# *1*

# *Lógica de Negocio*

## 1.1 Análisis y diseño de sistema OLTP (OnLine Transaction Processing)

Una vez que se ha llevado a cabo el levantamiento de los requerimientos y se han estudiado detalladamente (resolviendo con el cliente las dudas resultantes) es importante tener en cuenta que gran parte del tiempo invertido durante el desarrollo se utilizará en la etapa del diseño, pues es en esta fase donde se hacen los bosquejos teóricos que serán implementados en etapas posteriores.

Estos bocetos incluyen:

- El diseño de la base de datos que recogerá la información del sistema para ser procesada por las diferentes funcionalidades de la aplicación. Este se muestra en el Diagrama Entidad/Relación (Figura 1.1) que detalla la descripción de las tablas.

- En el Diagrama Jerárquico Funcional (Figura 1.2) se señalan, de manera general, los componentes funcionales en los que se divide la aplicación. Partiendo del diagrama anterior se tienen las tareas relacionadas con las tablas volátiles, las no volátiles y los diferentes tipos de reportes requeridos.

- La navegabilidad del sistema, la interacción con sus actores y la interacción entre las funciones se obtiene puntualizando en cada una de las tareas definidas del diagrama anterior para así llegar al Diagrama de Flujo de Datos (Figura 1.3).

En las siguientes páginas se exponen estos diagramas que muestran la abstracción del funcionamiento del sistema con sus diferentes elementos.

### 1.1.1 Diagrama Entidad Relación

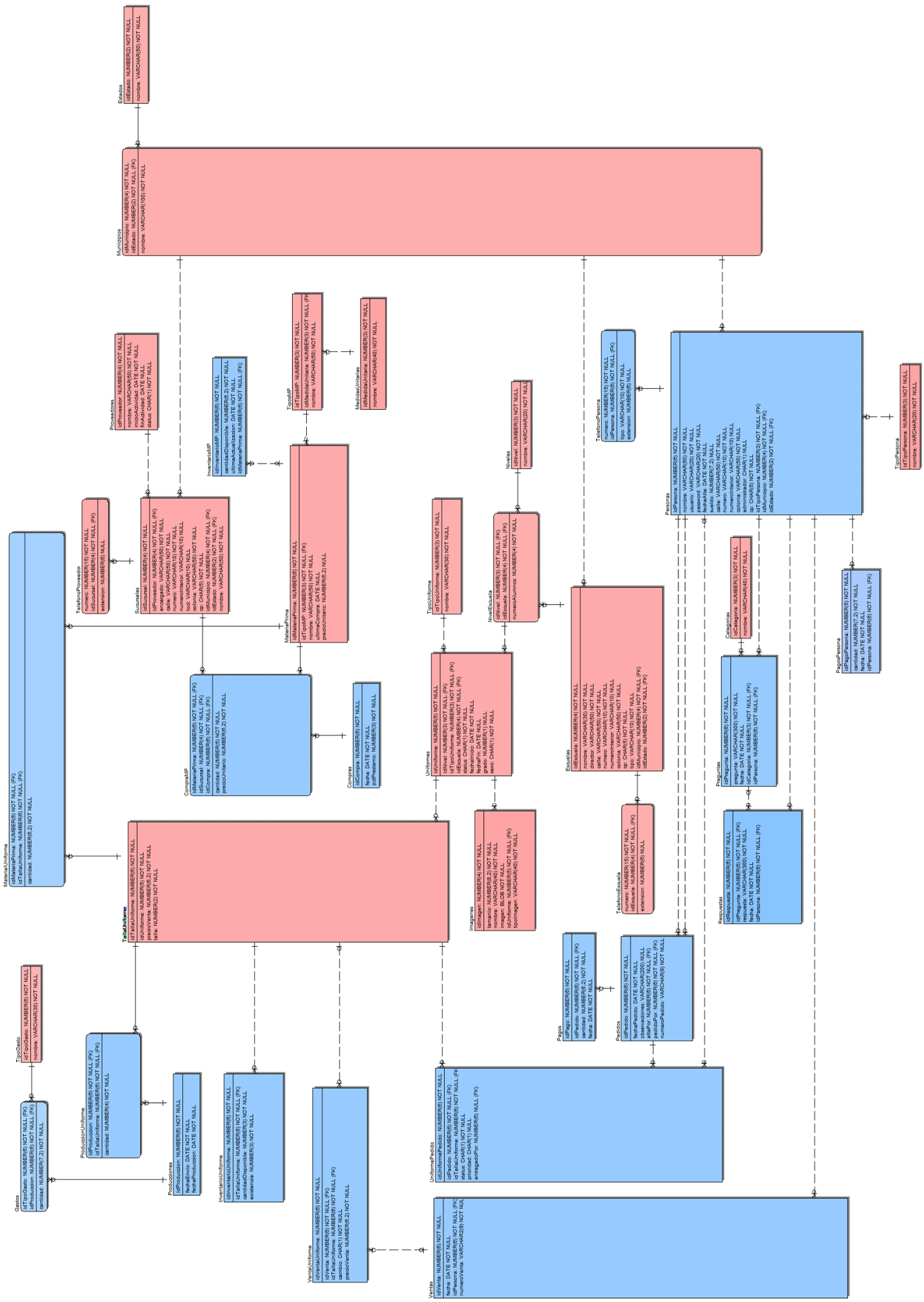

Figura 1.1: Diagrama Entidad Relación

### 1.1.2 Diagrama Jerárquico Funcional


Figura 1.2: Diagrama Jerárquico Funcional

### 1.1.3 Diagrama de Flujo de Datos

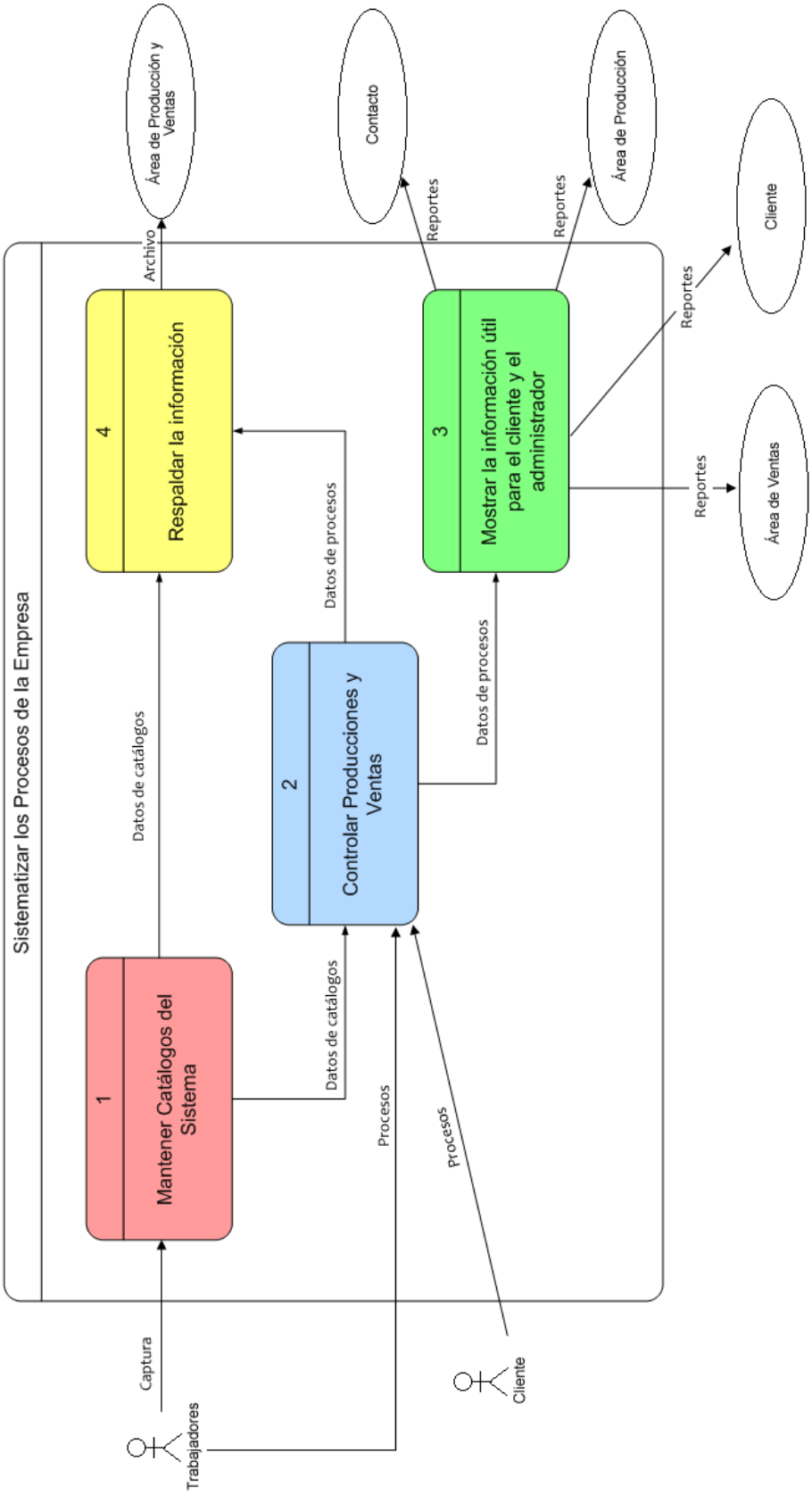


Figura 1.3: Diagrama de Flujo de Datos

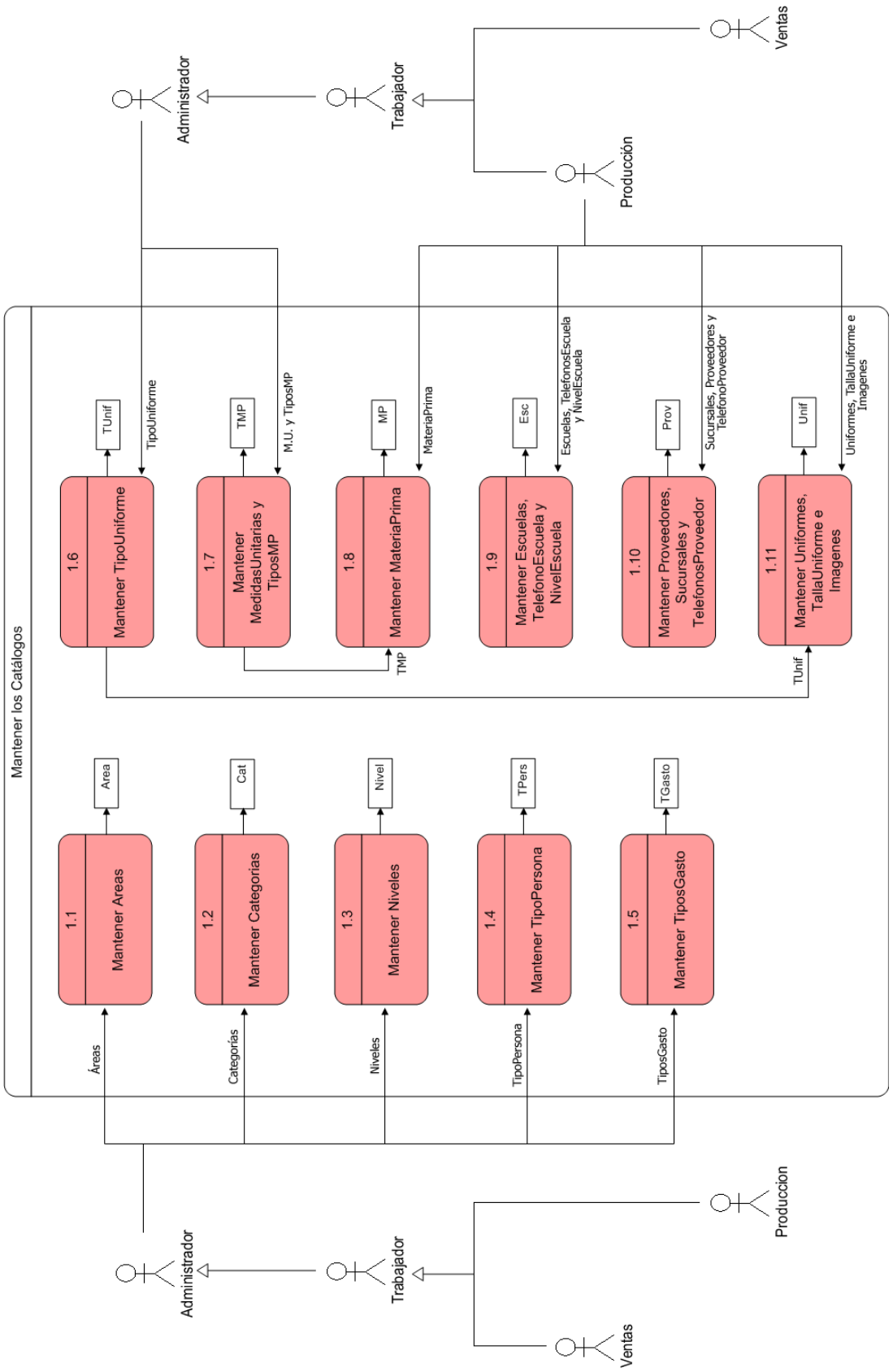


Figura 1.3 (*continuación*)

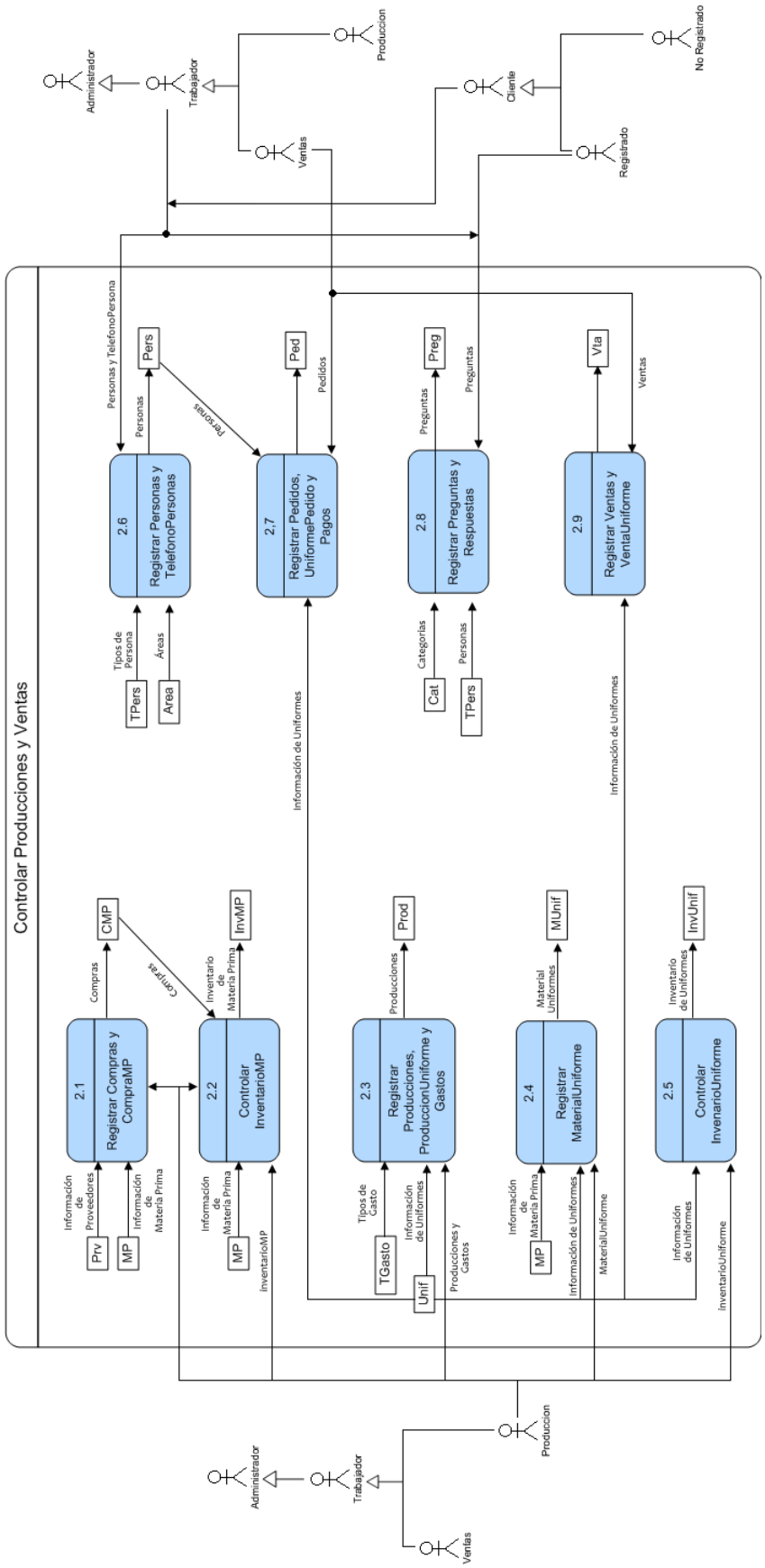


Figura 1.3 (*continuación*)

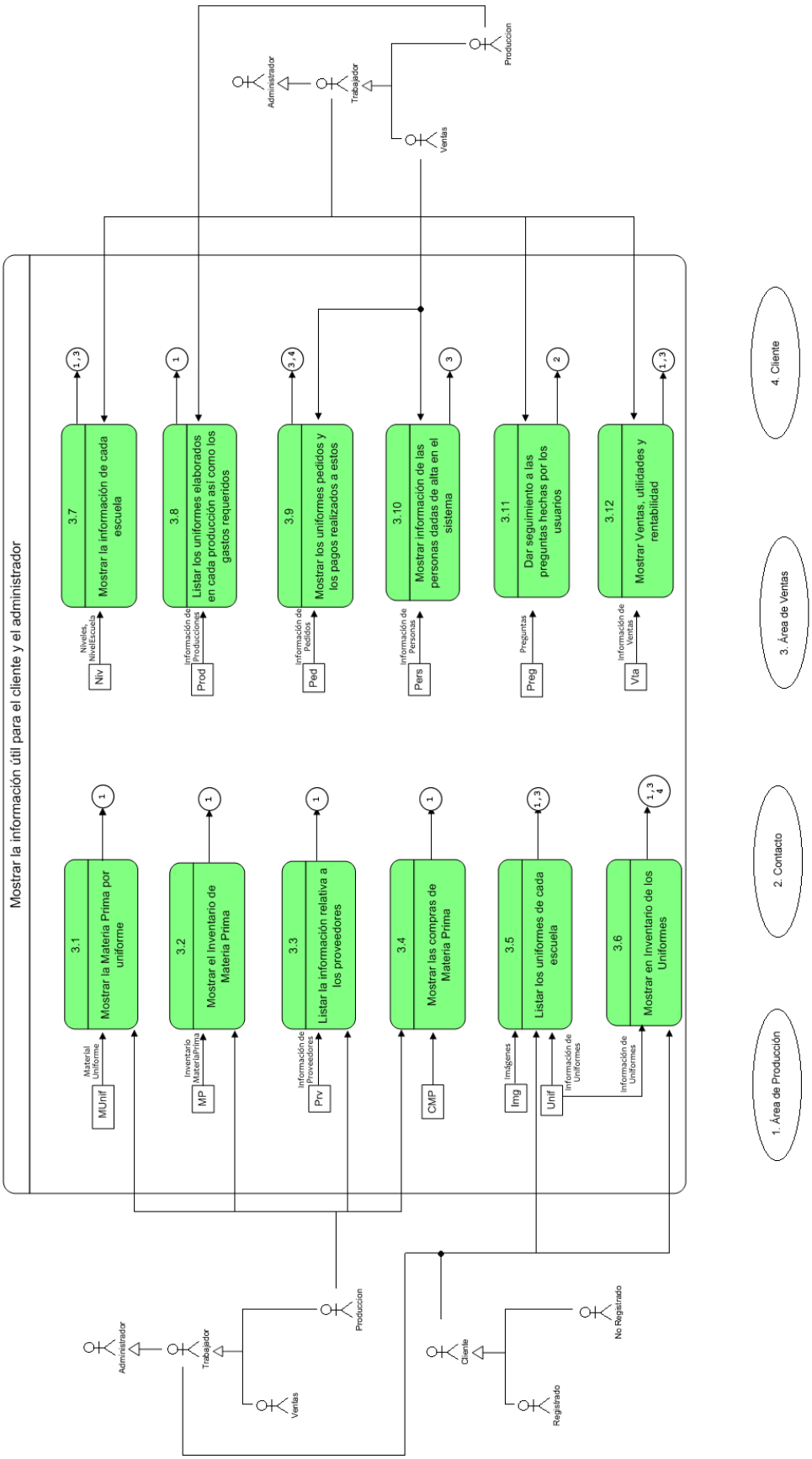


Figura 1.3 (*continuación*)

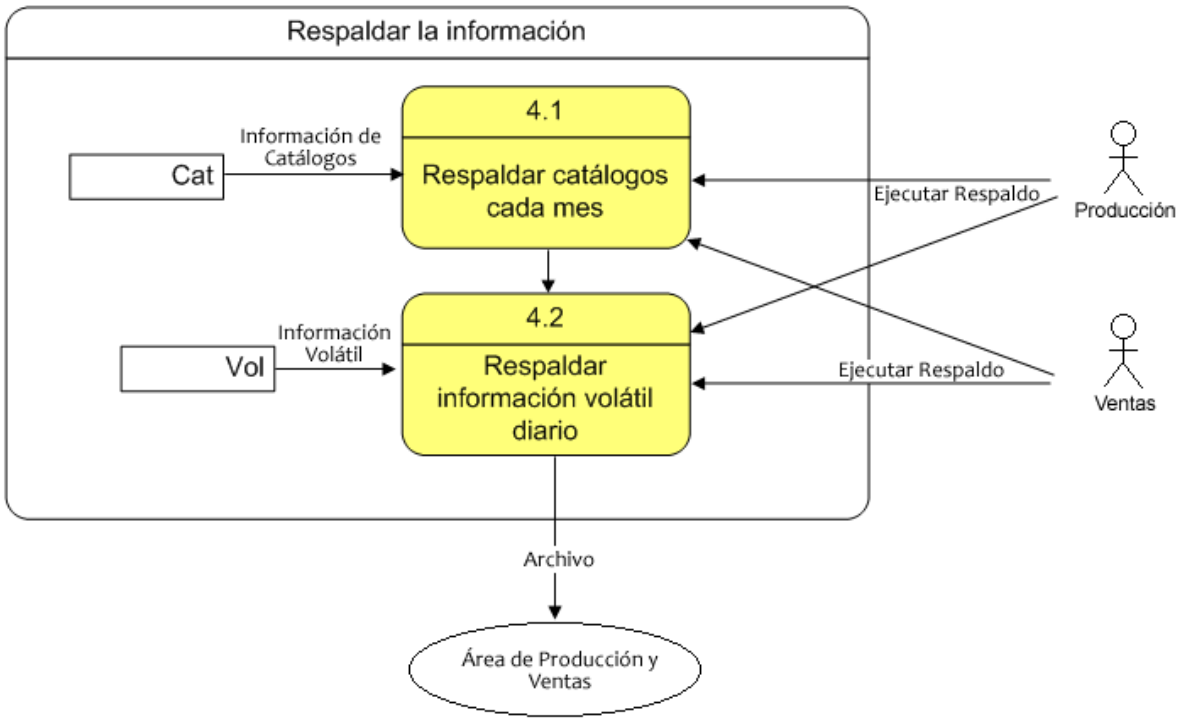


Figura 1.3 (*continuación*)

### 1.1.4 Tamaño de la Base OLTP

La información almacenada en la base de datos está organizada por medio de estructuras de almacenamiento lógicas (bloques, segmentos, tablespace) y físicas (archivos).

La base de datos se compone por al menos un tablespace, que es la unidad lógica de almacenamiento y en los cuáles se separan los diferentes grupos de objetos con que cuenta la base de datos (índices, tablas, usuarios). Cada tablespace tiene asociado diferentes archivos (datafiles) en los que se guarda la información física en el disco. El control de los tablespaces está dado por estructuras más pequeñas: bloques, segmentos y extensiones.

La unidad mínima de almacenamiento de datos es el bloque, compuesto por datos relativos a sí mismo (header) y los datos almacenados.

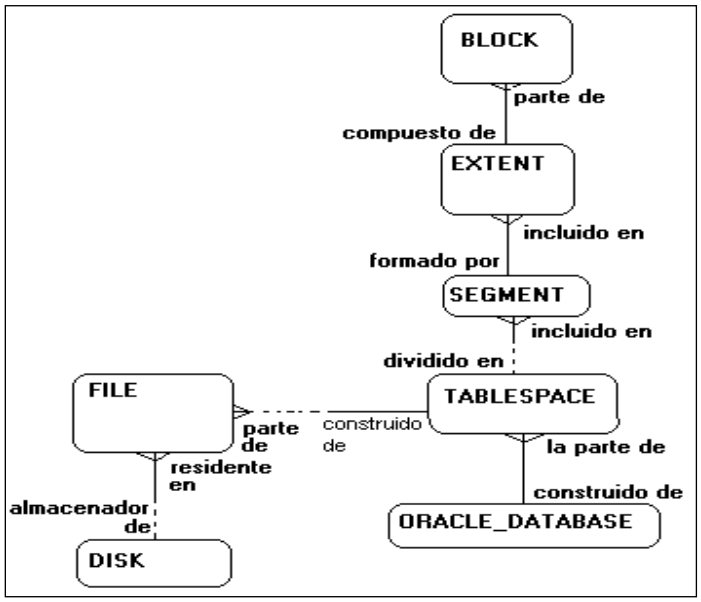


Figura 1.4: Almacenamiento físico en la Base de Datos

Un grupo de bloques forman la extensión, que tendrá un tamaño inicial fijo e irán creciendo conforme los datos y la configuración de ciertos parámetros: INITIAL, NEXT, PCTFREE, PCTUSED, INITTRANS, MAXTRANS, PCTINCREASE, MINEXTENTS y MAXEXTENTS (detallados más adelante).

Los segmentos están formados por extensiones y se encargarán de almacenar diferentes tipos de datos (rollback, temporales, índices y datos).

Es de suma importancia conocer el tamaño total de la base de datos para poder optimizar el rendimiento de esta, hacer un ajuste a su diseño (grado de normalización) o bien, elegir el hardware adecuado para su implementación.

Oracle ofrece ciertos parámetros en la creación de las tablas que permiten indicar el tamaño, número de extensiones y porcentaje de crecimiento que estas tendrán. A continuación se muestra la forma en que se llevan a cabo estos cálculos, que sumados, dan como resultado el tamaño total de la Base de Datos.

Tabla PROVEEDORES

| **Columna** | **TipoDato** | **Longitud máxima** | **Longitud + byte de separación** |
|---|---|---|---|
| inicioActividad | DATE NOT NULL | 7 | 8 |
| nombre | VARCHAR(50) NOT NULL | 30 + 20 = 50 | 51 |
| idProveedor | NUMBER(6) NOT NULL | Ceil(6/2+1) = 4 | 5 |
| status | CHAR(1) NOT NULL | 1 | 2 |
| finActividad | DATE NOT NULL | 7 | 8 |

74 + 5 = 79

**Posible espacio Libre:**
20 (debido al varchar2)

**Tamaño de bloque**
8 K

**Longitud de Registro:**
79

**Valor constante del Header**
360 bytes

**Número de Registros**
200

**Bytes libres del bloque**
7832

- **INITIAL:**

Es el tamaño de la extensión inicial, asignada una vez que se crea el objeto.

Initial = (bloques necesarios)(tamaño de bloque)

Bloques necesarios = (registros)(longitudRegistro)/bytesLibres

= (200)(79)/7832 = 2 bloques

INITIAL = 2 x 8 = 16 K

- **NEXT:**

Es el tamaño de la segunda extensión, una vez que INITIAL se ha llenado. Es recomendable que tanto INITIAL como NEXT sean del mismo valor.

- **PCTFREE:**

Indica el porcentaje de espacio libre que tendrá la tabla en cada renglón, debido a la existencia de datos nulos y de tipo VARCHAR2.

Se obtiene por medio de una regla de 3:

Si la longitud máxima de registro es el 100%, el posible espacio libre es X%

X = 100 (posibleEspacioLibre)/ longitudMaxima

= (100)(20)/79 = 25.3

PCTFREE = 25%

- **PCTUSED:**

Es el porcentaje mínimo de espacio requerido para la inserción de datos en un renglón.

Para calcular este parámetro, primero se calcula el porcentaje de espacio que ocupa cada registro

Si el tamaño de bloque es el 100%, la longitud del registro es el X%

X = 100(longitudDeRegistro)/tamañoDeBloque

= 100(79)/ 7832 = 1%

PCTUSED = 100 – (PCTFREE + X)

=100 – (25 + 1) =74

PCTUSED = 74%

- **INITTRANS:**

Número inicial de transacciones concurrentes en el bloque.

- **MAXTRANS:**

Número máximo de transacciones concurrentes en el bloque.

Se calcula dividiendo el tamaño del bloque entre el tamaño de registro.

MAXTRANS = 7832 / 79

MAXTRANS = 99

- **PCTINCREASE:**

Indica el porcentaje de crecimiento a partir de la tercera extensión (una vez que INITIAL y NEXT se llenan).

- **MINEXTENTS:**

Número mínimo de extensiones que serán asignadas a un objeto.

- **MAXEXTENTS:**

Número máximo de extensiones que un objeto podrá tener (el tamaño de estas dependerá del PCTINCREASE).

En la siguiente tabla se indica el resumen de los parámetros calculados para cada una de las tablas de la base de datos clasificadas según la volatilidad de su información y pensadas a un crecimiento de 10 años.

| Tabla | Tipo | Registros [Anual]* | INITIAL [K] | NEXT [K] | PCTFREE[%] | PCTUSED[%] | INITTRANS | MAXTRANS | PCTINCREASE | MINEXTENTS | MAXEXTENTS | Registros [10 años]** | Tamaño a 10 años |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Areas | Catálogo | 2 | 64 | 64 | 34 | 66 | 1 | 255 | 50 | 1 | 1 | 2 | 64 |
| Categorias | Catálogo | 5 | 64 | 64 | 20 | 80 | 1 | 160 | 50 | 1 | 1 | 10 | 64 |
| Escuelas | Catálogo | 10 | 64 | 64 | 34 | 66 | 1 | 33 | 50 | 1 | 1 | 20 | 64 |
| Estados | Catálogo | 2 | 64 | 64 | 42 | 58 | 1 | 133 | 50 | 1 | 1 | 2 | 64 |
| TipoGasto | Catálogo | 20 | 64 | 64 | 24 | 76 | 1 | 191 | 50 | 1 | 1 | 30 | 64 |
| Imágenes | Catálogo | 150 | 48 | 48 | 10 | 90 | 1 | 26 | 50 | 1 | 2 | 500 | 150 |
| MateriaPrima | Catálogo | 100 | 8 | 8 | 22 | 78 | 1 | 88 | 50 | 1 | 2 | 300 | 24 |
| MedidasUnitarias | Catálogo | 10 | 64 | 64 | 25 | 75 | 1 | 196 | 50 | 1 | 1 | 10 | 64 |
| Municipios | Catálogo | 500 | 64 | 64 | 35 | 65 | 1 | 69 | 50 | 1 | 1 | 500 | 64 |
| Niveles | Catálogo | 5 | 64 | 64 | 34 | 66 | 1 | 255 | 50 | 1 | 1 | 5 | 64 |
| NivelEscuela | Catálogo | 15 | 64 | 64 | 0 | 100 | 1 | 255 | 50 | 1 | 1 | 30 | 64 |
| Proveedores | Catálogo | 10 | 64 | 64 | 25 | 75 | 1 | 99 | 50 | 1 | 1 | 30 | 64 |
| Sucursales | Catálogo | 20 | 64 | 64 | 40 | 60 | 1 | 30 | 50 | 1 | 1 | 60 | 64 |
| TallaUniforme | Catálogo | 1500 | 40 | 40 | 0 | 100 | 1 | 255 | 50 | 1 | 1 | 3000 | 80 |
| TelefonoEscuela | Catálogo | 20 | 64 | 64 | 0 | 100 | 1 | 178 | 50 | 1 | 1 | 50 | 64 |
| TelefonoProveedor | Catálogo | 10 | 64 | 64 | 0 | 100 | 1 | 218 | 50 | 1 | 1 | 30 | 64 |
| TipoPersona | Catálogo | 2 | 64 | 64 | 33 | 67 | 1 | 255 | 50 | 1 | 1 | 2 | 64 |
| TiposMP | Catálogo | 20 | 64 | 64 | 32 | 68 | 1 | 126 | 50 | 1 | 1 | 30 | 64 |
| TipoUniforme | Catálogo | 20 | 64 | 64 | 13 | 87 | 1 | 201 | 50 | 1 | 1 | 30 | 64 |
| Uniformes | Catálogo | 150 | 64 | 64 | 0 | 100 | 1 | 224 | 50 | 1 | 1 | 300 | 64 |
| CompraMP | Volátil | 1000 | 24 | 24 | 0 | 100 | 1 | 255 | 50 | 1 | 4 | 10000 | 240 |
| Compras | Volátil | 100 | 64 | 64 | 0 | 100 | 1 | 255 | 50 | 1 | 1 | 1000 | 64 |
| Gastos | Volátil | 500 | 8 | 8 | 0 | 100 | 1 | 255 | 50 | 1 | 5 | 5000 | 120 |
| InventarioMP | Volátil | 100 | 64 | 64 | 14 | 86 | 1 | 218 | 50 | 1 | 1 | 300 | 64 |
| InventarioUniforme | Volátil | 1500 | 48 | 48 | 0 | 100 | 1 | 255 | 50 | 1 | 1 | 3000 | 100 |
| MaterialUniforme | Volátil | 15000 | 440 | 440 | 0 | 100 | 1 | 255 | 50 | 1 | 1 | 30000 | 880 |
| Pagos | Volátil | 1000 | 24 | 24 | 0 | 100 | 1 | 255 | 50 | 1 | 5 | 10000 | 256 |
| Pedidos | Volátil | 600 | 168 | 168 | 37 | 63 | 1 | 29 | 50 | 1 | 4 | 6000 | 1680 |
| Personas | Volátil | 2000 | 528 | 528 | 37 | 63 | 1 | 30 | 50 | 1 | 5 | 20000 | 5304 |
| Preguntas | Volátil | 500 | 168 | 168 | 31 | 69 | 1 | 24 | 50 | 1 | 5 | 5000 | 3232 |
| Producciones | Volátil | 100 | 64 | 64 | 0 | 100 | 1 | 255 | 50 | 1 | 1 | 1000 | 64 |
| ProduccionUniforme | Volátil | 2000 | 64 | 64 | 0 | 100 | 1 | 255 | 50 | 1 | 4 | 20000 | 640 |
| Respuestas | Volátil | 500 | 168 | 168 | 61 | 39 | 1 | 24 | 50 | 1 | 5 | 5000 | 1680 |
| TelefonoPersona | Volátil | 2000 | 72 | 72 | 28 | 72 | 1 | 218 | 50 | 1 | 2 | 20000 | 736 |
| UniformePedido | Volátil | 3000 | 80 | 80 | 0 | 100 | 1 | 255 | 50 | 1 | 4 | 30000 | 800 |
| Ventas | Volátil | 2000 | 40 | 40 | 0 | 100 | 1 | 255 | 50 | 1 | 3 | 20000 | 408 |
| VentaUniforme | Volátil | 10000 | 288 | 288 | 0 | 100 | 1 | 255 | 50 | 1 | 5 | 100000 | 2856 |
| Total | | 43956 | 3496 | | | | | | | | | 286016 | 20466 |

| | | |
|---|---|---|
| Total de datos: | 20466 | |
| + Índices (+20%) | 4093.2 | * Valores Aproximados |
| + Δ En incrementos de LOG | 7367.76 | ** Estimación a 10 años |
| | **32 M** | |

Tabla 1.1: Tamaño total de la Base de Datos

## 1.2 Especificaciones Técnicas

### 1.2.1 La interfaz de usuario

Se sabe que el ser humano tiende a reaccionar de manera diferente ante el estímulo propiciado por ciertos colores, debido a que estos forman parte del espectro lumínico y su energía tiende a afectarle directamente (ya sea positiva o negativamente).

Un error muy frecuente en el diseño de interfaces gráficas, es el uso erróneo de los colores para el mensaje que se desea transmitir y para el tipo de personas al que va dirigido. Por tal motivo, resulta conveniente plantear qué se quiere transmitir, las sensaciones que se desean estimular, el comportamiento que se esperaría obtener y el tipo de personas que usarán el sistema.

- El mensaje a transmitir:

La imagen de una empresa pequeña pero consolidada, que cuenta con experiencia y prestigio en la zona, conocimiento en el ramo, comprometida con la calidad y sus clientes y en búsqueda de expansión.

- Estímulos y comportamientos:

Que el cliente se sienta cómodo navegando por las opciones de la página, para que su atención esté enfocada en ella y no sienta la necesidad de cerrarla a los pocos minutos de haber ingresado.

De los trabajadores, se busca estimular el uso del sistema, pues en un principio podrían mostrar cierta aversión contra este. De manera que, al igual que con los clientes, se requiere que mientras hagan uso de este, lo encuentren atractivo, cómodo y fácil de usar.

- Hacía quién va dirigido:

No hay una edad específica en el uso de la aplicación. La página de internet será vista por niños, adolescentes, padres y quizá abuelos. El sistema de administración será usado por trabajadores (hombres y mujeres) de edad entre 20 y 60 años.

### 1.2.2 El color

Antes de hacer la elección de los colores, es indispensable entender el mensaje que cada uno de estos envía y cómo serán interpretados, con la finalidad de atraer aquellos que tendrán impactos positivas y evitar los que no.

- Blanco:

  Es el que posee la mayor sensibilidad frente a la luz por surgir de la unión de todos los colores. Su uso hace resaltar las tonalidades que se encuentren adyacentes a él.

  Siempre tendrá una connotación positiva pues se asocia con el optimismo, la paz, la felicidad, la simplicidad, la modestia y los buenos valores.

- Negro:

  Es la ausencia de todo color, no refleja ni emite luz. Es recomendable para catálogos on-line debido a que ayuda a resaltar los colores de las imágenes. Su uso aporta elegancia, paz, fortaleza, nobleza y ayuda a aumentar la profundidad de lo que se está viendo. Su efecto adelgazante en personas también se ve reflejado en una página saturada de elementos, ayudando a reducir la perspectiva de sobrecarga de estos.

  El exceso del negro produce un efecto intimidatorio y distante.

- Amarillo:

  Es el color más luminoso del espectro. Su uso moderado estimula la felicidad, la imaginación, la energía, la memoria y el apetito.

  En combinación con el negro, hace que la atención se dirija de inmediato al amarillo, por lo que es de gran utilidad para anuncios importantes que requieran atención.

  En su tono brillante, representa la juventud y espontaneidad de las cosas dejando detrás la formalidad y estabilidad. En un tono pálido representa envidia, celos, cobardía, debilidad y enfermedad.

  El exceso de amarillo produce agotamiento e irritación.

- Rojo:

  Ayuda a estimular el metabolismo del cuerpo humano, aumenta la presión sanguínea y el ritmo cardiaco.

Resalta el texto y las imágenes, poniéndolas en primer plano y ayudando a que las tomas de decisiones se lleven de manera rápida.

Por sí solo, representa fuerza, poder, vitalidad, agresividad, impulso y pasión. En combinación con el negro, estimula la imaginación, representa dominio y tiranía, combinándolo con blanco, representa inocencia y juventud.

Su uso en exceso provoca ansiedad, agitación y tensión.

- Naranja:

  Nace de la combinación del amarillo y el rojo. Produce mayor oxigenación en el cerebro por lo que estimula la actividad mental así como también la felicidad, la determinación, el apetito y una sensación acogedora (contraria al rojo).

  Es muy utilizado para la comunicación con la gente joven y para resaltar asuntos importantes dentro de una página.

  El naranja brillante representa amistad, amabilidad, carácter positivo y estimulante. En un tono más oscuro, o en combinación con el negro, representa engaño y desconfianza.

  Utilizar el naranja en exceso podría aumentar la ansiedad.

- Azul:

  Debido a que no se encuentra en los alimentos de la naturaleza, este color es un supresor del apetito, retarda el metabolismo y produce paz, tranquilidad, armonía, serenidad, estabilidad emocional y calma.

  En general, representa la elegancia, la verdad y la responsabilidad, en un tono claro da la sensación de frescura y frío, en su tono oscuro simboliza la sabiduría e inteligencia, mezclado con blanco la pureza, en combinación con el negro, la desesperación y con un color cálido provoca atención.

  El azul en exceso puede provocar depresión, tristeza y pasividad.

- Verde:

  Es el color al que más acostumbrado está el ojo humano por ser el que más se encuentra en la naturaleza, es beneficioso para el sistema nervioso, estimula la empatía y ayuda a equilibrar las emociones.

Representa la inexperiencia, el dinero y la seguridad, cuando en él predomina el amarillo se asocia con la cobardía, discordia y enfermedad, cuando predomina el azul se relaciona con la salud emocional, la protección y la sofisticación.
Su uso en exceso podría provocar baja de energía.

- Púrpura:
  Nace de la combinación del rojo y el azul. Produce sentimientos nostálgicos, serenidad y ayuda a los problemas nerviosos y mentales.
  Su uso va dirigido a niños, adolescentes y mujeres.
  Representa fantasía, dignidad, misterio, lucidez, templanza y experiencia.
  Abusar del uso del púrpura podría provocar dolor de cabeza y tristeza.

- Gris:
  El gris, por sí mismo, no es un color, sino una transición negro-blanco.
  Representa el éxito, la estabilidad, el prestigio y la tenacidad.

### 1.2.3 Diseño

El hecho de que un sistema (tanto web como de escritorio) desempeñe adecuadamente las funciones para el que fue programado, no es suficiente para garantizar el éxito de este. Un aspecto muy importante a considerar, es la interfaz gráfica (GUI por sus siglas en inglés *Graphical User Interface*) con la que el usuario va a interactuar.

Hoy en día, las GUI cada vez son más complejas y atractivas. Por lo que el programador se ve en la tarea de implementar varias funcionalidades gráficas en poco tiempo y si este no tiene un control adecuado sobre estas, el mantenimiento y correcciones pueden convertirse en un verdadero problema.

Para la GUI del sistema se utilizó CSS, una tecnología aplicada al HTML que la mayoría de las páginas actuales usan, JQuery que es una librería basada en Javascript para minimizar los trabajos de programación y JSP, que es el lenguaje de servidor soportado por J2EE (Java 2 Enterprise Edition).

## 1.3 CSS

CSS (Cascade Style Sheet) u hojas de estilo en cascada, es un lenguaje especificado por el W3C (World Wide Web Consortium) como un estándar para los navegadores web. Es utilizado para separar los contenidos de la página con la manera en que estos serán presentados.

El uso de hojas de estilos puede hacerse de tres maneras diferentes:

1. Estilo en línea: Aplicar el estilo directamente en la etiqueta en la que se requiera
2. Hoja de Estilo Interna: Se usa dentro de la etiqueta `<head>` y las reglas de estilo deben ser especificadas para cada una de las páginas que formen parte del sistema.
3. Hoja de Estilo Externa: Se centralizan las cláusulas de los estilos en un archivo externo de extensión `.css` y se mandan a llamar desde las páginas correspondientes.

La última opción es la que supone una mejor práctica para el uso de CSS puesto que de esta forma es como realmente se está aplicando la separación del contenido y la presentación concentrando los estilos en un archivo externo.

### 1.3.1 Estilos y Etiquetas

Una de las grandes ventajas de CSS es el ahorro y reutilización de código, puesto que una etiqueta HTML puede ser personalizada para que a lo largo del sistema sea usada con la misma definición evitando repetición de código y haciendo más fácil las tareas de mantenimiento.

Por ejemplo: Suponiendo que se desea tener de fondo una imagen llamada “logo.png” para todas las páginas que componen el sistema. Sin el uso de hojas de estilo. Esto quedaría de la siguiente manera:

```
1   <body background=”logo.png”>
2       […]
3   </body>
```

Código 1.1: Imagen de fondo en página HTML sin CSS.

Lo anterior funciona perfectamente, sin embargo, si el sistema cuenta con un número extenso de páginas, el atributo `background` deberá ser especificado en cada una de ellas, y si por requerimientos del sistema, se pide que cada cierto tiempo esta imagen cambie por otras, con

otro nombre y/u otra extensión, la actualización puede ser bastante tardada y tediosa. Es por ello, que CSS entra en juego para evitar este tipo de problemas.

Lo que se hace, es personalizar la etiqueta <body> por medio de una hoja de estilo, en la cual se especificarán los atributos correspondientes a esta:

```
1   body{
2        background-image:url(logo.png);
3   }
```

Código 1.2: Imagen de fondo en página HTML con CSS y etiquetas.

```
1    <head>
2       <link rel="stylesheet" href="estilos.css" type="text/css" />
3    </head>
4    <body>
5       […]
6    </body>
```

Código 1.2 (*continuación*)

Como se puede notar, cualquier cambio que se tenga que hacer a la imagen del fondo, bastaría con alterar la definición de la etiqueta <body> hecha en el archivo estilos.css para que en cuestión de segundos la modificación sea terminada.

Para mandar a llamar la hoja de estilo externa, se coloca la etiqueta <link> con sus atributos:

- rel: Muestra la relación que hay entre la página y el archivo externo
- href: Indica la ubicación del archivo .css
- type: Especifica qué tipo de archivo se está llamando

### 1.3.2 Estilos y Clases

Hay ocasiones en que la personalización de etiquetas no puede ser posible debido a que no sería viable que en todo el sistema una misma etiqueta tuviera el mismo comportamiento cuando sea usada, tal es el caso de `<font>`. No obstante, se pueden agrupar ciertas características compartidas y definirlas en una clase.

Por ejemplo: Si el sistema requiere que los títulos sean tamaño grande, color rojo y letra tipo "Arial", los subtítulos, tamaño mediano, color azul y letra tipo "Verdana", y el resto del texto, tamaño pequeño, color negro y letra tipo "Calibri". Sin utilizar hojas de estilo quedaría de la siguiente manera:

```
<body>
<font face=”Arial” color=”#FF0000“ size=”6”>Título</font>
<font face=”Verdana” color=”#0000FF“ size=”4”>Subtítulo</font>
<font face=”Calibri” color=”#000000“ size=”2”>Texto</font>
</body>
```

Código 1.3: Tipos de letra en página HTML sin CSS.

Al igual que en el ejemplo del punto anterior, la repetición de código en cada página es necesaria, por lo que es bastante ineficiente programar de esta manera. La solución de CSS consistiría en separar en clases las diferentes definiciones de las letras.

```
1    .Titulos{
2        font-family:Arial;
3        font-size:16px;
4        color:#FF0000;
5    }
6    .Subtitulos{
7        font-family:Verdana;
8        font-size:12px;
9        color:#0000FF;
10   }
```

Código 1.4: Tipos de letra en página HTML con CSS y clases.

```
11   .Texto{
12        font-family:Calibri;
13        font-size:8px;
14        color:#000000;
15   }
```

Código 1.4 (*continuación*)

```
1    <body>
2        <div class="Titulos">Título</div>
3        <div class="Subtitulos">Subtítulo</div>
4        <div class="Texto">Texto</div>
5    </body>
```

Código 1.4 (*continuación*)

Para indicar que se trata de una clase, se antecede con un punto (.) al nombre de esta, y posteriormente se declara su definición.

La etiqueta `<div>` permite agrupar en bloques las clases que se están llamando para su separación.

### 1.3.3 Estilos e Id's

La declaración de estilos por medio de su id, es bastante similar a la anterior, con la diferencia de que estos sólo podrán ser utilizados una vez en toda la página.

Una de sus principales aplicaciones es para posicionar los objetos en la GUI de manera que resulte más sencillo que con el uso de tablas, pues en vez de ello, se hace uso de capas que se declaran en la hoja de estilo por medio de un id (único para cada capa).

```
1     #Capa1 {
2           position:absolute;
3           left:313px;
4           top:49px;
5           width:256px;
6           height:233px;
7     }
8     #Capa2 {
9           position:absolute;
10          left:318px;
11          top:57px;
12          width:115px;
13          height:219px;
14     }
15     #Capa3 {
16         position:absolute;
17          left:440px;
18          top:56px;
19          width:122px;
20          height:220px;
21      }
```

Código 1.5: Capas en página HTML con CSS e ids.

```
1     <body>
2         <div id="Capa1">[…]</div>
3         <div id="Capa2">[…]</div>
4         <div id="Capa3">[…]</div>
5     </body>
```

Código 1.5 (*continuación*)

Para indicar que la declaración se hace a través de su id, se antecede el símbolo número (#) al nombre del id y posteriormente se declaran los atributos relativos a la posición de la capa, tales como largo, ancho, posición en la pantalla, color de fondo, imagen de fondo, márgenes, bordes, etcétera.

## 1.4 jQuery

jQuery es una librería de código abierto desarrollada en 2006 para facilitar la programación visual y funcional del lado del cliente. Es muy liviana y al estar basada en javascript, permite implementar en su totalidad las funcionalidades de este último, pero con las siguientes ventajas:

- Ahorro de líneas de código.
- Curva de aprendizaje bastante reducida en comparación con javascript.
- Integración sencilla con Ajax.
- Ahorro en tiempo de desarrollo y mantenimiento.
- Compatibilidad de navegadores.
- Fácil manejo con CSS.

### 1.4.1 Funciones de jQuery

La sintaxis que maneja jQuery es bastante sencilla y fácil de aplicar:

```
$(elemento).evento(función(){
   […]
});
```

Por medio de su función principal `$()` se seleccionan los elementos de la página, estos pueden ser por clase, por identificador o por etiquetas para ser aplicados a algún método de conveniencia. Las funciones que ofrece jQuery pueden clasificarse de acuerdo a su uso:

- **Manipulación de los elementos:**

| Función | Descripción |
|---|---|
| .addClass(clase) | Permite agregar una clase al elemento seleccionado. |
| .attr(atributo)<br>.attr(atributo,valor) | Obtiene el valor del atributo marcado en el primer parámetro. En caso de ser especificado el segundo, actualiza el atributo con el valor indicado. |

Tabla 1.2: Funciones principales de jQuery para la manipulación de elementos en una página.

| Función | Descripción |
|---|---|
| .css(propiedad)<br>.css(propiedad,valor) | Obtiene el valor de la propiedad CSS especificada en el primer parámetro. En caso de ser especificado el segundo, actualiza la propiedad con el valor indicado. |
| .html()<br>.html(codigoHTML) | Obtiene el código HTML del elemento seleccionado. De manera adicional recibe un argumento en código HTML, que de ser especificado, será agregado al elemento. |
| .removeAttr(atributo) | Elimina el atributo especificado de los elementos seleccionados. |
| .removeClass(clase) | Elimina la clase especificada de los elementos seleccionados. |

Tabla 1.2 (*continuación*)

- **Efectos Visuales:**

| Función | Descripción |
|---|---|
| .fadeIn(velocidad) | Modifica la transparencia del elemento desde 0. La velocidad debe ser indicada con “fast”, “slow” o “normal”. |
| .fadeOut(velocidad) | Modifica la transparencia del elemento desde el nivel actual hasta 0. |
| .fadeTo(velocidad, transparencia) | Modifica la transparencia del elemento desde el nivel actual hasta el indicado (comprendido entre 0 y 1). |
| .hide() | Oculta el elemento seleccionado. |
| .show() | Muestra el elemento seleccionado. |

Tabla 1.3: Funciones principales de jQuery para dar efectos visuales a los elementos de una página.

- **Eventos:**

| Función | Descripción |
|---|---|
| .blur(), .change(), .click(), .dblclick(), .focus(), .keydown(), .keypress(), .keyup(), .mousedown(), .mouseenter(), .mouseleave() ,.mousemove(), .mouseout(), .mouseover(), .mouseup(), | jQuery ofrece estas funciones que permiten la captura de un evento para ejecutar cierta acción. |
| .hover(mouseEntra1,mouseSale2) | Engloba en un mismo evento cuando el mouse entra y sale del área del elemento seleccionado y las acciones a realizar en cada caso. |
| .ready() | Asegura que los elementos de la página se encuentren listos para ser manipulados. |

Tabla 1.4: Funciones principales de jQuery para el manejo de eventos.

- **Navegación a través de los elementos:**

| Función | Descripción |
|---|---|
| . children() | Obtiene los elementos hijos del seleccionado. |
| .each() | Itera sobre los objetos jQuery para ejecutar cierta acción en cada uno de estos. |
| .parent() | Obtiene el elemento padre del seleccionado. |

Tabla 1.5: Funciones principales de jQuery para la navegación de elementos de una página.

### 1.4.2 Ajax (Asynchronous JavaScript and XML)

La tendencia actual de desarrollo web está encaminada al uso de Ajax para mejorar la eficiencia y vista de los sistemas haciéndolos más interactivos. Su principal aplicación es actualizar fragmentos de página sin tener que ser recargada completamente. jQuery ofrece distintos métodos para la integración de Ajax a la aplicación:

### 1.4.2.1 load()

Es la función más simple de ajax, encargada de mostrar una página en la capa indicada. Su uso se ejemplifica a continuación:

```
1     $(".links").click(function(){
2         var pagina=$(this).attr("href");
3         $("#Centro").load(pagina);
4         return false;
5     });
```

Código 1.6: Funcionamiento de `load()`.

Cuando se da clic en cualquier vínculo perteneciente a la clase `links`, con ayuda de la función `attr()` se guarda en la variable `pagina` el atributo `href` del link seleccionado, para posteriormente cargar la respuesta del servidor con la función `load()` en la capa `Centro` (previamente definida en CSS). Finalmente se regresa `false`, para que la página requerida no sea redirigida sino que sea cargada en la capa indicada.

### 1.4.2.2 ajax()

Aunque muy similar a la función anterior, `ajax()` es más completa y permite la configuración de parámetros extra para poder manipular datos del usuario. Por lo anterior, se aplica para enviar información de formularios y ser procesados. Sus parámetros principales se muestran a continuación:

| Parámetro | Descripción |
|---|---|
| async | Permite indicar que la carga se haga de manera síncrona o asíncrona. Es recomendable que se haga de manera asíncrona para que no interfiera con la carga página. |
| data | Especifica los datos del formulario que serán enviados al servidor. |
| error | En caso de que ocurra un error, se ejecutará y regresará los detalles de este. |

Tabla 1.6: Parámetros de `ajax()`.

| Parámetro | Descripción |
|---|---|
| success | Una vez que la ejecución haya sido exitosa, se envían los datos al servidor y se realiza la acción corresponiente. |
| type | Especifica el método de envío de datos que será utilizado: POST o GET |
| url | Indica la ruta que procesará los datos del formulario. |

Tabla 1.6 (*continuación*)

Un ejemplo de su uso para el procesamiento de formularios se muestra a continuación:

```
1     $("form").submit(function(){
2         $.ajax({
3             type:"POST",
4             async:true,
5             url:$(this).attr("action"),
6             data:$(this).serialize(),
7             success: function(data){
8                 $("#Centro").html(data);},
9             error:function(){
10                alert("Hubo un error");}
9         });
10         return false;
11     });
```

Código 1.7: Funcionamiento de `ajax()`.

La llamada a `ajax()` comienza una vez que se ha dado `submit` al formulario. En la línea 6, `serialize` se encarga de concatenar los datos en una cadena del tipo POST o GET para ser procesada por la URL indicada. Si la ejecución de `ajax()` se hizo de manera correcta, se llama al parámetro `success` y se envían los datos recogidos en el formulario para que el resultado del procesamiento se muestre en la capa que se indica, en este caso Centro.

### 1.4.2.3post() / get()

Estas funciones permiten lanzar peticiones del tipo POST o GET al servidor, según se requiera. Son especialmente útiles para el llenado dinámico de combos dependientes. Reciben tres parámetros: La URL que se encargará de procesar la petición, los datos que serán enviados al servidor con la forma {nombre : valor} y una función, llamada de callback, que será invocada una vez que termine el procesamiento. El siguiente ejemplo muestra un combo que contiene los estados del país y al momento que este cambia, se imprimen en un segundo combo los municipios del estado seleccionado.

```
1     $("#estados").change(function(){
2         var estado=$("#estados").attr("value");
3         $.post("/URLDeProcesamiento",
4             {idestado: estado},
5              function(data){
6             $("#municipios").html(data);
7         });
8     });
```

Código 1.8: Funcionamiento de `post()`.