}else{
Derecha($pagina);
}
Pie();
finalPag();

```

?&gt;

**ayuda.php**

```

<font size="+1" color="darkblue">
<center>
Bienvenidos a la ayuda del Catalogo de Simulaciones N&uacute;mericas<br />
<br />
Selecciona una opci&oacute;n
<br /> <br /><br />
<a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"];
?>?pagina=ayuda_a">Introducci&oacute;n <br /></a><br />
<a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"]; ?>?pagina=ayuda_b">Paginas de
informaci&oacute;n <br /></a><br />
<a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"]; ?>?pagina=ayuda_c">Consultas
B&aacute;sicas <br /></a><br />
<a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"]; ?>?pagina=ayuda_d">Consultas
Avanzadas <br /></a><br />
<a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"]; ?>?pagina=ayuda_e">Contacto <br
/></a><br /><br /><br />
</center>
</font>

```

**ayuda\_a.php**

```

<table width="100%" cellspacing="0" cellpadding="0">
<tr>
<td width="15%"> </td>
<td width="70%">
<center><font size="+2" color="darkblue">"Bienvenido al manual de ayuda del Observatorio Virtual
Universitario"</font><br /><br /> <br />
</center>
</td>
<td width="15%"></td>
</tr>
<tr>
<td></td>
<td>
<font size="+1" color="darkblue"><em>Introducci&oacute;n</em><br /><br />
</font>
<p>
<font color="darkblue">
En esta gu&iacute;a encontrara todo lo referente al funcionamiento de este portal de Internet.
Comenzaremos por explicarle lo que tiene usted en frente:

```

De su lado izquierdo encontrara el men&uacute; principal, que lleva 7 ligas en forma de lista, las cuales son participantes, sitios de inter&eacute;s, informaci&oacute;n general, consulta, publicaciones, b&uacute;squeda avanzada y ayuda. Cada una de estas ligas tiene una funci&oacute;n en especial, podr&aacute; acceder a dicha funci&oacute;n dando un click con el mouse en la liga que requiera, ya que se haya hecho lo anterior, el espacio donde se visualiza este texto cambiara, apareciendo lo referente a lo que se solicito, dando espacio a que consulte informaci&oacute;n, vea datos, asigne un c&oacute;digo en MySQL para realizar una b&uacute;squeda o de una serie de click para que se aparezca la b&uacute;squeda deseada, etc.

Se aconseja que lea la siguiente parte del manual para conocer mejor como funciona cada una de las ligas, este tutorial lo llevar paso por paso por cada una, a trav&eacute;s de los links que tiene abajo de este texto, que son "Atras", "Inicio" y "Siguiente".<br /><br /><br /><br />





decir solo podrá; elegir un dato para temperatura, uno para densidad y uno para la intensidad, esto lo hace situándose con el puntero del mouse en el lado izquierdo del número que requiera y dando click al botón radio, cuando haga esto aparecerá un punto verde que indica cual es la opción que eligió, y una vez elegido el medio, deberá presionar el botón que aparece abajo con el nombre de "Enviar".

<br /> <br />

Por ejemplo <br /><br />

Temperatura=10, Densidad=0.01, Intensidad= 1 <br />

<br />

Esto quiere decir que buscara todas las simulaciones que tengan el medio ya descrito como com.

Una vez apretado el botón, aparecerá en el centro un recordatorio del ISM que escogió, con las constantes en color rojo,

también aparecerá un menú para que pueda buscar las condiciones de la HVC. Esto tiene 5 opciones de búsqueda:

"Por Radio", "Por densidad", "Por velocidad", "Por 2 variables" y "Por todas las variables".

<br /><br />

**Por radio**

<br /><br />

Si la elección (esta se hace dando click en botón radio que se localiza en el lado izquierdo del texto que se quiere, y este se iluminará con

un color verde) es "Por radio" deberá de seleccionarse y posteriormente dar click en el botón que aparece abajo y tiene el nombre de

"Enviar", y cambiará una vez más la pantalla pasando ahora a que pueda seleccionar el valor numérico de la variable radio.<br />

Aquí aparecerá una serie de posibles constantes para esta variable, y se podrá elegir de 1 hasta todas las variables que

aparecen en pantalla (esto se hace dando click en la caja que tiene en el lado izquierdo el número, dicha caja tendrá una palomita de color

verde lo cual significa que ya fue seleccionada, para quitar la selección solo hay que dar un nuevo click en la caja y la palomita desaparecerá),

lo cual quiere decir que buscara todo lo registrado con el ISM elegido y con los valores o valor que se escogió para la variable radio,

sin importar cuales sean los valores de las variables de velocidad y densidad.

<br /><br />

**Por Densidad**

<br /><br />

Si se elige esta opción, pasará exactamente lo mismo que en el anterior, solo con la diferencia que se imprimirán en pantalla valores

de la variable densidad. Es decir, se podrá elegir de una hasta todas las constantes impresas en pantalla para la variable densidad, lo cual

implica, que se buscara todas las simulaciones que tengan el ISM que se escogió y que tengan los valores o el valor que se eligió para densidad,

sin importar los valores de velocidad y radio.

<br /><br />

**Por Velocidad**

<br /><br />

Pasa lo mismo solo que aquí se eligen constantes para la velocidad, y lo podrá hacer seleccionando uno o todos los posibles valores

de esta variable, lo cual implica que se buscarán simulaciones con el ISM elegido y en combinación con los valores de velocidad, sin

importar la densidad y el radio.

<br /><br />

**Por 2 variables**

<br /><br />

Esta es la opción en que puede elegir buscar una combinación de 2 variables, es decir "Por Densidad y Radio", "Densidad y Velocidad"







o si eres un usuario avanzado hazlo en B&uacute;squeda Avanzada.<br /> <br />  
 Cualquier duda consultar el tutorial en la liga con el nombre de Ayuda<br /><br /></font>  
 </center>

### **cons.php**

```
<?
//script para hacer la consulta click click click
include("./clases/ism.php"); //clase para elegir el medio ism
include("./clases/opciones.php");//clase para elegir la opcion cuando es por 2 variables
include("./clases/porN.php");//clase para desplegar el menu cuando se busca por n
include("./clases/porR.php");//clase para desplegar el menu cuando se busca por r
include("./clases/porV.php");//clase para desplegar el menu cuando se busca por v
include("./clases/por2.php");//clase para desplegar el menu cuando por 2
include("./clases/todas.php");//clase para desplegar el menu de todas
include("./clases/porNyR.php");//clase para desplegar el menu cuando se busca por r y n
include("./clases/porNyV.php");//clase para desplegar el menu cuando se busca por n y v
include("./clases/porVyR.php");//clase para desplegar el menu cuando se busca por v y r
$botpor2=$_POST['selec']; //valor del boton que se encuentra en el menu por 2
$op=$_POST['op'];//valor de la opcion que se elige en el menu por2
$n02=$_POST['n02'];//devolucion de la variable oculta n0 en el formulario por2
$t02=$_POST['t02'];//devolucion de la variable oculta t0 en el formulario por2
$b02=$_POST['b02'];//devolucion de la variable oculta b0 en el formulario por2
$opc=$_POST['opcion']; //valor de la opcion que se elige en el menu opciones
$n01=$_POST['n01'];//devolucion de la variable oculta n0 en el formulario opciones
$t01=$_POST['t01'];//devolucion de la variable oculta t0 en el formulario opciones
$b01=$_POST['b01'];//devolucion de la variable oculta b0 en el formulario opciones
$bot=$_POST['pasa'];//valor del boton que se encuentra en el menu opciones
$submit=$_POST['Enviar'];//valor del boton que se encuentra en el menu ism
$n0=$_POST['n0'];//devolucion de la variable oculta n0 en el formulario ism
$t0=$_POST['t0'];//devolucion de la variable oculta t0 en el formulario ism
$b0=$_POST['b0'];//devolucion de la variable oculta b0 en el formulario ism
if($botpor2=="Enviar"){ //caso por si es elegido el menu por2
  switch ($op) {
    case porNyR:
      porNyR($n02,$t02,$b02); //llamada a la funcion porNyR
      break;
    case porNyV:
      porNyV($n02,$t02,$b02); //llamada a la funcion porNyV
      break;
    case porVyR:
      porVyR($n02,$t02,$b02);//llamada a la funcion porVyR
      break;
  }
}
elseif($bot=="Enviar"){//caso por si es elegido el menu opciones
switch ($opc) {
  case porR:
    porR($n01,$t01,$b01);//llamada a la funcion porR
    break;
  case porV:
    porV($n01,$t01,$b01);//llamada a la funcion porV
    break;

  case porN:
    porN($n01,$t01,$b01);//llamada a la funcion porn
    break;
  case por2:
    por2($n01,$t01,$b01);//llamada al menu por2
    break;
}
```

```

case todas:
    todas($n0,$t0,$b0);//llamada a la funcion todas
break;
}
}
elseif($submit=="Enviar"){ //llamada a la funcion ism
    opcion($n0,$t0,$b0); //llamada al menu opciones
}
else{//si no son llamados ninguno de los menus se ejecuta la funcion ism
ism();//llamada al menu ism
}
?>

```

### ***Motor de búsqueda (result.php)***

```

<?php
//script para realizar la busqueda
include("../query.php");
$n0=$_POST['n0'];
$t0=$_POST['t0'];
$b0=$_POST['b0'];
$n=$_POST['n'];
$n1=$_POST['n1'];
$n2=$_POST['n2'];
$r=$_POST['r'];
$r1=$_POST['r1'];
$r2=$_POST['r2'];
$v=$_POST['v'];
$v1=$_POST['v1'];
$v2=$_POST['v2'];
$v3=$_POST['v3'];
$v4=$_POST['v4'];

$b=query($n0,$t0,$b0,$n,$n1,$n2,$n3,$r,$r1,$r2,$v,$v1,$v2,$v3,$v4);//asigna el contenido del archivo a
la variable b
$conn = mysql_connect("localhost", "root", "ec0l0g1a"); //hace conexion con mysql
mysql_select_db("zeus2");//se conecta con la base de datos zeus
$peticion=mysql_query($b);//hace la peticion de busqueda con la variable b
$existe = mysql_num_rows($peticion);//devuelve el numero de filas del resultado
if($existe==""){
    header("Location: ../astros.php?pagina=regresa");
}
if ($row = mysql_fetch_array($peticion)){//si existe manda en arreglos la busqueda
    echo "<table border = '1'> \n"; //imprime tabla con un borde de un pixel
    echo "<center> \n";
    echo "<tr> \n";
    mysql_field_seek($peticion,0); //devuelve el offset del resultado

    while ($field = mysql_fetch_field($peticion) ){//extrae la informacion de una columna y devuelve
como objeto
        echo "<td><b><center>$field->name</center></b></td> \n";
        echo "<tr> \n";
        do { //repetición hasta que termine con los resultados de la busqueda
            echo "<tr> \n";
            echo "<td><center>".$row['id']."</center></td> \n"; //imprime el resultado que es
nombrado como id
            echo "<td><center>".$row['n0']."</center></td> \n";//imprime el resultado de la busqueda
nombrado como n0

```

```

        echo "<td><center>".$row['t0']."</center></td> \n";//imprime el resultado de la busqueda
nombrado como b0
        echo "<td><center>".$row['b0']."</center></td> \n";//imprime el resultado de la busqueda
nombrado como t0
        echo "<td><center>".$row['n']."</center></td> \n";//imprime el resultado de la busqueda
nombrado como n
        echo "<td><center>".$row['r']."</center></td> \n";//imprime el resultado de la busqueda
nombrado como v
        echo "<td><center>".$row['v']."</center></td> \n";//imprime el resultado de la busqueda
nombrado como r
        echo "<td><center><a href='descargartar.php?id=$row[id]'>".$row['tar']."</center></a></td>
\n"; //imprime el resultado de la busqueda nombrado como tar en una liga
        echo "<td><center><a href='descargarm.php?id=$row[id]'>".$row['movie']."</center></a></td>
\n"; //imprime el resultado de la busqueda nombrado como movie en una liga
    } while ($row = mysql_fetch_array($peticion) ); //hasta que en la variable row ya no haya ningun
dato de matriz
    echo "</table> \n";
}
mysql_close($conn); //cierra la base de datos*/
?>

```

### ***simulacion.php***

```

<?php
include("./libs/obvio.php"); //funcion con contenido de html
include("./clase/emula.php");//funcion con el contenido del ingreso de variables
include("./clase/directorio.php");//funcion para crear directorio ***pendiente***
include("./clase/simbol.php");//funcion para evitar simbolos en el ingreso de datos
include("./clase/ceros.php");//funcion para evitar el ingreso de ceros de mas
include("./clase/archivo.php");//funcion para cambiar las sentencias del inzeus
include("./clase/inter.php");//funcion para llamar a xzeus34, hacer el tar y copiarlo al directorio de
descargas
include("./clase/bd.php");//funcion para ingresar datos a la BD
inicioPag("Bienvenido al portal de Zeus 3-D", "./libs/estilo.css");//llamado a la funcion de obvio
Encabezado();//llamado a la funcion de obvio
if($enviar)//si se da click en ejecutar
{
    if(empty($b0) or ctype_alpha($b0) or empty($t0) or ctype_alpha($t0) or
        empty($n0) or ctype_alpha($n0) or empty($r) or ctype_alpha($r) or
        empty($n) or ctype_alpha($n) or empty($v) or ctype_alpha($v)){//verifica que no esten vacios o
que no contegan caracteres el ingreso de variables
        $muestra="<br /><br />Una variable esta vacia o contiene caracteres<br /><br /><br /><br
/>";//despliega, si esta al menos uno vacio o con caracteres
    } else{//si los datos son numericos
        $checan0=simbolos($n0);//llama a la funcion simbolos del archivo simbol.php y verifica simbolos
en n0
        $checat0=simbolos($t0);//llama a la funcion simbolos del archivo simbol.php y verifica simbolos
en t0
        $checab0=simbolos($b0);//llama a la funcion simbolos del archivo simbol.php y verifica simbolos
en b0
        $checar=simbolos($r);//llama a la funcion simbolos del archivo simbol.php y verifica simbolos en
r
        $checan=simbolos($n);//llama a la funcion simbolos del archivo simbol.php y verifica simbolos
en n
        $checav=simbolos($v);//llama a la funcion simbolos del archivo simbol.php y verifica simbolos
en v
        if($checan0!="" or $checat0!="" or $checab0!="" or
            $checar!="" or $checan!="" or $checav!=""){ //si alguna de las variables estan vacias
            $muestra="error en simbolos";//se despliega este mensaje
        }
    }
}

```

```

else//si los datos no contienen simbolos
{
    $sincerosn0=ceros($n0);//llama a la funcion ceros del codigo ceros.php y elimina los ceros
    inecesarios de n0
    $sinceros0=ceros($t0);//llama a la funcion ceros del codigo ceros.php y elimina los ceros
    inecesarios de t0
    $sincerosb0=ceros($b0);//llama a la funcion ceros del codigo ceros.php y elimina los ceros
    inecesarios de b0
    $sincerosr=ceros($r);//llama a la funcion ceros del codigo ceros.php y elimina los ceros
    inecesarios de r
    $sincerosn=ceros($n);//llama a la funcion ceros del codigo ceros.php y elimina los ceros
    inecesarios de n
    $sincerosv=ceros($v);//llama a la funcion ceros del codigo ceros.php y elimina los ceros
    inecesarios de v
    if ($sincerosn0=="" or $sinceros0=="" or $sincerosb0=="" or
        $sincerosr=="" or $sincerosn=="" or $sincerosv=="") //si alguna de las variables esta vacia
    {
        $muestra="no puede llevar mas de 2 puntos";//despliega este mensaje
    }
    else//si son corregidos los ceros
    {

$matriz=$sincerosn0."_".$sinceros0."_".$sincerosb0."_".$sincerosn."_".$sincerosr."_".$sincerosv;//parte
del nombre del directorio, tar y mpg
    $nombre="RUN_".$matriz;//nombre completo del directorio
    $path="/".$nombre;//path donde se realizara la simulacion
    if (file_exists($path))//si el path existe
    {
        $muestra="Esta emulaci&oacute;n ya fue hecha, si desea los archivos de click <a
href=\"./astros.php\">aqu&iacute;</a>";//despliega este mensaje
    }
    else//de lo contrario
    {
        $crear=mkdir($path, 0777);//crea el directorio path con permisos 777
        if($crear==true){//si se creo
            inzeus($nombre,$sincerosn0,$sinceros0,$sincerosb0,$sincerosn,$sincerosr,$sincerosv);//se
llama a la funcion inzeus que esta dentro de archivo.php para que haga los ajustes en inzeus para la
simulacion
            system('cp xzeus34 ./.$nombre./xzeus34');//se copia xzeus34 al path con el mismo nombre
de xzeus34
            crea($matriz,$nombre);//se crea el archivo que sera ocupado como background
            base($matriz,$sincerosn0,$sinceros0,$sincerosb0,$sincerosn,$sincerosv,$sincerosr);
            exec("/usr/bin/php ./back.php > /dev/null 2>&1 &");//instruccion para hacer el background
            //pcntl_exec ("./.$nombre."xzeus34");
            $muestra="La emulaci&oacute;n ".$nombre." esta en proceso";//muesra este mensaje
        }
        else //si no fue creado el directorio
        {
            $muestra= "ocurrio un error";//imprime este mensaje
        }
    }
}

}

//parte complementaria para mostrar los mensajes una vez apretado el boton simular(abajo)
?>
<tr>
<td colspan="5" align="center">

```

```

<table cellpadding="0" cellspacing="0">
  <tr>
    <td colspan="3">
      <br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
    <?php
      echo "<font color='darkblue' size='+2'>$.muestra."</font>";
    ?>
    <br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
  </td>
</tr>
</table>
</td>
</tr>
<?php
Pie();
finalPag();
//fin de lo que sucede cuando se oprime simular
}else{//si no se aprita el boton
muestravariabales();//parte de la funcion emula
  Pie();//incluido en emula
  finalPag();//incluido en obvio
}
?>

```

### ***regresa.php***

```

<font color="darkblue" size="+1">
<b>La simulaci&oacute;n no se ha realizado aun </b><br /><br />
Si desea simularla hagalo pulsando <a href="/simulacion.php">aqu&iacute;e;</a> e ingrese sus datos
</font>

```

## APÉNDICE B

### *castros.php*

```

<?php
function Encabezado()
{
?>
<table width="100%" cellpadding="0" cellspacing="0" border="0"> <!--tabla principal-->
  <tr>
    <td bgcolor="orange" height="5" width="30%">
    </td>
    <td bgcolor="orange" height="5" width="40" >
    </td>
    <td bgcolor="orange" height="5" width="30%" >
    </td>
  </tr><!--termina sección de color de borde superior-->
  <tr >
    <td >
      <!--insercion de imagen-->
    </td>
    <td >
      <center><!--centrado del titulo-->
        <font size="+2" color="darkblue"> <b> <!--tamaño de fuente 12, color de la misma azul
marino, y en negritas-->
          Observatorio Virtual Universitario<br /> Simulaciones Num&eacute;ricas
        </b>
        </font>
      </center>
    </td>
    <td>
    </td>
  </tr>
  <tr> <!--seccion de color del borde de la cabecera inferior-->
    <td bgcolor="orange" height="5" >
    </td>
    <td bgcolor="orange" height="5" >
    </td>
    <td bgcolor="orange" height="5" >
    </td> <!--termina seccion de color del borde de la cabecera inferior-->
  </tr>
<?php
}
function Izquierda()
{
?>
  <tr >
    <td colspan="3"> <!--filas que ocupa esta column-->
      <table width="100%" cellpadding="0" cellspacing="0" ><!--tabla anidada-->
        <tr>
          <td ><br /><br /><font size="+2" color="darkblue"> Secciones</font><br /><br /><br />
          <ul>
            <font size="+1"><li> <!--seccion de ligas con tamaño de +1 del estandar, además de ser
ordenadas en forma de lista desordenada-->
              <a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"];
?>?pagina=participantes">Participantes <br /></a><br />
            </li>
            <li > <a href="<?php echo
$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=sitios">Sitios de inter&eacute;s <br /></a><br />
            </li>
          </ul>
        </tr>
      </table>
    </td>
  </tr>
}

```

```

        <li><a href="<?php echo $HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=info"
>Informaci&oacute;n General <br /> </a><br />
        </li>
        <li><a href="<?php echo
$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=cons">Consulta <br /> </a> <br />
        </li>
        <li><a href="<?php
echo$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=publicaciones"> Publicaciones <br /> <br />
        </a></li>
        <li><a href="<?php echo$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=consulta">
B&uacute;squeda avanzada <br /> <br />
        </a></li>
        <li><a href="<?php echo$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=ayuda">
Ayuda <br /></a></li><br /><br />
        </font>
    </ul>
</td><!--termina secci&oacute;n de ligas-->
<?php
}
function Derecha($pagina="inicio") //declaraci&oacute;n por default del contenido de la variable pagina=inicio--
>
{
    ?>
    <td >
        <?php
            $path="./paginas/".$pagina.".php";//definici&oacute;n del path de las paginas
            include($path);
        ?>
    </td>
</tr>
</table><!--fin de la tabla anidada-->
</td>
</tr>
<?php
}
function Pie()
{
    ?>
    <tr><!--seccion de color del borde superior de pie de p&eacute;gina-->
        <td bgcolor="orange" height="5"></td>
        <td bgcolor="orange" height="5" >
        </td>
        <td bgcolor="orange" height="5">
        </td>
    </tr><!--termina seccion de color del borde superior de pie de p&eacute;gina-->
    <tr>
        <td >
        </td>
        <td ><br /> <!--contenido del pie de p&eacute;gina con color azul marino en la fuente-->
        <center> <font color="darkblue">Copyright Derechos Reservados Instituto de Astronom&iacute;a,
DGSCA, FES Arag&oacute;n <br />
        programador Daniel Valle V.<br /> <br />
        </font> </center>
        </td>
    </tr>
    <tr><!--seccion de color del borde inferior de pie de p&eacute;gina-->
        <td bgcolor="orange" height="5"></td>
        <td bgcolor="orange" height="5" >
        </td>

```





```

<form method="post" action="<?php echo
$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=cons" > <br />
  <input type="radio" value="porR" name="opcion" checked> Radio (kpc)<br />
  <input type="radio" value="porN" name="opcion"> Densidad (cm<sup>-3</sup>) <br />
  <input type="radio" value="porV" name="opcion"> Velocidad (km/s) <br />
  <input type="radio" value="por2" name="opcion"> Por 2 variables <br />
  <input type="radio" value="todas" name="opcion"> Por todas las variables <br /><br /><br />
  <input type="hidden" value="<?echo $n0;?>" name="n01">
  <input type="hidden" value="<?echo $b0;?>" name="b01">
  <input type="hidden" value="<?echo $t0;?>" name="t01">
  <input type="submit" name="pasa" value="Enviar">
</form>
</td>
<td></td>
</table>
</font>
<?php
}
?>

```

### ***ism.php***

```

<?php
function ism(){
?>
<font size="+2" color="darkblue">  <!--color y tamaño de la fuente-->
<br /><br /><br />Características del medio ISM<br /><br /><br /><br />
</font>
<table >
  <tr border="1" bordercolor="darkblue"> <form method="post" action="<?php echo
$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=cons" ><!--declaración del formulario y a la página
que le enviara valores-->
    <td align="center" width="33%"> Densidad Numérica (cm<sup>-3</sup>) </td><!--
asignacion de valor a la variable n0-->
    <td align="center" width="33%">Temperatura (K) </td> <!--lista de valores para la variable t0-->
    <td align="center" width="33%"> Intensidad del campo magnético (microG)</td><!--
lista de valores para la variable b0-->
  </tr>
  <tr>
    <td ><br />
    <table border="0"><tr><td align="right" width="70%">
      <input type="radio" name="n0" value="0.01" checked></td><td align="left"> 0.01 <br />
    </td></tr></table></td>
    <td ><br />
    <table border="0"><tr><td align="right" width="75%">
      <input type="radio" name="t0" value="10" checked></td><td align="left"> 10 <br />
    </td></tr></table></td>
    <td ><br />
    <table border="0"><tr><td width="90%" align="right">
      <input type="radio" name="b0" value="1" checked></td><td align="left"> 1 <br />
    </td></tr></table></td>
  </tr>
  <tr>
    <td >
    <table><tr><td align="right" width="70%">
      <input type="radio" name="n0" value="0.1"></td><td align="left"> 0.10 <br />
    </td></tr></table></td>
    <td >
    <table border="0"><tr><td align="right" width="75%">
      <input type="radio" name="t0" value="100"> </td><td align="left">100 <br />
    </td></tr></table></td>
  </tr>

```

```

</td></tr></table></td>
<td>
<table border="0"><tr><td width="90%" align="right">
<input type="radio" name="b0" value="2"></td><td align="left"> 2 <br />
</tr></table></td></td>
</tr>
<tr>
<td>
<table border="0"><tr><td align="right" width="70%">
<input type="radio" name="n0" value="0.50"></td><td align="left"> 0.50 <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td align="right" width="75">
<input type="radio" name="t0" value="1000"></td><td align="left"> 1000 <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td width="90%" align="right">
<input type="radio" name="b0" value="3"></td><td align="left"> 3 <br />
</td></tr></table></td>
</tr>
<tr>
<td>
<table border="0"><tr><td align="right" width="70%">
<input type="radio" name="n0" value="1"></td><td align="left"> 1.00 <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td align="right" width="75">
<input type="radio" name="t0" value="10000"></td><td align="left"> 10000 <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td width="90%" align="right">
<input type="radio" name="b0" value="4"></td><td align="left"> 4 <br />
</td></tr></table></td>
</tr>
<tr>
<td>
<table border="0"><tr><td align="right" width="70%">
<input type="radio" name="n0" value="10"></td><td align="left"> 10.0 <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td align="right" width="75">
</td><td align="left"> <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td width="90%" align="right">
<input type="radio" name="b0" value="5"></td><td align="left"> 5 <br />
</td></tr></table></td>
</tr>
<tr>
<td>
<table border="0"><tr><td align="right" width="70%">
</td><td align="left"> <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td align="right" width="75">
</td><td align="left"> <br />
</td></tr></table></td>
<td>
<table border="0"><tr><td width="90%" align="right">
<input type="radio" name="b0" value="6"></td><td align="left"> 6 <br />

```

```

        </td></tr></table></td>
    </tr>
    <tr>
        <td></td>
        <td align="center"><br /><br /><input type="submit" name="Enviar" value="Enviar"><br /><br
/><br /><br /><br /></td><!-- boton para enviar resultados-->
        </form><!-- fin del formulario-->
    </td></td>
    </tr>
</table>
<?php
}
?>

```

### ***porN.php***

```

<?php
//funcion para consulta de n
function porN($n0,$t0,$b0)
{
?>
<center>
<font size="+1" color="darkblue">
    Se har&acute; una b&uacute;squeda con respecto a la Densidad (cm<sup>-3</sup>)
    <br /><br /><br />
    Inserte valor(es) de n (cm<sup>-3</sup>) <br /><br /><br />
    <form method="post" action="/paginas/result.php">
    <table>
        <tr>
            <td align="right"> 0.01 </td>
            <td> <input type="checkbox" name="n" value="0.01" checked /> <br></td>
            <td></td>
        </tr>
        <tr>
            <td align="right"> 0.10</td>
            <td> <input type="checkbox" name="n1" value="0.10" /><br></td>
            <td></td>
        </tr>
        <tr>
            <td align="right"> 0.50 </td>
            <td> <input type="checkbox" name="n2" value="0.50" /><br></td>
            <td></td>
        </tr>
        <tr>
            <td align="right"> 1.0 </td>
            <td> <input type="checkbox" name="n3" value="1" /><br> </td>
            <td> <input type="hidden" name="pag" value="busquedaN" /></td>
        </tr>
        <tr>
            <td><input type="hidden" value="<?echo $n0;?>" name="n0">
                <input type="hidden" value="<?echo $b0;?>" name="b0">
                <input type="hidden" value="<?echo $t0;?>" name="t0"></td>
            <td><br /><br />
            <input type="submit" value="Consultar" /></td>
            <td></td>
        </tr>
    </table>
    </form>
</font>
</center>

```

```
<?php
}
?>
```

### ***porR.php***

```
<?php
//funcion para consulta de r
function porR($n0,$t0,$b0)
{
?>
<center>
<font size="+1" color="darkblue">
  Se har&aacute; una b&uacute;squeda con respecto al par&aacute;metro del Radio (kpc)
  <br /><br /><br />
  Inserte valor(es) de r (kpc) <br /><br /><br />
  <form method="post" action="/paginas/result.php">
  <table>
    <tr>
      <td align="right">0.05</td>
      <td><input type="checkbox" name="r" value="0.05" checked /> <br /></td>
    </tr>
    <tr>
      <td align="right"> 0.1 </td>
      <td> <input type="checkbox" name="r1" value="0.1" /><br /></td>
    </tr>
    <tr>
      <td align="right"> 0.15 </td>
      <td> <input type="checkbox" name="r2" value="0.15" /><br /></td>
      <td> <input type="hidden" name="pag" value="busquedaR" /></td>
    </tr>
    <tr>
      <td><input type="hidden" value="<?echo $n0;?>" name="n0">
        <input type="hidden" value="<?echo $b0;?>" name="b0">
        <input type="hidden" value="<?echo $t0;?>" name="t0"></td>
      <td><br /><br /> <input type="submit" value="Consultar" /></td>
    </tr>
  </table>
  </form>
  </font>
  </center>
  <?php
  }
  ?>
```

### ***porV.php***

```
<?php
//funcion para consulta de v
function porV($n0,$t0,$b0)
{
?>
<center>
<font size="+1" color="darkblue">
  Se har&aacute; una b&uacute;squeda con respecto al par&aacute;metro de Velocidad (km/s)
  <br /><br /><br />
  Inserte valor(es) de v (km/s) <br /><br /><br />
  <form method="post" action="/paginas/result.php">
  <table>
```

```

<tr>
  <td align="right"> 50 </td>
  <td><input type="checkbox" name="v" value="50" checked /><br /></td>
</tr>
<tr>
  <td align="right"> 100 </td>
  <td><input type="checkbox" name="v1" value="100" /><br /></td>
</tr>
<tr>
  <td align="right"> 150 </td>
  <td><input type="checkbox" name="v2" value="150" /><br /></td>
</tr>
<tr>
  <td align="right"> 200 </td>
  <td><input type="checkbox" name="v3" value="200" /><br /></td>
</tr>
<tr>
  <td> 300 </td>
  <td><input type="checkbox" name="v4" value="300" /></td>
  <td> <input type="hidden" name="pag" value="busquedaV" /></td>
</tr>
<td><input type="hidden" value="<?echo $n0;?>" name="n0">
  <input type="hidden" value="<?echo $b0;?>" name="b0">
  <input type="hidden" value="<?echo $t0;?>" name="t0"></td>
<td><br /><br /><input type="submit" value="Consultar" /> </td>
</tr>
</table>
</form>
</font>
</center>
<?php
}
?>

```

### ***por2.php***

```

<?php
function por2($n0,$t0,$b0)
{
?>
<font color="darkblue" size="+1">
<br /><br />
Escoge una de las opciones <br /><br />
<table width="100%" cellpadding="0" cellspacing="0" >
  <tr>
    <td>
      <td>
    </td>
    <td>
      <form method="post" action="<?php echo
$HTTP_SERVER_VARS["PHP_SELF"];?>?pagina=cons" > <br />
        <input type="radio" value="porNyR" name="op" checked> Densidad (cm<sup>-3</sup>) y Radio
(kpc)<br />
        <input type="radio" value="porNyV" name="op"> Densidad (cm<sup>-3</sup>) y Velocidad
(km/s) <br />
        <input type="radio" value="porVyR" name="op"> Velocidad (km/s) y Radio (kpc) <br /><br />
        <input type="hidden" value="<?echo $n0;?>" name="n02">

```

```

        <input type="hidden" value="<?echo $b0;?>" name="b02">
        <input type="hidden" value="<?echo $t0;?>" name="t02">
        <input type="submit" name="selec" value="Enviar">
    </form>
</td>
<td></td>
</table>
</font>
<?php
}
?>

```

### ***todas.php***

```

<?php
//funcion para consulta de todas
function todas($n0,$t0,$b0)
{
?>
<center>
<font size="+1" color="darkblue"><br /><br /><br />
    Se har&aacute; una b&uacute;squeda con respecto a los par&aacute;metros Velocidad (km/s), Radio
(kpc) y Densidad (cm<sup>-3</sup>)
    <br /><br /><br />
    Seleccione valor(es) de v (km/s), r (kpc) y n (cm<sup>-3</sup>) <br /><br /><br />
    <form method="post" action="./paginas/result.php">
    <table>
        <tr>
            <td>Velocidad (km/s)<br /></td>
            <td></td>
            <td></td>
        </tr>
        <tr>
            <td></td>
            <td align="right"> 50 </td>
            <td><input type="checkbox" name="v" value="50" checked /> <br /></td>
        </tr>
        <tr>
            <td></td>
            <td align="right"> 100 </td>
            <td><input type="checkbox" name="v1" value="100" /><br /></td>
        </tr>
        <tr>
            <td></td>
            <td align="right" > 150 </td>
            <td><input type="checkbox" name="v2" value="150" /><br /></td>
        </tr>
        <tr>
            <td></td>
            <td align="right"> 200 </td>
            <td><input type="checkbox" name="v3" value="200" /><br /></td>
        </tr>
        <tr>
            <td></td>
            <td align="right" > 300 </td>
            <td><input type="checkbox" name="v4" value="300" /><br /></td>
        </tr>
        <tr>
            <td> <br /><br /> Radio (kpc) <br /></td>
            <td></td>

```

```

        <td></td>
    </tr>
    <tr>
        <td></td>
        <td align="right"> 0.05 </td>
        <td><input type="checkbox" name="r" value="0.05" checked /> <br /></td>
    </tr>
    <tr>
        <td></td>
        <td align="right"> 0.1 </td>
        <td><input type="checkbox" name="r1" value="0.1" /><br /></td>
    </tr>
    <tr>
        <td></td>
        <td align="right">0.15 </td>
        <td><input type="checkbox" name="r2" value="0.15" /><br /></td>
    </tr>
    <tr>
        <td> <br /><br /> Densidad (cm<sup>-3</sup>) <br /></td>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td></td>
        <td align="right"> 0.01 </td>
        <td><input type="checkbox" name="n" value="0.01" checked /> <br /></td>
    </tr>
    <tr>
        <td></td>
        <td align="right"> 0.10</td>
        <td><input type="checkbox" name="n1" value="0.10" /><br /></td>
    </tr>
    <tr>
        <td></td>
        <td align="right"> 0.50</td>
        <td><input type="checkbox" name="n2" value="0.50" /><br /></td>
    </tr>
    <tr>
        <td> </td>
        <td align="right"> 1.0</td>
        <td><input type="checkbox" name="n3" value="1" /><br /></td>
    </tr>
    <tr>
        <td><input type="hidden" value="<?echo $n0;?>" name="n0">
        <input type="hidden" value="<?echo $b0;?>" name="b0">
        <input type="hidden" value="<?echo $t0;?>" name="t0"></td>
        <td><br /><br /><input type="submit" value="Consultar" /> <br /><br /><br /><br /></td>
        <td><input type="hidden" name="pag" value="busquedaNVR" /></td>
    </tr>
</table>
</form>
</font>
</center>
<?php
}
?>

```



**porNyR.php**

```

<?php
//funcion para consultar datos de n y r
function porNyR($n0,$t0,$b0)
{
?>
<center>
<font size="+1" color="darkblue"><br /><br /><br />
  Se har&aacute; una b&uacute;squeda con respecto a los par&aacute;metros de Radio (kpc) y la
  Densidad (cm<sup>-3</sup>)
  <br /><br /><br />
  Seleccione valor(es) de r (kpc) y n (cm<sup>-3</sup>) <br /><br /><br />
  <form method="post" action="/paginas/result.php">
  <table>
    <tr>
      <td>Radio (kpc)<br /></td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td></td>
      <td align="right">0.05 </td>
      <td > <input type="checkbox" name="r" value="0.05" checked /> <br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right">0.1 </td>
      <td><input type="checkbox" name="r1" value="0.1" /><br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right">0.15 </td>
      <td><input type="checkbox" name="r2" value="0.15" /></td>
    </tr>
    <tr>
      <td><br /> Densidad (cm<sup>-3</sup>) <br /></td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 0.01 </td>
      <td> <input type="checkbox" name="n" value="0.01" checked /> <br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 0.10 </td>
      <td><input type="checkbox" name="n1" value="0.10" /><br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 0.50 </td>
      <td><input type="checkbox" name="n2" value="0.50" /><br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 1.0 </td>
      <td><input type="checkbox" name="n3" value="1" /><input type="hidden" name="pag"
value="busquedaNR" /></td>

```

```

</tr>
<tr>
  <td><input type="hidden" value="<?echo $n0;?>" name="n0">
  <input type="hidden" value="<?echo $b0;?>" name="b0">
  <input type="hidden" value="<?echo $t0;?>" name="t0"></td>
  <td><br /><br /><input type="submit" value="Consultar" /> <br /><br /><br /></td>
</tr>
</table>
</form>
</font>
</center>
<?php
}
?>

```

### *porNyV.php*

```

<?php
function porNyV($n0,$t0,$b0)
{
?>
<center>
<font size="+1" color="darkblue"><br /><br /><br />
  Se har&aacute; una b&uacute;squeda con respecto a los par&aacute;metros Densidad (cm<sup>3</sup>) y Velocidad (km/s)
  <br /><br /><br />
  Seleccione valor(es) en v (km/s) y n (cm<sup>-3</sup>) <br /><br /><br />
  <form method="post" action="./paginas/result.php">
  <table>
    <tr>
      <td>Velocidad (km/s)<br /></td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 50 </td>
      <td><input type="checkbox" name="v" value="50" checked /> <br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 100 </td>
      <td><input type="checkbox" name="v1" value="100" /><br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 150 </td>
      <td><input type="checkbox" name="v2" value="150" /><br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 200 </td>
      <td><input type="checkbox" name="v3" value="200" /><br /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right"> 300 </td>
      <td><input type="checkbox" name="v4" value="300" /></td>
    </tr>
  </table>

```

```

<tr>
  <td> <br /><br /> Densidad (cm<sup>-3</sup>) <br /></td>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td align="right"> 0.01 </td>
  <td><input type="checkbox" name="n" value="0.01" checked /> <br /></td>
</tr>
<tr>
  <td></td>
  <td align="right"> 0.10</td>
  <td><input type="checkbox" name="n1" value="0.10" /><br /></td>
</tr>
<tr>
  <td></td>
  <td align="right"> 0.50</td>
  <td><input type="checkbox" name="n2" value="0.50" /><br /></td>
</tr>
<tr>
  <td></td>
  <td align="right"> 1.0</td>
  <td><input type="checkbox" name="n3" value="1" /><br />
  <input type="hidden" name="pag" value="busquedaNV" /></td>
</tr>
<tr>
  <td><input type="hidden" value="<?echo $n0;?>" name="n0">
  <input type="hidden" value="<?echo $b0;?>" name="b0">
  <input type="hidden" value="<?echo $t0;?>" name="t0"></td>
  <td><br /><br /><input type="submit" value="Consultar" /> <br /><br /><br /><br /></td>
  <td></td>
</tr>
</table>
</form>
</font>
</center>
<?php
}
?>

```

### ***porVyR.php***

```

<?php
function porVyR($n0,$t0,$b0)
{
?>
<center>
<font size="+1" color="darkblue"><br /><br /><br />
  Se har&aacute; una b&uacute;squeda con respecto a los par&aacute;metros de Radio (kpc) y Velocidad
  (km/s)
  <br /><br /><br />
  Seleccione valor(es) de v (km/s) y r (kpc) <br /><br /><br />
  <form method="post" action="/paginas/result.php">
  <table>
  <tr>
  <td>Velocidad (km/s)<br /></td>
  <td></td>
  <td></td>

```

```

</tr>
<tr>
  <td></td>
  <td align="right"> 50 </td>
  <td><input type="checkbox" name="v" value="50" checked /> <br/></td>
</tr>
<tr>
  <td></td>
  <td align="right"> 100 </td>
  <td><input type="checkbox" name="v1" value="100" /><br /></td>
</tr>
<tr>
  <td></td>
  <td align="right" > 150 </td>
  <td><input type="checkbox" name="v2" value="150" /><br /></td>
</tr>
<tr>
  <td></td>
  <td align="right"> 200 </td>
  <td><input type="checkbox" name="v3" value="200" /><br /></td>
</tr>
<tr>
  <td></td>
  <td align="right" > 300 </td>
  <td><input type="checkbox" name="v4" value="300" /><br /></td>
</tr>
<tr>
  <td><br /><br />Radio (kpc)<br /></td>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td align="right"> 0.05 </td>
  <td><input type="checkbox" name="r" value="0.05" checked /> <br /></td>
</tr>
<tr>
  <td></td>
  <td align="right">0.1 </td>
  <td><input type="checkbox" name="r1" value="0.1" /><br /><input type="hidden" name="pag"
value="busquedaVR" /></td>
</tr>
<tr>
  <td></td>
  <td align="right">0.15 </td>
  <td><input type="checkbox" name="r2" value="0.15" /><input type="hidden" name="pag"
value="busquedaVR" /><br /></td>
</tr>
<tr>
  <td><input type="hidden" value="<?echo $n0;?>" name="n0">
  <input type="hidden" value="<?echo $b0;?>" name="b0">
  <input type="hidden" value="<?echo $t0;?>" name="t0"></td>
  <td><br /><br /><input type="submit" value="Consultar" /> <br /><br /><br /></td>
  <td></td>
</tr>
</table>
</form>
</font>
</center>
<?php

```

```
}
?>
```

### *query.php*

```
<?php
function query($n0,$t0,$b0,$n, $n1,$n2, $n3,$r, $r1, $r2,$v, $v1, $v2, $v3, $v4){
$inicio="SELECT id,n0,t0,b0,n,r,v,tar,movie FROM datos WHERE ";
$medio="c.id=d.id and n0='$n0' and t0='$t0' and b0='$b0' and "; $m=array($n, $n1,$n2, $n3); //inicia
matriz(m) para las variables de n
$matrizr=array($r, $r1, $r2); //inicia matriz(matrizr) para las variables de r
$matrizv=array($v, $v1, $v2, $v3, $v4); //inicia matriz (matrizv) para las variables v
$max=count($m); //cuenta los datos que se encuentran en m
$max_r=count($matrizr); //cuenta los datos que se encuentran en matrizr
$max_v=count($matrizv); //cuenta los datos que se encuentran en matrizv
$mr=array(); //matriz mr vacia
$mn=array(); //matriz mn vacia
$mv=array(); //matriz mv vacia
for ($i=0; $i<=$max; $i++) { //ciclo para ver cuantas variables existen en cada matriz
    if($m[$i]==""){ //en caso de que el arreglo i este vacio
        unset($m[$i]); //destruccion del arreglo i de la matriz m
    }else{ //si el arreglo no esta vacio
        $nmn=array_push($mn,$m[$i]); //introduccion de los arreglos encontrados de la matriz m en la matriz
mn
    }
}
for ($i=0; $i<=$max_r; $i++) { //ciclo para ver cuantas variables existen en la matriz matrizr
    if($matrizr[$i]==""){ //en caso de que el arreglo i este vacio
        unset($matrizr[$i]); //destruccion del arreglo i de la matriz matrizr
    }else{ //si el arreglo no esta vacio
        $nmr=array_push($mr,$matrizr[$i]); //introduccion de los arreglos encontrados de la matriz matrizr
en la matriz mr
    }
}
for ($i=0; $i<=$max_v; $i++) { //ciclo para ver cuantas variables existen en la matriz matrizv
    if($matrizv[$i]==""){ //en caso de que el arreglo i este vacio
        unset($matrizv[$i]); //destruccion del arreglo i de la matriz matrizv
    }else{ //si el arreglo no esta vacio
        $nmv=array_push($mv,$matrizv[$i]); //introduccion de los arreglos encontrados de la matriz matrizv
en la matriz mv
    }
}
$max_n1=count($mn); //cuenta los elementos de la matriz mn
$max_r1=count($mr); //cuenta los elementos de la matriz mr
$max_v1=count($mv); //cuenta los elementos de la matriz mv
$primera=array(); //declaracion de matriz vacia
$segunda=array(); //declaracion de matriz vacia
$tercera=array(); //declaracion de matriz vacia
$cuarta=array(); //declaracion de matriz vacia
if($max_v1===0){
    if($max_n1===0){ //en caso de que la matriz mn este vacia
        if($max_r1===0){ //si la matriz mr esta vacia
            $final= "El formulario esta vacio";
        }else if($max_r1===1){ //si la matriz mr solo tiene un arreglo
            $final=$inicio .$medio." r=".$mr[0];
        }
        else{ //si la matriz mr tiene mas de un arreglo
            for ($i=1; $i<=$max_r1; $i++) { //inicia un ciclo para vaciar los datos de la matriz mr a
otra
                array_push($primera,$medio." r=".$mr[$i-1]); // se agrega en la matriz primera la variable
medioconcatenada con r= y concatenada con el valor de la matriz mr en su arreglo i-1
            }
        }
    }
}
```

```

    }
    $ultima=array_shift($primera);//se saca el primer arreglo de la matriz primera
    $mas=count($primera);//se cuenta los arreglos de la matriz primera
    for ($i=1; $i<=$mas; $i++) { //inicia un ciclo para vaciar los datos de primera en segunda
        array_push($segunda,$primera[$i-1]." or "); //se agrega a la matriz segunda el dato de
primera en su arreglo i-1 concatenado con un or
    }
    array_push($segunda,$ultima);// se agrega el primer elemento que se sacó con la
variable ultima a la matriz segunda
    $salida=implode(" ",$segunda); //se juntan los arreglos de una matriz en una cadena,
separados por un espacio en blanco
    $final=$inicio.$salida; //se concatena la variable inicio con la variable salida
}
}
else{ //en caso de que matriz mn no este vacia
    if($max_r1===0){ //se pregunta si mr esta vacia
        if($max_n1===1){ //si matriz r esta vacia pregunta si m es igual a un dato en la matriz
            $final= $inicio .$medio." n=".$mn[0]; //si mn=1 dato se concatena las variables inicio,
medio y el dato de la matriz mn en su arreglo 0
        }else{
            for ($i=1; $i<=$max_n1; $i++) { //inicia un ciclo para vaciar los datos de la matriz mr a
otra
                array_push($primera,$medio." n=".$mn[$i-1]); // se agrega en la matriz primera la
variable medio concatenada con n= y concatenada con el valor de la matriz mn en su arreglo i-1
            }
            $ultima=array_shift($primera);//se saca el primer arreglo de la matriz primera
            $mas=count($primera);//se cuenta los arreglos de la matriz primera
            for ($i=1; $i<=$mas; $i++) { //inicia un ciclo para vaciar los datos de primera en
segunda
                array_push($segunda,$primera[$i-1]." or "); //se agrega a la matriz segunda el dato de
primera en su arreglo i-1 concatenado con un or
            }
            array_push($segunda,$ultima);// se agrega el primer elemento que se sacó con la
variable ultima a la matriz segunda
            $salida=implode(" ",$segunda); //se juntan los arreglos de una matriz en una cadena,
separados por un espacio en blanco
            $final=$inicio.$salida; //se concatena la variable inicio con la variable salida
        }
    }else{ //en caso de que mn y mr no sean vacia
//for anidados
        for ($i=1; $i<=$max_n1; $i++) { //ciclo para vaciar los elementos de mn en primera
            array_push($primera,$medio." n=".$mn[$i-1]." and "); //se vacia elementos de mn
concatenados con medio, con n= y con un and en la matriz primera
            for ($j=1; $j<=$max_r1; $j++) { //ciclo para vaciar los elementos de mr en segunda
                array_push($segunda,$primera[$i-1]." r=".$mr[$j-1]); //se vacia en la matriz segunda
los elementos de la matriz primera en su arreglo $i-1
            } //concatenados con una r= y esto concatenado con
los resultados de mr en su arreglo j-1
        }
        $ultima=array_shift($segunda); //se extrae el primer elemento de la matriz segunda
        $cont=count($segunda); //Se cuenta los elementos de la matriz segunda
        for ($i=1; $i<=$cont; $i++) { //inicia ciclo para extraer los elementos de la matriz
segunda
            array_push($tercera,$segunda[$i-1]." or "); //se extrae de la matriz segunda en su
arreglo i-1 para añadirlo a la matriz tercera
        }
        array_push($tercera,$ultima);//se añade la variable ultima a la matriz tercera
        $salida=implode(" ",$tercera); //se juntan los arreglos de una matriz en una cadena,
separados por un espacio en blanco
        $final=$inicio.$salida; //se concatena la variable inicio con la variable salida
    }
}
}
}

```

```

    }
  }
} else {
  if($max_n1===0 and $max_r1===0){ //si las matriz mr y mn estan vacias
    if($max_v1===1){//si matriz r esta vacia pregunta si m es igual a un dato en la matriz
      $final= $inicio .$medio." v=".$mv[0];//si mv=1 dato se concatena las variables inicio, medio y el
dato de la matriz mv en su arreglo 0
    } else {
      for ($i=1; $i<=$max_v1; $i++) { //inicia un ciclo para vaciar los datos de la matriz
mv a otra
        array_push($primera,$medio." v=".$mv[$i-1]);// se agrega en la matriz primera la
variable medio concatenada con v= y concatenada con el valor de la matriz mv en su arreglo i-1
      }
      $ultima=array_shift($primera);//se saca el primer arreglo de la matriz primera
      $mas=count($primera);//se cuenta los arreglos de la matriz primera
      for ($i=1; $i<=$mas; $i++) { //inicia un ciclo para vaciar los datos de primera en
segunda
        array_push($segunda,$primera[$i-1]." or ");//se agrega a la matriz segunda el dato
de primera en su arreglo i-1 concatenado con un or
      }
      array_push($segunda,$ultima);// se agrega el primer elemento que se sacó con la
variable ultima a la matriz segunda
      $salida=implode(" ",$segunda); //se juntan los arreglos de una matriz en una
cadena, separados por un espacio en blanco
      $final=$inicio.$salida; //se concatena la variable inicio con la variable salida
    }
  } else if($max_n1===0 and $max_r1!=0){//si el conteo de la matriz mn es diferente
de 0 y el conteo de mn es igual a 0
    for ($i=1; $i<=$max_v1; $i++) { //ciclo para vaciar los elementos de mv en
primera
      array_push($primera,$medio." v=".$mv[$i-1]." and ");//se vacia elementos de mn
concatenados con medio, con n= y con un and en la matriz primera
    }
    for ($j=1; $j<=$max_r1; $j++) { //ciclo para vaciar los elementos de mr en
segunda
      array_push($segunda,$primera[$i-1]." r=".$mr[$j-1]);//se vacia en la matriz
segunda los elementos de la matriz primera en su arreglo $i-1
    } //concatenados con una r= y esto concatenado con
los resultados de mr en su arreglo j-1
  }
  $ultima=array_shift($segunda); //se extrae el primer elemento de la matriz segunda
  $cont=count($segunda);//Se cuenta los elementos de la matriz segunda
  for ($i=1; $i<=$cont; $i++) { //inicia ciclo para extraer los elementos de la matriz
segunda
    array_push($tercera,$segunda[$i-1]." or ");//se extrae de la matriz segunda en su
arreglo i-1 para añadirlo a la matriz tercera
  }
  array_push($tercera,$ultima);//se añade la variable ultima a la matriz tercera
  $salida=implode(" ",$tercera); //se juntan los arreglos de una matriz en una cadena,
separados por un espacio en blanco
  $final=$inicio.$salida; //se concatena la variable inicio con la variable salida
} else if($max_n1!=0 and $max_r1===0){//si el conteo de la matriz mn es diferente
de 0 y el conteo de mn es igual a 0
  for ($i=1; $i<=$max_v1; $i++) { //ciclo para vaciar los elementos de mv en
primera
    array_push($primera,$medio." v=".$mv[$i-1]." and ");//se vacia elementos de mn
concatenados con medio, con n= y con un and en la matriz primera
  }
  for ($j=1; $j<=$max_n1; $j++) { //ciclo para vaciar los elementos de mn en
segunda

```

```

        array_push($segunda,$primera[$i-1]." n=".$mn[$j-1]);//se vacia en la matriz
segunda los elementos de la matriz primera en su arreglo $i-1
            } //concatenados con una n= y esto concatenado con
los resultados de mn en su arreglo j-1
        }
        $ultima=array_shift($segunda); //se extrae el primer elemento de la matriz segunda
        $cont=count($segunda);//Se cuenta los elementos de la matriz segunda
        for ($i=1; $i<=$cont; $i++) { //inicia ciclo para extraer los elementos de la matriz
segunda
            array_push($tercera,$segunda[$i-1]." or ");//se extrae de la matriz segunda en su
arreglo i-1 para añadirlo a la matriz tercera
            }
            array_push($tercera,$ultima);//se añade la variable ultima a la matriz tercera
            $salida=implode(" ",$tercera); //se juntan los arreglos de una matriz en una cadena,
separados por un espacio en blanco
            $final=$inicio.$salida; //se concatena la variable inicio con la variable salida
        } else{
            for ($i=1; $i<=$max_v1; $i++) { //ciclo para vaciar los elementos de mv en primera
                array_push($primera,$medio." v=".$mv[$i-1]." and ");//se vacia elementos de mn
concatenados con medio, con n= y con un and en la matriz primera
                for ($j=1; $j<=$max_n1; $j++) { //ciclo para vaciar los elementos de mn en
segunda
                    array_push($segunda,$primera[$i-1]." n=".$mn[$j-1]." and ");//se vacia en la
matriz segunda los elementos de la matriz primera en su arreglo $i-1
                        } //concatenados con una n= y esto
concatenado con los resultados de mn en su arreglo j-1
                    }
                    $c2=count($segunda);//Se cuenta la elementos de la matriz segunda
                    for ($i=1; $i<=$c2; $i++) { //ciclo para vaciar los elementos de segunda en
tercera
                        for ($j=1; $j<=$max_r1; $j++) { //ciclo para vaciar los elementos de mr en
tercera
                            array_push($tercera,$segunda[$i-1]." r=".$mr[$j-1]);//se vacia en la matriz
segunda, y se concatena con los elementos de la matriz mr en su arreglo j-1
                                }
                            }
                            $ultima=array_shift($tercera); //se extrae el primer elemento de la matriz tercera
                            $cont=count($tercera);//Se cuenta los elementos de la matriz tercera
                            for ($i=1; $i<=$cont; $i++) { //inicia ciclo para extraer los elementos de la matriz
tercera
                                array_push($cuarta,$tercera[$i-1]." or ");//se extrae de la matriz tercera en su arreglo
i-1 para añadirlo a la matriz cuarta
                                    }
                                    array_push($cuarta,$ultima);//se añade la variable ultima a la matriz tercera
                                    $salida=implode(" ",$cuarta); //se juntan los arreglos de una matriz en una cadena,
separados por un espacio en blanco
                                    $final=$inicio.$salida; //se concatena la variable inicio con la variable salida
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return $final;
}
?>

```



***descargatar.php***

```

<?
$conn = mysql_connect("localhost", "root", "");
mysql_select_db("zeus2");
$qry = "SELECT tar FROM archivo WHERE id=$id";
$res = mysql_query($qry);
$nombre = mysql_result($res, 0, "tar");
$path = "../descargas_t/".$nombre;
$fr=file_exists ($path);
if ($fr==false){
echo "<font color='red' size='+3'>El archivo no esta disponible </font>";
exit;
}
header("Content-type: application/x-zip-compressed");
Header("Content-Length: ".filesize($path));
header("Content-Disposition: attachment; filename=".$nombre);
readfile($path);
?>

```

***descargam.php***

```

<?
//script para descargar la pelicula
$conn = mysql_connect("localhost", "root", "");
mysql_select_db("zeus2");
$qry = "SELECT movie, id FROM archivo WHERE id=$id";
$res = mysql_query($qry);
$nombre = mysql_result($res, 0, "movie");
$path = "../descargas_m/".$nombre;
$fr=file_exists ($path);
if ($fr==false){
echo "<font color='red' size='+3'>El archivo no esta disponible </font>";
exit;
}
header("Content-type: video/mpeg");
header ("Content-Length: ".filesize($path));
header("Content-Disposition: attachment; filename=".$nombre);
readfile($path);
?>

```

***emula.php***

```

<?php
function Encabezado()
{
?>
<table width="100%" cellpadding="0" cellspacing="0" border="0"> <!--tabla principal-->
  <tr>
    <td bgcolor="orange" height="5" width="30%">
    </td>
    <td bgcolor="orange" height="5" width="40" >
    </td>
    <td bgcolor="orange" height="5" width="30%" >
    </td>
  </tr><!--termina sección de color de borde superior-->
  <tr >
    <td >
      <!--insercion de imagen-->
    </td>

```



```

    <br /><br /> <br /><br /> <br />
</font>
</font>
</td>
<td align="left">
<font color="darkblue">
<input type="text" name="n0" />&nbsp;&nbsp;&nbsp;(cm <sup>-3</sup>)
<br /><br /><br /><br /><br /><br />
</font>
</td>
<td align="right">
<font color="darkblue">
Radio:
<br /><br /><br /><br /><br />
</font>
</td>
<td align="left">
<font color="darkblue">
<input type="text" name="r" />&nbsp;&nbsp;&nbsp;(kpc)
</font>
<br /><br /><br /><br /><br />
</td>
</tr>
<tr>
<td align="right">
<font color="darkblue">
Temperatura:
<br /><br /><br /><br /><br />
</font>
</td>
<td align="left">
<font color="darkblue">
<input type="text" name="t0" />&nbsp;&nbsp;&nbsp;(K)
</font>
<br /><br /><br /><br /><br />
</td>
<td align="right">
<font color="darkblue">
Densidad:
<br /><br /><br /><br /><br />
</font>
</td>
<td align="left">
<font color="darkblue">
<input type="text" name="n" />&nbsp;&nbsp;&nbsp;(cm<sup>-3</sup>)
</font>
<br /><br /><br /><br /><br />
</td>
</tr>
<tr>
<td align="right">
<font color="darkblue">
Intensidad del campo m&acute;gnetico:
<br /><br /><br /><br /><br />
</font>
</td>
<td align="left">
<font color="darkblue">
<input type="text" name="b0" />&nbsp;&nbsp;&nbsp;(μG)
</font>

```



***simbol.php***

```

<?
function simbolos($cadenas){

$scorta=explode(".", $cadenas);
if($scorta[0]=="" and $scorta[1]=="")
{
return "esta vacia";
}
elseif($scorta[0]=="" and $scorta[1]!="")
{
if ctype_digit($scorta[1])
{
return;
}
else
{
return "Error; favor de revisar que no tenga simbolos a excepcion de un punto";
}
}
elseif($scorta[0]!="" and $scorta[1]=="")
{
if ctype_digit($scorta[0])
{
return;
}
else
{
return "Error; favor de revisar que no tenga simbolos a excepcion de un punto";
}
}
else
{
if ctype_digit($scorta[0]) and ctype_digit($scorta[1])
{
return;
}
else
{
return "Error; favor de revisar que no tenga simbolos a excepcion de un punto";
}
}
}
?>

```

***archivo.php***

```

<?
function inzeus($directorio,$n0,$t0,$b0,$n,$r,$v){
$fp=fopen("./".$directorio."/inzeus","a+");
$archivo = "inzeus";
$lineas = file($archivo);
$matriz=array();
for ($i=0; $i<=35; $i++) {
array_push($matriz, "$lineas[$i]");
}
$cadena1="      teta_hvc=270.0, dn_hvc=$n, v0_hvc=$v., r_hvc=$r,\n";
$cadena2="      dn0=$n0, b10=$b0.e-6 , b20=$b0.e-6, t0 = $t0.      $\n";
array_push($matriz,$cadena1);

```

```

array_push($matriz,$cadena2);
$cuente=count($matriz);
for ($k=0; $k<=$cuente; $k++) {
    fwrite($fp, "$matriz[$k]");
}
}
?>

```

### ***inter..php***

```

<?
function crea($nombre,$directorio){
    $archivo="./back.php";
    $fp=fopen($archivo,"w+");
    $contenido="<?php
    chdir('./".$directorio."");
    exec('./xzeus34');
    exec('tar -zcvf zhto."$nombre.".tgz zhto*');
    exec('cp zhto."$nombre.".tgz ../descargas_t');
    exec('rm zhto0* -f');
    ?>";
    fwrite($fp,$contenido);
    fclose($fp);
}
?>

```

### ***bd.php***

```

<?
function base($nombre,$n0,$t0,$b0,$n,$v,$r){
    $conn = mysql_connect("localhost", "root", "ec0l0g1a");
    mysql_select_db("auxiliar2",$conn);
    $peticion="INSERT INTO simulacion (n0,t0,b0,n,v,r) VALUES ('$n0','$t0','$b0','$n',
    '$v','$r)";
    $peticion2="INSERT INTO archivo (tar,movie) VALUES ('zhto.$nombre.tgz','movie.$nombre.mpg)";
    mysql_query($peticion);
    mysql_query($peticion2);
}
?>

```

## GLOSARIO

*.NET*: Proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones.

*.NET framework*: El "*framework*" o marco de trabajo, constituye la base de la plataforma .NET y denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

*Access*: Gestor de bases de datos, realizado por Microsoft.

*ActiveX*: Aplicación desarrollada por Microsoft que ofrece dinamismo en páginas de Internet.

*ADO (ActiveX Data Objects)*: Mecanismo desarrollado por Microsoft que usa los programas de computadoras para comunicarse con las bases de datos, darles órdenes y obtener resultados de ellas.

*Algoritmo*: Conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

*Ambigüedades*: Diferentes formas de interpretación.

*AND*: Instrucción para especificar un "y" en un lenguaje de programación o una sentencia de búsqueda en un gestor de base de datos.

*Apache*: Servidor de páginas Web desarrollado por la *Apache Software Foundation*.

*Aplicación*: Programa que resuelve un proceso determinado para un cliente final.

*Apuntador*: Un apuntador es un objeto que apunta a otro objeto. Es decir, una variable cuyo valor es la dirección de memoria de otra variable.

*Arreglo*: Estructura de datos en la que a cada elemento se le asigna un índice único.

*Atributo*: Es cada una de las cualidades, propiedades o características de un elemento.

*Backends*: Modo administrativo protegido por nombre de usuario y contraseña.

*Background*: Toda tarea o trabajo que se realiza en segundo plano o con prioridad baja.

*BASIC (Beginners All-purpose Symbolic Instruction Code)*: Lenguaje de programación inicialmente diseñado para principiantes.

*Benchmarks*: Técnica utilizada para medir el rendimiento de un sistema.

*Boceto*: Dibujo hecho de forma esquemática y sin preocuparse de los detalles o terminaciones para representar una idea.

Bucle: Sentencia que se realiza repetidas veces.

*Bugs*: Errores.

Burbuja: Sinónimo de proceso.

Burbujas de generación espontánea: Procesos que surgen de una manera no intencional.

C: Un lenguaje de programación transportable que se puede usar para desarrollar software.

C++: Más que C. Es un lenguaje C avanzado, basado en objetos.

Campo: Es la mínima unidad de información a la que se puede acceder.

CGI (Common Gateway Interface o Interface de Acceso Común): Programas usados para hacer llamadas a rutinas o controlar otros programas o bases de datos desde una página *web*. También pueden generar directamente *html*.

CGI-BIN: Directorio de un servidor *web* en donde suelen almacenarse los programas *CGI* y se destaca que "bin" es una contracción de "binario".

Cibernauta: Usuario del Internet.

Cluster: Conjunto de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora.

Compilar: Proceso por el cual se "traduce" un programa escrito en un lenguaje de programación a lo que realmente entiende el ordenador.

Cluster de conmutación por error: Mover los recursos de una aplicación y reiniciarlos en otro nodo del clúster de servidores.

Cluster de servidores: Grupo de sistemas independientes, conocidos como nodos, que ejecutan *SQL SERVER* y trabajan conjuntamente como un sistema único para asegurar que los recursos y las aplicaciones de importancia decisiva permanecen disponibles para los clientes.

Concatenar: Juntar 2 cosas en una cadena, ya sean 2 variables o una variable con un texto.

Contenedor de datos: Algo donde se guarda información.

Cookie: Cualquier cadena de texto que se instala en el disco duro de la computadora del usuario y cada vez que este entra al Portal envía información que se utilizara para identificar al usuario.

Dato: Es la unidad o cantidad mínima de información

*DB2*: Sistema de base de datos creado por IBM.



*Default*: Valor que se le da al inicio a alguna variable.

Dependencia transitiva: Es aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave.

Depurador: Programa especializado en la corrección y eliminación de errores en un código fuente durante su edición, o capturando diversos datos durante su ejecución.

Digital: Información representable a través de ceros y unos.

Dinámica: Atributo que se refiere a la manera en que los elementos de conexión son establecidos y liberados en el tiempo.

Directorio: Grupo de archivos relacionados entre sí que se guardan bajo un nombre.

Directorio Raíz: Lugar donde se almacenan directorios y archivos relacionados entre si.

*Do.. While*: Sentencia de programación que especifica que se va a realizar ciertas acciones (*do*), mientras se cumple una condición (*while*).

*Document Object Model*: Forma de representar documentos estructurados (tales como una página web *html* o un documento *XML*) que es independiente de cualquier lenguaje orientado a objetos.

DTD: Definición del tipo de documento.

*E-Commerce*: Comercio entre empresas y clientes que se realiza a través de Internet.

ECMA (*European Computer Manufacturers Association*): Es una asociación fundada en 1961 dedicada a la estandarización de sistemas de información.

EIFFEL: Lenguaje de programación orientado a objetos.

Entidad: Todo aquello que se exhibe.

Estándar: Unidad de medida adoptada y aceptada comúnmente como criterio.

Estructura de árbol: Forma de representación de datos que emula a un árbol.

Explorer: Es el navegador de Microsoft.

*Fedora Core*: Distribución Linux que sustituye a las versiones de *Red Hat Linux*.

Fichero: Archivo que contiene un conjunto de registros de datos.

*Firebird*: Sistema de administración de base de datos de código abierto, basada en la versión 6 de *Interbase*.

*Firefox*: Navegador *web* del proyecto *Mozilla*.

*For*: Sentencia de un lenguaje de programación en la que ejecutara una acción tantas veces como una variable iniciada en una cantidad alcanza un valor especificado.

*Frame*: Marco utilizado en un sitio *web*.

*FROM*: Instrucción de SQL para especificar de que tabla de la base de datos se sacara información

Gestor: Programa que se encarga de una tarea específica.

*GNU*: *Software* desarrollado para distribución sin fines de lucro.

*GNU autoconf*: Herramienta que permite producir “*shell scripts*” que automáticamente configuren nuestro programa con el fin de que se adapte correctamente a cualquier sistema.

Grafo: Organización de un conjunto de datos en la cual cada elemento puede tener uno o varios predecesores y uno o varios sucesores.

*Group By*: Instrucción para que muestre un sistema de base de datos grupos de información.

Hardware: Dispositivos físicos que comprenden un sistema de computación.

Hipertexto: Tipo de documento que tiene enlaces con otros.

Hipervínculo: Es un enlace o liga cuyo propósito es proporcionar el medio para acceder a un recurso en Internet.

*HVC (Hight –Velocity Clouds)*: Nubes de alta Velocidad.

*If*: Sentencia de un lenguaje de programación que hace referencia a una condicionante, y hace una tarea si (*if*) se cumple la condición.

*If..Else*: Sentencia de programación que hace referencia a que se haga una tarea si se cumple una condicionante, en caso de no hacerlo (*else*) ejecutara otra labor.

Información: Agrupación de datos con el objetivo de lograr un significado específico más allá de cada uno de éstos.

*INFOMIX*: Gestor de base de datos creado por *Informix Software Inc.*

*INTERBASE*: Sistema de Administración de Base de Datos Relacionales desarrollada y comercializada por la compañía Borland Software Corporation

Interfaz: Parte de un programa informático que permite a éste comunicarse con el usuario o con otras aplicaciones permitiendo el flujo de información.

*Inzeus*: Archivo que contiene las especificaciones en la ejecución de una simulación numérica hecha con el código hidrodinámico *ZEUS 3d*.

*ISM (International Solidarity Movement)* o Medio Interestelar: Es el nombre que los astrónomos dan al polvo y gas que impregnan al espacio interestelar.

*Java*: Lenguaje de programación creado por *Sun Microsystem* que permite crear pequeñas aplicaciones en diferentes tipos de ordenadores y sistemas operativos.

*Javascript*: Lenguaje interpretado con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

*JDBC (Java Data Base Connectivity)*: Aplicación de la interfaz de un programa utilizado para conectar programas escritos en *Java* a los datos de las bases de datos comunes.

*Joins*: Operaciones en la base de datos relacionando 2 o más columnas.

*JPG (Joint Photographic Experts Group)*: Formato de compresión de imágenes, tanto en color como en escala de grises, con alta calidad (a todo color).

*Kernel*: Parte esencial del sistema operativo, responsable de la asignación de recursos, las interfaces de hardware de bajo nivel, la seguridad, etc.

Lenguaje de programación: Código utilizado para la creación de programas.

Lenguaje de tratamiento de textos: Código utilizado para dar formato a texto.

Librería: Conjunto de procedimientos y funciones (subprogramas) agrupadas en un archivo con el fin de ser aprovechadas por otros programas.

*LINUX*: Sistema Operativo de distribución libre nacido a partir del *UNIX*.

Llave: Signo o clave que va al comienzo para describir una columna

*Localhost*: Se refiere a la localización del sistema utilizado actualmente.

Matriz: Conjunto de elementos que suelen ser números ordenados en filas y columnas.

Método orientado a objeto: Forma de proceder una entidad combinando su estado y comportamiento.

Modo Consola: Forma de utilizar un sistema operativo sin una interfaz gráfica.

MPG: Formato de codificación de audio y video .

Mono: proyecto de código abierto impulsado por Novell para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET.

*mSQL*: Motor ligero de base de datos que fue diseñado para un acceso rápido a datos almacenados.

Multi-Hilo: Múltiples tareas ejecutándose en un programa.

*MyISAM*: Tecnología de almacenamiento de datos usada por defecto por el sistema administrador de bases de datos relacionales *MySQL*.

Navegadores o *browsers*: Son una aplicación o software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en *html*, desde servidores *web* de todo el mundo a través de Internet.

*Netscape*: Un browser para páginas *web* y el nombre de un compañía.

Nodo: Punto de unión donde convergen 2 vertientes.

*ODBC*: *Open DataBase Connectivity*. Interfaz normalizada, o intermedia, para acceder a una base de datos desde un programa.

*Offset*: Sistema de impresión indirecto.

*OPENSOURCE*: Cualidad de algunos softwares de incluir el código fuente en la distribución del programa.

*OPERA*: Navegador de Internet creado por la empresa noruega Opera Software en 1994.

*OR*: Instrucción para especificar un “o” en un lenguaje de programación o una sentencia de búsqueda en un gestor de base de datos.

*ORACLE*: Sistema de administración de base de datos (o RDBMS por el acrónimo en inglés de *Relational Data Base Management System*), fabricado por *Oracle Corporation*.

Ordenador: Sinónimo de computadora.

*Order By*: Instrucción para que muestre información en algún orden un sistema de base de datos.

Páginas Activas: Son las utilizadas para mantener bases de datos, crear buscadores dinámicos, hacer carritos de compras, y todo aquello que necesite una interacción del navegante y el servidor para elaborar un resultado.

Paradigma orientado a objetos: Supuestos teóricos de datos que combinan estado y comportamiento.

*PERL*: Lenguaje de programación muy utilizado para la elaboración de aplicaciones *CGI*.

Portal: Conjunto de páginas de Internet reunidas bajo una marca, dirección, tema, asunto o interés.

Proceso: Conjunto de instrucciones entregadas a la CPU para el cumplimiento de una etapa específica señalada por los comandos de algún programa.

**Programa:** Instrucciones de computación estructuradas y ordenadas que al ejecutarse hacen que una computadora realice una función particular.

**Protocolo:** Conjunto de reglas que deben ser respetadas para que pueda ser realizado un proceso de comunicaciones.

**Protocolo HTTP:** Conjunto de reglas que permite el intercambio de información hipertextual en páginas *web*.

**PYTHON:** Lenguaje de programación interpretado e interactivo, capaz de ejecutarse en una gran cantidad de plataformas.

**Query:** Conjunto de condiciones o preguntas usadas para extraer información de la base de datos.

**Record Type:** Conjunto de registros con los mismos campos.

**Redundancia de datos:** Información que es almacenada varias veces en la misma base de datos.

**Registro:** Unidad de almacenamiento destinada a contener cierto tipo de datos.

**Renderizar:** Obtener algo con la máxima calidad posible.

**Repositorio:** Sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

**Rollback:** Operación que vuelve a su estado anterior a la base de datos.

**Rutina:** Subprograma que utiliza el programa principal sólo cuando lo considera necesario para realizar una tarea específica.

**Script:** Secuencia de código sin compilar que puede ejecutar acciones mediante un software que la interpreta.

**Segmento:** Parte que se exhibe de las bases de datos.

**Select:** Instrucción para seleccionar algún campo de una base de datos.

**Servidor web:** Es un programa que implementa el *protocolo HTTP*.

**Servlet:** Un programa Java del lado del servidor que ofrece funciones suplementarias al servidor.

**Sesión:** Cada una de las visitas que realiza un navegante al sitio *web*.

**Shell.** Es el intérprete de comandos que se establece entre el usuario y el *kernel*.

**Shell script:** Programa que contiene un sistema de comandos ejecutables.

**Sintaxis:** Reglas que gobiernan la estructura de un lenguaje.

**Sistema:** Conjunto de elementos interrelacionados que trabajan juntos para obtener un resultado deseado.

**Sistema de Distribución:** Sistemas cuyos componentes hardware y software, que están en ordenadores conectados en red, se comunican y coordinan sus acciones mediante el paso de mensajes, para el logro de un objetivo. Se establece la comunicación mediante un protocolo prefijado por un esquema cliente-servidor.

**Socket:** Objeto de software utilizado por un cliente para conectarse a un servidor.

**Software:** Conjunto de instrucciones o programas usados por una computadora para hacer una determinada tarea.

**SQL (Structured Query Language):** Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

**SQLite:** Pequeña librería programada en lenguaje C que implementa un completo motor de base de datos que no precisa configuración.

**Switch:** Sentencia de un lenguaje de programación que evalúa una variable para ejecutar cierta acción.

**Sybase:** Es un motor de bases de datos altamente optimizado para inteligencia empresarial, desarrollado por la empresa *Sybase*.

**Tar.gz:** Formato para la compresión de archivos del sistema operativo *UNIX*.

**TCL:** Lenguaje de *script* creado por John Ousterhout.

**TCP:** protocolo que se encarga de la transferencia de los paquetes a través de Internet.

**TOMCAT:** Contenedor de *servlets* desarrollado bajo el proyecto *Jakarta* en la Apache Software Foundation.

**UNIX:** Sistema operativo especializado en capacidades de multiusuario y multitarea el cual fue la base inicial del Internet.

**URL:** En inglés quiere decir "*Uniform Resource Locator*," y se refiere a la dirección única que identifica a una página *web* en Internet.

**Variables de ambiente:** Son las utilizadas para personalizar el manejo de los programas y la comunicación entre ellos.

**Visual Basic Script:** Lenguaje de *scripts* de Microsoft basado en el lenguaje de programación *Visual Basic*.

VBA (*Visual Basic for application* o Visual Basic para Aplicaciones): Lenguaje basado en VB y adaptado para aplicarse en las herramientas de *Office*, dando un mayor control al usuario.

*Web*: Conjunto de páginas de Internet reunidas bajo un mismo tema.

*Webmaster*: Persona responsable del mantenimiento de un servidor *web*.

*Web semántica*: Espacio donde la información tendría un significado bien definido, de manera que pudiera ser interpretada tanto por agentes humanos como por agentes computerizados.

*Where*: Instrucción para especificar un atributo de la información que se extraerá de una base de datos.

*While*: Instrucción de lenguaje de programación que significa que va a hacer determinada tarea mientras se cumple una condición.

*XMLHttpRequest*: Conjunto de herramientas, rutinas y protocolos que pueden ser invocados desde *javascript*.