SISTEMA AUTOMÁTICO PARA TRANSFERENCIA DE DATOS HACIA UNA BASE DE DATOS ORACLE

JOSE EMMANUEL QUISPE SEGURA
GABRIEL FERNANDO SIERRA ROMERO

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERIA DE SISTEMAS
ESCUELA DE CIENCIAS NATURALES E INGENIERIA
SISTEMAS DE INFORMACIÓN E INGENIERÍA DE SOFTWARE
BUCARAMANGA
SEPTIEMBRE 15 DE 2005

SISTEMA AUTOMÁTICO PARA TRANSFERENCIA DE DATOS HACIA UNA BASE DE DATOS ORACLE

JOSE EMMANUEL QUISPE SEGURA
GABRIEL FERNANDO SIERRA ROMERO

TRABAJO DE GRADO PARA OBTENER EL TITULO DE INGENIERO DE SISTEMAS

DIRECTOR:
ING. MARTHA LUCIA ORELLANA, UNAB
CODIRECTOR:
ING. RAFAEL GOMEZ, ICP
ASESOR:
DINO ISAAC RODRÍGUEZ, ICP

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERIA DE SISTEMAS
ESCUELA DE CIENCIAS NATURALES E INGENIERIA
SISTEMAS DE INFORMACIÓN E INGENIERÍA DE SOFTWARE
BUCARAMANGA
SEPTIEMBRE 15 DE 2005

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

Firma del Presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bucaramanga, 15 Septiembre de 2005

DEDICATORIA

AGRADECIMIENTOS

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

ARE: Automatic Results Entry (Entrada Automática de Resultados), es un archivo plano generado por las interfaces de conexión donde se almacenan los datos necesarios a transferir al sistema de información de los laboratorios.

ASCII: Acrónimo de American Standard Code for Information Interchange (Código Normalizado Americano para el Intercambio de Información). En computación, un esquema de codificación que asigna valores numéricos a las letras, números, signos de puntuación y algunos otros caracteres. Al normalizar los valores utilizados para dichos caracteres, ASCII permite que los ordenadores o computadoras y programas informáticos intercambien información.

ENSAYOS: Acción que se le realiza a una muestra para conocer su composición química, su resistencia, su estabilidad, volatibilidad, etc., dependiendo del tipo del material.

INTERFAZ: punto en el que se establece una conexión entre dos elementos, que les permite trabajar juntos. La interfaz es el medio que permite la interacción entre esos elementos. En el campo de la informática se distinguen diversos tipos de interfaces que actúan a diversos niveles, desde las interfaces claramente visibles, que permiten a las personas comunicarse con los programas, hasta las imprescindibles interfaces hardware, a menudo invisibles, que conectan entre sí los dispositivos y componentes dentro de los ordenadores o computadoras.

INSTRUMENTOS DE LABORATORIO: Maquinas electrónicas que sirven para realizar diferentes tipos de pruebas a diferentes muestras. Ejemplo: Espectrómetro, Cromatógrafo.

MUESTRAS: Tipo de material al cual se le realiza diferentes pruebas para analizar los elementos químicos que lo conforman.

PLATAFORMAS: Sinónimo de Sistema Operativo; software básico que controla una computadora. El sistema operativo tiene tres grandes funciones: coordina y manipula el hardware del ordenador o computadora, como la memoria, las impresoras, las unidades de disco, el teclado o el Mouse; organiza los archivos en diversos dispositivos de almacenamiento, como discos flexibles, discos duros, discos compactos o cintas magnéticas, y gestiona los errores de hardware y la pérdida de datos.

PRUEBAS: Sinónimo de Ensayo.

SÍMBOLOS: Es una entidad abstracta que no necesitamos definir formalmente, como por ejemplo una letra del alfabeto o un dígito. Conjunto de caracteres ASCII.

TRANSFER: Nombre dado al Software de Transferencia automática de datos creado para este proyecto.

RESUMEN

En el Laboratorio de Resistencia de Materiales del Instituto Colombiano del Petróleo ICP se requiere tiempo y personal capacitado para configurar una interfaz para la manipulación y transferencia automática de archivos texto generados por cada instrumento de ensayos de muestras, hacia el Sistema de Información de Laboratorios, utilizando la herramienta software con que cuentan actualmente. Adicionalmente, no se cuenta con facilidades que permitan seleccionar la información que va a ser transferida, lo que genera acumulación de una alta cantidad de datos ociosos en la base de datos, que desmejoran la eficiencia de la aplicación.

Por este motivo se hace necesario disponer de una aplicación que le permita al usuario, primero seleccionar los datos que ameriten ser transferidos y segundo transferir los datos seleccionados, en forma automática, al Sistema de Información de Laboratorios, sin que esto implique conocimientos y manipulación de código en algún lenguaje o herramienta específica, por parte del usuario final. De esta manera se evita el tratamiento de información por personal de Instituto evitando errores de digitación que pudieran presentarse, y se filtran adecuadamente los datos que serán exportados al Sistema.

Con este proyecto se pretende proporcionar al usuario una aplicación software que procese archivos ASCII generados por los diferentes ensayos, en el Laboratorio de Resistencia de Materiales, y transfiera los datos al Sistema de Información de Laboratorios, con un diseño flexible que permita y facilite el manejo de los datos generados por otros instrumentos de ensayo en otros laboratorios.

INTRODUCCIÓN

El sistema de información está enfocado a manipular cadenas de caracteres o archivos tipo texto y la transferencia de los mismos, donde es posible pensar en la realización computacional de un procesador de texto, utilizando una herramienta de software, como un generador de expresiones regulares.

A partir de un enfoque de adquisición y manipulación de datos, recolectando y dando formato a la información recibida del entorno externo, interviene un componente de interacción que permitirá al usuario seleccionar información de acuerdo con lo requerido por cada uno de los ensayos y un componente de control de salida para la transferencia de los datos que apoye la generación de resultados.

El proceso de desarrollo está basado en el Modelo Lineal Secuencial en cascada con iteraciones de ajustes integradas. Las siguientes herramientas fueron estudiadas en la etapa de búsqueda de soluciones para el problema: C++, Java y Perl, para la implementación del mecanismo de manipulación de caracteres; Oracle Developer y Visual Studio para la creación de la interfaz de usuario; y Oracle, como manejador de base de datos.

1. OBJETIVO DEL PROYECTO

1.1 OBJETIVO GENERAL

Construir una interfaz de comunicaciones, de un equipo determinado del Laboratorio de Resistencia de Materiales del Instituto Colombiano del Petróleo, ICP, con la red corporativa, que permita la transferencia automática de datos al Sistema de Información de Laboratorios, donde el usuario pueda seleccionar la información que amerite ser transferida para dar respuesta a las necesidades de sus clientes, y teniendo en cuenta las consideraciones necesarias para obtener una herramienta genérica y flexible que permita y facilite su implementación posterior en los equipos de otros laboratorios, independientemente de la forma en que sea generada la información de las pruebas por estos equipos.

1.2. OBJETIVOS ESPECIFICOS

Implementar un mecanismo de manipulación de caracteres, que tomando como entrada un archivo ASCII original generado por el equipo de laboratorio, extraiga datos específicos de este archivo y los organice y despliegue en un formato determinado.

Diseñar e implementar una interfaz que permita al usuario seleccionar, del formato determinado, los datos a ser transferidos al Sistema de Información de Laboratorios.

Habilitar la transferencia automática de resultados del equipo de laboratorio, al Sistema de Información de Laboratorios, para las pruebas que sean definidas.

Contemplar las consideraciones de diseño necesarias para proporcionar al ICP una herramienta flexible que facilite su implementación posterior para la selección y transferencia de resultados generados por otros equipos de laboratorio en la Institución.

2. JUSTIFICACIÓN

Numerosos equipos de los laboratorios del ICP, generan datos que deben ser ingresados manualmente al Sistema de Información de laboratorios, pues carecen de una interfaz que permita su transferencia automática al sistema o en algunos casos su nivel de programación es complejo. Los equipos de laboratorio que generan los datos obtenidos en las diferentes pruebas, no ofrecen facilidades para el tratamiento o selección de esta información antes de ser llevada al Sistema de Información de Laboratorios, originando la acumulación de datos inactivos que congestionan la Aplicación.

Los equipos utilizados por ensayos de laboratorios pertenecen a la unidad de Servicios Técnicos Especializados, que tiene como finalidad dar soporte a clientes internos y externos del ICP.

Dentro de la operación de los laboratorios del ICP, una labor muy importante es el aseguramiento de la información. Los datos generados por los instrumentos del laboratorio deben ser transferidos al Sistema de Información de Laboratorios para garantizar la integridad y disponibilidad de estos para ser consultados por clientes internos (de ECOPETROL) y generar informes de resultados para los clientes externos.

El ICP cuenta con las herramientas de conexión IDM/SEEDPAK2 y ARE (Automatic Result Entry), que permiten, con base en el archivo generado en la prueba, generar un segundo archivo con los datos dispuestos en la forma requerida para ser transferidos al Sistema de Información de Laboratorios. Aunque esta herramienta permite la conexión con la Base de Datos, su uso se dificulta debido a que es necesario realizar configuraciones personalizadas para adaptarla

a cada prueba, tarea que demanda tiempo y dedicación de las personas con los conocimientos necesarios para realizar esta labor. Adicionalmente, no ofrece la facilidad de seleccionar la información que se requiere transferir a la Base de Datos.

No todas las plataformas sobre las que corren estas aplicaciones son compatibles con la Base de Datos de Laboratorios, lo que hace necesario el manejo de los datos para su grabación por interfaces desarrolladas independientemente para cada prueba o la disponibilidad de personal para realizar el ingreso manual de los mismos, con los posibles errores de digitación que esto implica.

Se hace necesario disponer de una aplicación que permita al personal de los laboratorios seleccionar los datos que se necesiten enviar al Sistema de Información de Laboratorios, garantizando la integridad de estos datos, y que pueda ser fácilmente personalizada para los diferentes tipos de resultados que puedan generar los ensayos. El objetivo de la selección es separar los datos ociosos, de los datos requeridos para la elaboración de los informes. La selección de los datos podrá ser realizada en el mismo laboratorio y por las mismas personas que participen en la prueba, evitando el tratamiento de información por personal de Sistemas.

La herramienta a construir procesará archivos ASCII generados por las diferentes pruebas; archivos donde el formato de presentación de los datos varía dependiendo del tipo de laboratorio y del tipo de prueba que se realice. Adicionalmente, permitirá la selección de estos datos por el usuario final y su transferencia, en forma automática, al Sistema de Información de Laboratorios, evitando errores en la trascripción de la información y permitiendo que el tiempo que el analista emplearía en labores rutinarias de digitación, pueda emplearlo en el análisis de la información obtenida.

El ahorro en tiempo de manipulación e ingreso de resultados a la Base de Datos será considerable, si se tiene en cuenta que actualmente algunos laboratorios llegan a correr 100 pruebas en un día con generación hasta de 2000 datos por cada prueba. Con el desarrollo de este Proyecto se podrá ofrecer oportunidad y calidad en la información, agilizando los procesos del laboratorio y de toma de decisiones.

3. ANTECEDENTES

En los años 70, pocas personas podrían haber descrito sabiamente lo que significaba un programa de computadora o software. Hoy, la mayoría de estudiantes y profesionales piensan y tratan de entender el software. Para comprenderlo es necesario examinar las características que lo diferencian de otras cosas que el hombre puede construir; esto es un proceso creativo que comprende análisis, diseño, implementación y pruebas, y que se traduce finalmente en una forma de aplicación o un sistema de información.

Hoy en día el software tiene un doble papel. Es un producto y al mismo tiempo el medio para ser entregado. Este producto hace el control informático que incorpora desde hardware hasta redes de computadoras. El software es un transformador de información, produciendo, gestionando, adquiriendo, modificando, mostrando o transmitiendo información que puede ser simple o compleja según el entorno en el que reside.

Los sistemas se han convertido en una parte fundamental para la administración de la información. Son la máquina que conduce a la toma de decisiones comerciales y sirve de base para la investigación científica y la solución de problemas de ingeniería. El software está inmerso en sistemas de todo tipo: de transporte, médico, procesos industriales, telecomunicaciones, entre otros. Es ineludible a nuestro mundo moderno. A medida que seguimos avanzando en el tiempo, será el que nos conduzca a nuevos avances tecnológicos.[1]

3.1 SILAB

El Sistema de Información de Laboratorios, SQL*LIMS ó SILAB de la empresa Applied Biosystems, fue adquirido en el año 1994 por medio de un contrato con PerkinElmer, con el propósito de dotar a ECOPETROL de una solución informática que apoyara la toma de decisiones basada en la información sobre calidad de las materias primas, corrientes de procesos y productos que generan los laboratorios del CIB (Complejo Industrial de Barrancabermeja), CAR (Refinería de Cartagena) e ICP.

Este sistema tuvo una fase de implementación en el ICP, desde enero de 1998 hasta noviembre de 1999, que comprendió adecuaciones del sistema, revisión de conectividad e integración del sistema en los laboratorios, dotando al ICP de una solución informática para gerenciar las actividades realizadas en los laboratorios, como resultado de un proyecto de investigación.

Para administrar los datos de SILAB, se encuentra implementado desde el año 1995, en el ICP, el sistema gestor de base de datos ORACLE 8i. Con este gestor de bases de datos se han obtenido buenos resultados por su estabilidad, seguridad, fiabilidad, siendo uno de los motores más potentes actualmente en el mercado.

3.2 IDM/SEEDPAK2

El software de conectividad consta de varios programas desarrollados en Lenguaje IDXL, uno para cada equipo de laboratorio y tipo de prueba.

---

[1] PRESSMAN, Roger. Ingeniería del Software, un enfoque práctico. Quinta edición. Madrid 2002. p.3.

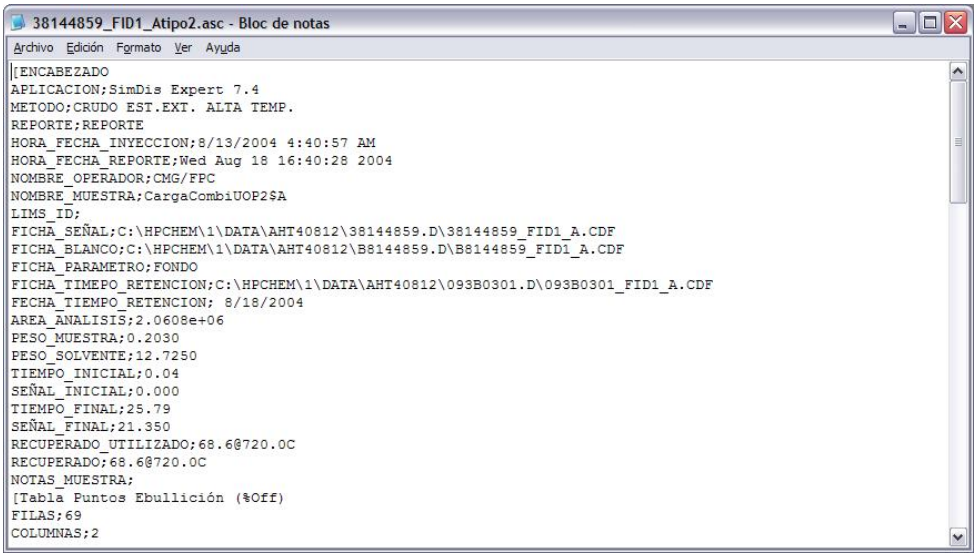
Este software procesa los archivos ASCII recibidos del equipo de laboratorio, para generar el archivo ARE que posteriormente es interpretado por el monitor ARE de SILAB. El código fuente de los programas creados en lenguaje IDXL, queda en archivos texto con extensión .IDX

SILAB realiza un escáneo periódico cada minuto, consultando los archivos ARE creados por el software IDM/SEEDPAK2, y transfiriéndolos a la base de datos ORACLE. Los archivos transferidos son renombrados con la extensión .DON, identificando así a los archivos transferidos con éxito.

3.3 SOBRE LOS ARCHIVOS ASCII

Un laboratorio posee muchos instrumentos que realizan varios ensayos a diferentes muestras y cada ensayo genera un tipo de archivo ASCII. La cantidad de registros de los archivos ASCII y su estructura varía dependiendo del Laboratorio, del Instrumento y del Ensayo que lo genera. La siguiente figura ilustra un ejemplo de uno de estos archivos texto. Ver Figura 1. (Ejemplo de un archivo ASCII generado por un instrumento de pruebas).

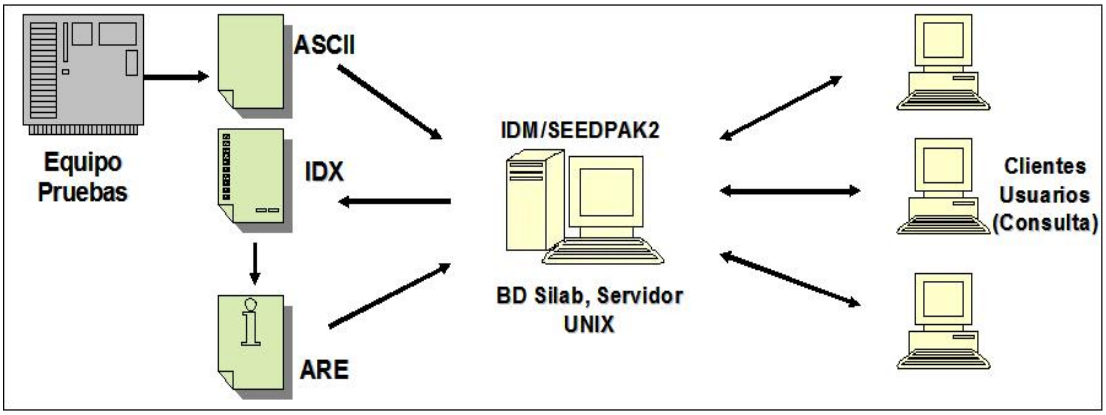
Figura 1. Ejemplo de un archivo ASCII generado por un instrumento de pruebas.



Los archivos ASCII varían por cada tipo de prueba, tanto en el contenido como en la forma en que están dispuestos los datos. Se analizaron diferentes tipos de archivos para encontrar diferencias y similitudes que nos permitieran contemplar un diseño flexible del sistema, para que la aplicación pudiese ser utilizada en otros equipos y laboratorios.

A partir del archivo ASCII, el usuario del laboratorio por medio de una interfaz IDM/SEEDPAK2, utiliza programas IDX configurados por el usuario, importa el archivo plano. Posteriormente se genera un nuevo archivo con una estructura diferente y con una extensión ".are", estructura en que deben ser enviados los datos al Sistema de Información de Laboratorios, a través de FTP (Protocolo de Transferencia de Archivos) a un servidor UNIX. Ver Figura 2 (Proceso actual para la transferencia de archivos).

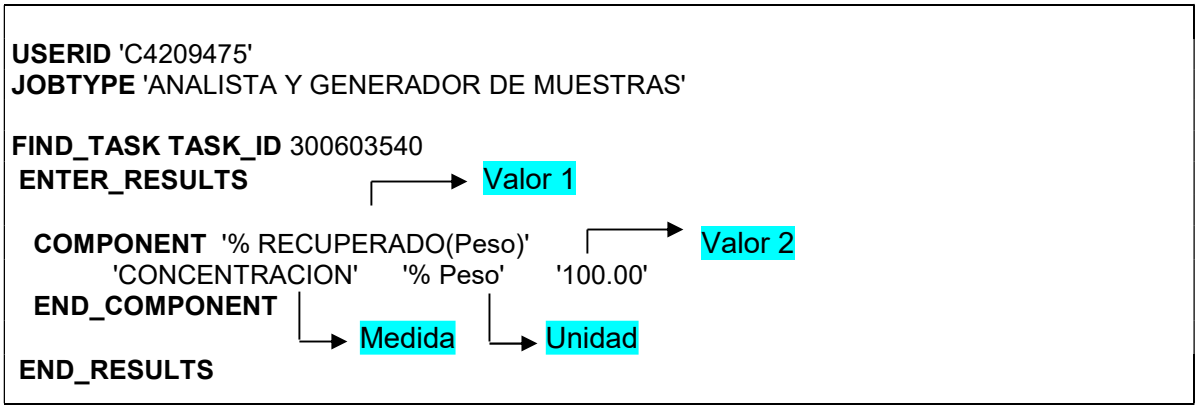
Figura 2. Proceso actual para la transferencia de archivos.



3.4 SOBRE LOS ARCHIVOS ARE

Los archivos .ARE son el resultado de la manipulación de los archivos ASCII. Estos archivos mantienen una estructura fija para que el sistema SILAB pueda interpretar estos datos y subirlos a la base de datos de los laboratorios.

Estructura de archivos ARE.



Los campos **USERID** y **JOBTYPE** corresponden al número de identificación y el cargo del usuario. El campo **TASK_ID** identifica la tarea en que se está trabajando en ese momento en el Laboratorio. Dentro de las palabras **ENTER_RESULTS** y **END_RESULTS** se almacenan componentes constituidos por cuatro partes. Los **Valores 1 y 2** son los resultados generados por las diferentes pruebas; valores que pueden tener asociadas una **Medida** y una **Unidad.** De acuerdo con los resultados del ensayo, estos archivos pueden presentar uno o más componentes por archivo.

3.5 ELEMENTOS DEL SOFTWARE A CONSTRUIR

Para dar solución al problema propuesto en este Proyecto, se debe considerar la naturaleza del problema, así como el dominio de la información del software, sus funciones requeridas, su comportamiento, rendimiento e interconexión. Entre los elementos que lo conformarían están un componente de adquisición de datos que recopile y dé formato a la información recibida del entorno externo, un componente de interacción que permita al usuario seleccionar información de acuerdo con lo requerido por cada una de las pruebas, y un componente de control de salida que responda al entorno externo.

La herramienta de conectividad se encargará de tomar los datos arrojados por el equipo de laboratorio y permitirá la manipulación de la información requerida por cada tipo de prueba para posteriormente transferir el resultado de la selección en un archivo .ARE al Sistema de Información de los Laboratorios.

4. CONCEPTOS BÁSICOS

Al analizar varias alternativas para la manipulación de caracteres o de archivos ASCII abordamos el paradigma de los analizadores léxicos, expresiones regulares y otros avances teóricos que cada una de estas alternativas aporta a la manipulación de caracteres.

Decidimos estudiar el tema de los analizadores léxicos y expresiones regulares considerando la variedad en las estructuras de los archivos planos generados por los diferentes instrumentos de prueba con que cuenta el Instituto. El estudio de estos archivos nos llevó a contemplar el uso de palabras y/o expresiones reservadas que nuestro programa de manipulación de caracteres debería identificar en el archivo original, con base en esta identificación extraer los datos de interés, ya fueran las mismas palabras y/o expresiones u otras cercanas a ellas, con unos parámetros de cercanía previamente determinados en una base de datos auxiliar que diseñamos para este fin.

4.1 ARCHIVOS ASCII

Archivos que utilizan solamente caracteres del estándar ASCII; estos archivos también denominados tipo texto o archivos planos, contienen código estándar americano para intercambio de información (American Standard Code for Information Interchange) es el formato más común para archivos de texto en computadores e Internet. Los sistemas operativos basados en UNIX y DOS (excepto Windows NT) usan el ASCII para sus archivos de texto.

4.2 ANALIZADOR LÉXICO (AL)

Un analizador Léxico o AL[2], se encarga de buscar los componentes léxicos o palabras que componen una fuente, según unas reglas o patrones. La entrada del AL podemos definirla como una secuencia o cadena de caracteres. El AL tiene que dividir la secuencia de caracteres en palabras con significado propio y después convertirla a una secuencia de terminales desde el punto de vista del analizador sintáctico, que es la entrada para los analizadores sintácticos. [3]

4.3 EXPRESIONES REGULARES

Una expresión regular [4] es un lenguaje representado por un grupo de caracteres y metacaracteres que ofrece gran poder y flexibilidad en el procesamiento de texto. A través de la expresión de coincidencia de patrones es posible interpretar rápidamente vastas cantidades de texto y ejecutar varias operaciones, como buscar información, buscar patrones, eliminar texto, extraer valores y mucho más. Las expresiones regulares son una representación de un lenguaje. Una expresión regular establece un patrón que verifica que una palabra pertenece a un lenguaje. No se trata de lenguajes de programación ni de idiomas: se trata de un conjunto de símbolos. [5]

---

[2] REVELLES, Jorge. Análisis de Léxico. Departamento de Ingeniería de Software. Universidad de Granada. España. 2004. http://giig.ugr.es/~jrevelle/docencia/pl/ como archivo PDF: Tema02PL.pdf. Pagina visitada Agosto de 2004.

[3] GÁLVEZ ROJAS, Sergio. Análisis Lexicográfico,.Docente Titular de Escuela Universitaria. Traductores Compiladores e Interpretes. 23 de Junio de 2001. www.lcc.uma.es/~galvez/ftp/tci/ como archivo PDF: Tictema2.pdf. Pagina visitada Agosto de 2004.

[4] GALUPPO, Fabio. Expresiones Regulares. Artículo publica originalmente en www.universalthread.com/spanish/magazine, traducido en www.microsoft.com/spanish /msdn/comunidad/mtj.net/voices/art101.asp. Pagina visitada en Octubre de 2004.

[5] RODRIGUEZ, Daniel. Expresiones Regulares - Conceptos Avanzados. Danirc. www.ibiza-beach.com/, 18/07/2001 como PDF: bulma-736.pdf. Pagina visitada Octubre de 2004.

PATRÓN

Un patrón es una cierta secuencia de caracteres que podemos estar buscando en una cadena. El patrón es seleccionado por el usuario en la forma de una expresión regular y este patrón se compara con todo el contenido de la cadena. El concepto de patrón es simple. Puede ser un patrón recurrente o puede ocurrir sólo una vez en diez mil cadenas de datos.[6] Antes de comenzar a buscar un patrón específico recurrente, se puede pensar en las partes del patrón que son redundantes. Con frecuencia, esto nos puede guiar a seleccionar una expresión regular que pueda encontrar la mayoría de veces a todas las ocurrencias de patrones que necesitamos buscar. Por ejemplo, si el patrón siempre comienza y termina con la misma secuencia de caracteres, pero siempre difiere en la mitad, entonces su expresión regular debe buscar al principio y al final del patrón, mientras ignora todo lo del medio. [7]

4.4 INTERFAZ DE BASE DE DATOS

Interfaz de Base de Datos o DBI (-*Data Base Interfaz*-) es un módulo para el acceso a bases de datos, es decir, mediante DBI podremos acceder a bases de datos con nuestros scripts o programas desarrollados. DBI define un conjunto de funciones, variables y convenciones que ofrecen una interfaz de base de datos consistente e independiente de la base de datos que se esté utilizando, es decir, DBI es la interfaz estándar para base de datos de un lenguaje determinado, lo cual no significa que no haya otros, pero, normalmente todo lo que se puede hacer con otro módulo (que no use DBI para acceder a bases de datos), se puede hacer con DBI, de forma más fácil y portable. DBI es independiente de la base de datos con

---

[6] BABOOM Software. Perl en Español. Expresiones Regulares. 2004. perlenespanol.baboonsoftware.com/archives-tut/000072.html. Pagina visitada en Octubre de 2004.

[7] WYKE, R. Allen, DONALD B. Thomas. Fundamentos de Programación en Perl. Traducción Gustavo Elías Fonsenca, Osborne – McGraw Hill. Primera Edición Bogotá. 2002.

la que se está trabajando, lo cual significa que podremos trabajar con bases de datos como Oracle, Sybase, Informix, MySql, bases de datos con soporte ODBC (MS-Access, SQL Server), etc. Por el momento DBI sólo trabaja con bases de datos relacionales y no con bases de datos orientadas a objetos.

4.5 MANEJADOR DE BASE DE DATOS

Un Manejador o Driver de Base de Datos DBD (Data Base Driver) se encarga de llevar las peticiones realizadas por nuestro programa (usando DBI) en una base de datos específica. Existe un módulo DBD para cada tipo de base de datos, dicho módulo se encarga de pasar las peticiones que realizamos en DBI a peticiones a la base de datos sobre la que estamos trabajando. Para trabajar con una base de datos determinada nos hace falta tener controlado el módulo DBD correspondiente; por ejemplo, para trabajar con la base de datos Oracle nos hace falta el módulo DBD-Oracle. Para trabajar con bases de datos en Perl sólo nos hace falta tener instalado Perl, la base de datos con la que vamos a trabajar, el módulo DBI y el módulo DBD para la base de datos instalada.

5. HERRAMIENTAS

En el desarrollo de nuestra aplicación se trabajó sobre las herramientas Perl para el tratamiento de caracteres, Visual Basic 6 para el desarrollo de la interfaz de usuario, y todo esto soportado bajo el gestor de base de datos Oracle.

Perl; *Practical Extraction and Reporting,* (lenguaje práctico de extracción y reportes) fue la herramienta escogida entre otras herramientas estudiadas como Java[8] y C++.[9] Este lenguaje trata interfaces de entrada común (CGI), ofrece facilidades para procesador de archivos, creador de Scripts de MS Windows de Interfaces Gráficas de Usuario (Graphical user Interfaces, GUI), acceso a bases de datos, entre otras. A diferencia de otros lenguajes, Perl se basa en expresiones regulares en algún momento, ya sea para verificar si alguna variable tiene un valor deseado, o para sustituir una palabra por otra. De esta manera contamos realmente con su máxima extensión para la creación de uno de nuestros objetivos: manipulación de archivos tipo texto basándonos en las expresiones regulares[10].

Gracias a la facilidad de aprendizaje de este lenguaje, se trabajó lo siguiente para el desarrollo de la aplicación:

- Manejo de datos escalares.
- Estructuras de control.
- Programación estructurada: Datos temporales y datos privados. Flujos de programas o subrutinas.

---

[8] FROUFE Agustín. Tutorial de Java, Características de Java. 1 de Febrero de 1997. http://www.cica.es/formacion/JavaTut/Intro/carac.html. Pagina visitada en Julio de 2004.

[9] ORTIZ M.C. BERNABÉ, Herbert. Lenguaje C++. La Paz, Baja California Sur. 20 de Junio del 2001. http://www.itlp.edu.mx/posgrado/lengprog/c.htm. Pagina visitada en Julio de 2004.

[10] WYKE, DONALD. Op. cit., p.

- Manejo de archivos y directorios.
- Comprensión, manejo y creación de expresiones regulares.
- Interacción con bases de datos por medio del DBI (Interfaz de Base de Datos) de Perl.
- Conexión a una base de datos MYSQL con el DBD (Manejador de la Base de datos).
- Conexión con MS Visual Basic
- Importación de subrutinas desde los paquetes.
- Importación de paquetes y módulos.

Las siguientes herramientas de desarrollo fueron estudiadas para encontrar solución a la manipulación de texto. Ver Tabla 1.

Tabla 1. Evaluación de Herramientas de Desarrollo

C++	PERL	JAVA
Es un lenguaje compilado de nivel medio, que combina elementos de lenguaje de alto nivel con la funcionalidad del lenguaje ensamblador.	Es un lenguaje interpretado que tiene varias utilidades, es orientado a la búsqueda, extracción y formato de archivos de tipo texto.	Es un lenguaje compilado simple que ofrece la funcionalidad de un leguaje potente reduciendo errores comunes de programación.
Apoya eficientemente la programación orientada a objetos soportando herencia y polimorfismo.	Permite la programación orientada a objetos utilizando conceptos como: herencia, encapsulación y polimorfismo.	Excelente apoyo a la programación orientado a objetos, trabaja sus datos como objetos y con interfaces de esos objetos, soportando las tres características del paradigma de la orientación de los objetos: encapsulación, herencia y polimorfismo
Lenguaje relativamente pequeño; se puede describir en poco espacio y aprender rápidamente.	Lenguaje relativamente fácil de aprender útil y versátil	Lenguaje no tan fácil de aprender pero es una herramienta poderosa de programación.
La manipulación de caracteres de este lenguaje se hace a partir de arreglos y sus funciones lo que obliga un poco más de trabajo par la extracción y manipulación de caracteres.	Posee un mecanismo de búsqueda de patrones y expresiones regulares extraordinariamente potente haciendo un lenguaje poderoso para el tratamiento de caracteres.	No existe un tipo de datos primitivo que sirva para la manipulación de cadenas de caracteres. En su lugar se utilizan clases ya definidas. Esto significa que en Java las cadenas de caracteres son, objetos que se manipulan como tales.

Incluye archivos llamados librerías. Las librerías contienen el código objeto de muchos programas que permiten hacer cosas comunes.	Utiliza paquetes (conjunto de funciones agrupadas dentro de un solo archivo) y módulos a los que nos referimos globalmente como bibliotecas.	Proporciona librerías y herramientas para que los programas puedan ser distribuidos.
Complejidad al implementar protocolos de transmisión de datos FTP y HTTP.	Facilita la implementación de protocolos de transporte como FTP, HTTP.	Tiene la capacidad de interconexión TCP/IP y existen librerías de rutinas para acceder e interactuar con protocolos como HTTP y FTP.
La conexión a base de datos se realiza con ODBC (Conectividad a Base de Datos Abiertas). [11]	La conexión a la base de datos utiliza Interfaz de Base de Datos "DBI" y el Manejador de Base de Datos "DBD". API de fácil implementación y utilización. [12]	La conexión y accesos a base de datos mediante JDBC (Conexión Base de Datos de Java). [13]

5.1 PERL (Practical Extraction and Reporting Language)

Mucho antes de que Java o Javascript invadiera Internet y los escenarios circundantes o que la Web naciera, Perl tenía una fuerte presencia en su comunidad. Desde automatizar la administración de tareas de UNIX, hasta realizar el análisis sintáctico rutinario de los archivos, Perl se ha usado como un verdadero lenguaje utilitario.

Perl significa lenguaje práctico de extracción y reportes (Practical Extraction and Reporting Language), ya se trate de la interfaz de entrada común (CGI), procesamiento de archivos, creación de Scripts de Windows de Microsoft, Interfaces Gráficas de Usuario (Graphical User Interfaces, GUI), acceso a bases de datos, etc.

Perl es uno de los lenguajes de programación más antiguos de aquellos que se discuten comúnmente para uso en la Web. Sus orígenes se remontan a 1987 y

---

11 ORTIZ. BERNABÉ, Op. cit., p. 1.
12 BABOOM, Op. cit., p. 1.
13 GALUPPO, Op. cit., p. 1.

desde entonces, Perl ha ofrecido cuatro versiones principales más y dos semi mayor. En la actualidad, Perl se precia de tener un grupo grande de programadores y se puede encontrar lealtad dentro del compacto grupo de usuarios, comparables con la de los usuarios de Linux, OS/2 y Mac OS. [14]

5.1.1 El comienzo

Larry Wall, el autor del lenguaje, liberó Perl por primera vez en el grupo de Usenet Comp.Sources, el 18 de octubre de 1987. Este nuevo leguaje, que fue distribuido gratuitamente por el señor Wall, se derivó del lenguaje de programación C y más tarde recibió la influencia de lenguajes como Basic, awk, sed y shell de UNIX; Perl tomó lo mejor de estos y los integró como un solo lenguaje funcional. La gente con poca o ninguna experiencia en programación es capaz de aprenderlo rápidamente y de comenzar a programar en Perl. Además de la facilidad para aprenderlo, Perl sencillamente es un lenguaje útil. Desde el comienzo, Perl tuvo una capacidad increíble para manipular texto, archivos y procesos del sistema. Este enfoque libre, fácil, y útil le permite al lenguaje Perl tener éxito rápidamente. [15]

Aunque desarrollado originalmente en un entorno UNIX, actualmente hay versiones para casi todos los sistemas operativos; los scripts son compatibles entre las diversas plataformas, de forma que es un verdadero lenguaje multiplataforma. Muchos fabricantes lo incluyen en sus versiones de UNIX; también Linux lo incluye.

5.1.2 Perl 4

Detrás de Perl 1.0 vino Perl 2.0, que fue puesto en circulación el 5 de Junio de 1988. Por aquel tiempo, el número de programadores de Perl había crecido, lo

---

[14] GALUPPO, Op. cit., p. 1.
[15] Ibid., p. 1.

mismo que el uso del lenguaje. Diferentes tipos de programadores estaban usando Perl para sus tareas cotidianas y algunas personas de las principales corporaciones empezaron a promover el lenguaje.

Pasó un año y medio antes de que el mundo viese Perl 3.0, el cual fue puesto en circulación el 18 de octubre de 1989, unos dos años después de que la versión 1.0 entrara en la red. Con este lanzamiento, se distribuyó inicialmente Perl con la licencia pública general, LPG (General Public Licence, GPL) de GNU versión 1.0, la cual permite la distribución de software libre. Para entonces, Perl ya había despegado.

Miles de programadores estaban usándolo y la Web, que estaba en su fase inicial, realmente le dio un fuerte empujón a Perl. Perl había sido usado ampliamente por administradores de UNIX, pero rápidamente se estaba convirtiendo en la norma para escribir scripts CGI (Common Gateway Interface) para procesar datos de formularios enviados por Internet. En marzo de 1991, se puso en circulación Perl 4.0 con la GPL, así como la nueva Licencia Artística Perl (Perl Artistic Licence). Perl 4 fue la última versión vista en tres años y medio, mientras que el lenguaje era sometido a una completa reelaboración. Sin embargo, esto no significa que no se hubiese progresado durante aquellos años. De hecho, este fue el tiempo en el que Perl empezó a sobresalir y a ser reconocido. En enero de 1992 Matthias Neerache puso en circulación la primera versión de Perl para Macintosh (MacPerl 4.0.2). A finales del mismo año, MacPerl llegó a la versión 4.0.5, la cual incluía soporte para administración de base de datos (DBM) y conectores (sockets). 1993 fue testigo del trabajo continuando en MacPerl y en la versión final del núcleo del lenguaje Perl 4, Perl 4.036. La versión final de MacPerl, 4.1.8, la cual implementaba 4.036, pronto siguió a las anteriores. [16]

---

[16] . WYKE. Op. cit., p.

5.1.3 Perl 5

Para este tiempo, la comunidad Perl ya estaba lista para una nueva versión del lenguaje. Estaban apareciendo de repente implementaciones por todas partes, el acceso a base de datos estaba disponible y los scripts adicionales prefabricados eran abundantes. Sin embargo, era el momento para una nueva versión del lenguaje.

En octubre de 1994 se puso en circulación el Perl 5. Esta versión traía muchas mejoras y llevó el lenguaje a un siguiente nivel. Perl 5 fue la primera entrega que empujó el una vez más sencillo lenguaje para crear scripts, más allá de las tareas administrativas simples, a convertirse en el más potente faro.

5.1.4 Soporte de plataformas

El número de plataformas soportadas es una de las razones por las cuales Perl ha tenido tan buena acogida. Aunque Perl se diseñó originalmente sobre y para plataformas UNIX, no transcurrió mucho tiempo antes de que fuese soportado para otras plataformas populares. Algunas de las implementaciones eran versiones únicas basadas en el código fuente que se hicieron trabajar en un sistema operativo particular. Otros llevaron la funcionalidad más allá de la distribución y agregaron operaciones específicas a un sistema. En conclusión, sin embargo, Perl ha mantenido un conjunto bastante consistente de funcionalidad a través de estos sistemas y, aunque existen las diferencias, en muchos casos un script para un sistema puede ejecutarse en otro.

Al hablar sobre las plataformas soportadas, las cuales se refieren a las versiones del lenguaje para Windows, UNIX y Mac OS, son implementaciones, o binarios precompilados de Perl y a veces tienen sus propios sistemas de numeración de versiones.

Windows

Originalmente existieron dos versiones de Perl para la plataforma de 32 bits de Windows. Una era Perl para Win32 de ActiveState. Luego con Perl 5.004, la distribución estándar de Perl incluyó soporte para la plataforma de 32 bits de Windows. Las dos tienen muchas similitudes porque buena parte de la distribución estándar estaba basada en el trabajo de ActiveState; a pesar de esto existían algunas diferencias. Desde entonces, éstos códigos base se han mezclado y el resultado fue ActivePerl que es mantenido por ActiveState.

UNIX

UNIX es la plataforma en donde todo empezó. De hecho, a medida que se aprende el lenguaje de Perl se pueden ver órdenes y rasgos provenientes de UNIX, profundamente caracterizados dentro del lenguaje. Esto no es del todo malo, porque, si UNIX es excelente en algo, es en lo que puede lograrse con un simple script de shell o con el acceso a la línea de comandos. [17]

5.1.5 Expresiones regulares en Perl

En principio esto puede hacerse usando los métodos del objeto string, pero el problema surge cuando no tenemos una subcadena fija y concreta, sino que queremos buscar un texto que responda a un cierto esquema, como por ejemplo: buscar aquellas palabras que comienzan con http: y finalizan con una \, o buscar palabras que contengan una serie de números consecutivos, etc.; es en estos casos cuando tenemos que utilizar las expresiones regulares. Todos los programadores de Perl han usado las expresiones regulares en algún momento, ya sea para verificar si alguna variable tiene un valor deseado, o para sustituir una

---

[17] WYKE. Op. cit., p.

palabra por otra. Pero aún así son contados los programadores que realmente entienden en su máxima extensión esta gran creación. Las expresiones regulares existen en todos los lenguajes de programación desde JAVA y C++ hasta PHP y Perl. En todas las sintaxis son muy similares, exceptuando ciertos cambios en algunos caracteres. [18]

5.1.6 Funcionamiento de Perl

Primero, y sobre todo, se debe entender que Perl se comporta como un lenguaje interpretado, no compilado como C o C++. Esto significa que los programas Perl no se compilan en binario (1s y 0s) antes de que puedan correrse. De hecho, como programador, no se compila del todo programas Perl; simplemente se crea un archivo de texto con su código Perl.

Los programas Perl convierte su código en código de bytes (lo cual se hace antes de ejecutarlo) y luego sí se ejecuta. Cuando un programa corra sobre Windows 2000, Mac OS 9 y Red Hat Linux, se compila en cada uno de estos sistemas. Por el contrario, Perl simplemente permite ejecutar el mismo script en cada sistema. [19]

5.2 VISUAL BASIC 6

Visual Basic 6 es un programa considerado como un sistema de objetos interactuando entre sí, es decir, su programación es orientada a objetos (OOP). Los ambientes de desarrollo visuales facilitan aún más la construcción de programas y solución de problemas, porque permiten abstraer al ingeniero de software de todo la interfaz gráfica del problema.

---

[18] GALUPPO. Op. cit., p. 1.
[19] WYKE. Op. cit., p.

Todo problema, aún aquellos sencillos de información, se consideran y resuelven como módulos de código gigante que contienen todo el código necesario (variables, procedimientos, funciones, interfaces, etc.) para solucionar el problema. En programación visual o interfaces con el usuario son generados por el compilador y el ingeniero de software sólo se concentra en resolver el problema planteado. Visual Basic, es un compilador que permite usar cualquiera de los tres enfoques en la solución de problemas de información que puedan y deban ser resueltos empleando el computador y el lenguaje. [20]

Para el propósito de este proyecto usamos el enfoque de programación en ambientes visuales usando este lenguaje de programación.

5.2.1 Historia

El lenguaje de programación BASIC (Beginner's All purpose Symbolic Instruction Code) nació en el año 1964 como una herramienta destinada a principiantes, buscando una forma sencilla de realizar programas, empleando un lenguaje casi igual al usado en la vida ordinaria, y con instrucciones muy sencillas y escasas. Teniendo en cuenta el año de su nacimiento, este lenguaje cubría casi todas las necesidades para la ejecución de programas. Los autores fueron los científicos John G. Kemeny (Budapest, 1926 – USA 1992) y Thomas E. Kurtz (Illinois 1928) Su trabajo original se llamó True Basic.

Con la aparición de los primeros ordenadores personales, dedicados comercialmente al usuario particular, allá por la primera mitad de los 80, el BASIC resurgió como lenguaje de programación pensado para principiantes, y muchos de estos pequeños ordenadores domésticos lo usaban como único sistema operativo (Sinclair, Spectrum, Amstrad). Con la popularización del PC, salieron varias

---

[20] VAQUERO S., Antonio. QUIROZ V., Gerardo. Microsoft Visual Basic 6, Manual del Programador. Microsoft Corporation. Mc-Graw Hill. Madrid. 1998. p.

versiones del BASIC que funcionaban en este tipo de ordenadores (Versiones BASICA, GW-BASIC), pero todas estas versiones del BASIC no hicieron otra cosa que terminar de rematar este lenguaje. Los programadores profesionales no llegaron a utilizarlo, teniendo en cuenta de las desventajas de este lenguaje respecto a otras herramientas (PASCAL, C, CLIPPER). El BASIC con estas versiones para PC llegó incluso a perder crédito entre los profesionales de la informática.

Las razones para ello eran lógicas:

- No era un lenguaje estructurado.
- No existían herramientas de compilación fiables.
- No disponía de herramientas de intercambio de información.
- No tenía librerías.
- No se podía acceder al interior de la máquina.
- Un largo etcétera de desventajas respecto a otros lenguajes de programación.

Tal fue ese abandono por parte de los usuarios, que la aparición del Quick-BASIC de Microsoft, una versión ya potente del BASIC, que corregía casi todos los defectos de las versiones pasó prácticamente inadvertida, a no ser porque las últimas versiones del sistema operativo MS-DOS incluían una versión de Quick-BASIC algo recortada (Q-Basic) como un producto mas dentro de la amplia gama de ficheros ejecutables que acompañan al sistema operativo, y aprovecha de él el editor de textos (Cada vez que se llama al EDIT estamos corriendo el editor del Q-Basic). Esta versión del popular BASIC ya es un lenguaje estructurado, lo que permite crear programas modularmente, mediante subrutinas y módulos, capaz de crear programas ya competitivos con otros lenguajes de alto nivel. Sin embargo llegaba tarde, pues los entornos MS-DOS estaban ya superados por el entorno gráfico Windows.

Sin embargo algo había en el BASIC que tentaba a superarse: su gran sencillez de manejo. Si a esto se le añade el entorno gráfico Windows, el aprovechamiento al máximo de las posibilidades de Windows en cuanto a intercambio de información, de sus librerías, de sus drivers y controladores, manejo de bases de datos, etc. el producto resultante puede ser algo que satisfaga todas las necesidades de programación en el entorno Windows. La suma de todas estas cosas es VISUAL - BASIC. Esta herramienta conserva del BASIC de los años 80 únicamente su nombre y su sencillez, y tras su lanzamiento al mercado, la aceptación a nivel profesional hizo borrar por fin el "mal nombre" asociado a la palabra BASIC.

En el año 2001 se comenzó a comercializar la versión 6.0 de este producto. Desde su salida al mercado, cada versión supera y mejora la anterior. Dados los buenos resultados a nivel profesional de este producto, y el apoyo prestado por el fabricante para la formación de programadores, Visual-Basic se ha convertido en la primera herramienta de desarrollo de aplicaciones en entorno Windows. [21]

5.2.2 ¿Que es Visual Basic?

Es un lenguaje de programación que se ha diseñado para facilitar el desarrollo de aplicaciones en un entorno grafico (GUI-GRAPHICAL USER INTERFACE), Visual Basic 6, nos proporciona un juego completo de herramientas que facilitan el desarrollo rápido de aplicaciones.

La palabra Visual hace referencia al método que se utiliza para crear la interfaz grafica de usuario (GUI). En lugar de escribir numerosas líneas de código para describir la apariencia y la ubicación de los elementos de la interfaz, simplemente se puede agregar objetos preconstruidos en su lugar dentro de la pantalla.

---

[21] VAQUERO. QUIROZ, Op. cit., p.

La palabra Basic hace referencia al lenguaje BASIC (Beginners All-Porpuse Symbolic Instruction Code), un lenguaje que es muy utilizado por los programadores de la informática o computación. Primero fue GW-BASIC, luego se transformó en QuickBASIC y actualmente se lo conoce como Visual Basic y la versión más reciente es la 6, esta versión combina la sencillez del BASIC con un poderoso lenguaje de programación Visual que juntos permiten desarrollar robustos programas de 32 bits para Windows.

La ventaja principal de este lenguaje de programación es su sencillez para programar aplicaciones de cierta complejidad para Windows, y sus desventajas son la necesidad de archivos adicionales además del ejecutable y cierta lentitud en comparación con otros lenguajes. Hoy en día este último factor es cada vez menos determinante debido a la gran potencia de los ordenadores de última generación. [22]

5.2.3 Características Generales de Visual Basic

Visual-Basic es una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan en una gran parte a partir del diseño de una interfaz gráfica. En una aplicación Visual Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interfaz gráfica.

Es por tanto un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Combina ambas tendencias. Ya que no podemos decir que VB pertenezca por completo a uno de esos dos tipos de programación, debemos inventar una palabra que la defina: PROGRAMACIÓN VISUAL[23]. La creación de un programa bajo

---

[22] Suárez Bernardo, Luis. Curso de Visual Basic. www.jrubi.com. 6 de abril 2003. www.telecable.es/personales/jrubi/index.htm?curso.htm. Pagina visitada octubre de 2004.

[23] VAQUERO. Op. cit., p.

Visual Basic lleva los siguientes pasos:

Análisis: Es el estudio de las necesidades que han dado origen a la creación de ese programa. Es lo que se llama *Análisis* de la aplicación. Es la primera fase tuvimos en este programa y en muchos casos es la más olvidada entre los programadores. La aplicación no se inicio con el teclado, sino sobre un papel.

Creación de un interfaz de usuario: La interfaz será la principal vía de comunicación hombre máquina, tanto para salida de datos como para entrada. Fue necesario partir varias ventanas - Formularios - a las que les añadimos los controles necesarios.

Definición de las propiedades de los controles: Se dio la forma, posición, y todas las características necesarias a los controles que es colocado en la aplicación. Estas propiedades determinarán la forma estática de los controles, es decir, como son los controles y para qué sirven.

Generación del código asociado a los eventos que ocurran a estos controles: A dar respuestas a eventos (click, doble click, una tecla pulsada, etc.) le llamamos procedimiento en visual basic y son generados de acuerdo a las necesidades del programa.

Generación del código del programa. Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto.

Sin embargo, VB ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código se introducen en unos bloques llamados Módulos, en otros bloques llamados Funciones y otros llamados Procedimientos. Estos Procedimientos no responden a un evento llamado por un control o

formulario, sino que responden a un evento producido durante la ejecución del programa. [24]

[24] VAQUERO. Op. cit., p

6. DESARROLLO DEL SISTEMA

El proceso de desarrollo está basado en **Modelo Lineal Secuencial** en cascada con iteraciones de ajustes integradas. Ver Figura 3 (Modelo Lineal Secuencial en cascada con iteraciones). Este modelo sugiere un enfoque sistemático secuencial, para el desarrollo del software que comienza en un nivel de investigación y progresa con el análisis, diseño, generación de código, pruebas y mantenimiento.

Esta metodología de cascada realmente sufre un cambio al desarrollarse en fases, generando tareas en forma concurrente con un grado moderado de ajustes mientras se hace revisión y refinamiento, en cada cambio de las etapas de este modelo.[25]

Figura 3. Modelo Lineal Secuencial en cascada con iteraciones.

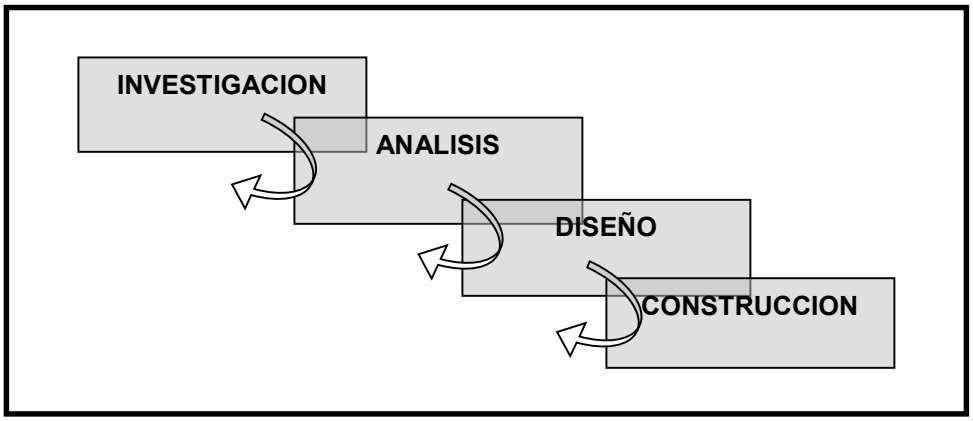


Figura adaptada de RUBLE [26].

[25] RUBLE, A. David. Análisis y Diseño Practico para Sistemas Cliente-Servidor con GUI, México: Prentice Hall, 1998. p 10 -11.

[26] RUBLE, p.11. Figura 1-3.

Las siguientes fases básicas de ingeniería se llevaron a cabo para el desarrollo de este proyecto.

Fases básicas de la ingeniería:

- **Investigación y Conocimiento del Problema.**
- **Análisis.**
- **Diseño.**
- **Generación de código.**
- **Pruebas y mantenimiento.**

6.1 INVESTIGACIÓN Y CONOCIMIENTO DEL PROBLEMA

El ICP presenta un problema para el tratamiento de datos para determinados laboratorios, motivo por el cual nació este proyecto. Por medio de visitas realizadas al Instituto se investigó el proceso de generación de datos y su transferencia, en el Laboratorio de Resistencia de Materiales y otros laboratorios como: Catálisis, Cromatografía, Geoquímica – Ambiental, Espectrometría. En estos laboratorios se recogió información de los archivos creados por instrumentos de ensayos, así como también información de la manipulación de los datos para así comprender la naturaleza del problema.

6.2 ANÁLISIS

El análisis es un método, plan o procedimiento de clasificación para hacer algo, este proceso determina *Qué se necesita hacer*, antes de decidir *Cómo* debe hacerse. El conocimiento adquirido de los procesos en los diferentes laboratorios, nos dio a conocer la magnitud del problema para dar a conocer el comportamiento

del sistema a implementar, estableciendo interfaces de usuario, restricciones de diseño y qué criterios de validación, definiendo requerimientos claves del sistema.

6.2.1 Proceso de Adquisición de Requerimientos

Los requerimientos se deben descubrir antes de empezar a construir un sistema o producto, y que puede ser algo que el producto debe hacer o una cualidad que el producto debe tener. Un requerimiento existe ya sea porque el tipo de sistema demanda ciertas funciones o cualidades, o porque el cliente quiere que ese requerimiento sea parte del producto final. En la definición de los requerimientos del sistema se tratan de comprender todas las necesidades del usuario. En primer lugar, empezamos a especificar el comportamiento externo del sistema desde el punto de vista del usuario para crear una arquitectura general del sistema.

A través de entrevistas con el Coordinador del Laboratorio de Resistencia de Materiales, un operador de este laboratorio y el DBA (Administrador de Base de Datos) del Instituto, logramos conocer y recopilar los requerimientos necesarios para el desarrollo de la aplicación y generar un informe de requerimientos aprobados por usuario final. En tabla 2 (Requerimientos del sistema), se presentan los requerimientos de la aplicación.

Tabla 2. Requerimientos del sistema.

REQUERIMIENTOS	CONSIDERACIONES
Ubicar el directorio en donde se encuentran los archivos para importar.	El directorio en donde se encuentran los archivos será definido por el usuario fácilmente y modificable por cada laboratorio que vaya a hacer uso de esta aplicación.
Ubicar el archivo que genera el instrumento, para importarlo desde la aplicación.	El archivo deberá tener permisos de lectura. La extensión de los archivos generados deben ser .TXT o .ASC, archivos con diferente extensión se importaran como documentos planos (*.txt).
Importar, desde la aplicación, los archivos que se encuentran en el disco duro generados por un instrumento de un laboratorio dado.	Los archivos generados no deben ser manipulados por ningún usuario o software antes de que se importen a la aplicación. El archivo no debe estar vació.

REQUERIMIENTOS	CONSIDERACIONES
Clasificar los archivos por tipos, según el laboratorio, instrumento y ensayo.	Se clasificarán los archivos por tipos, según el Laboratorio, Instrumento, ensayo de donde provengan y su formato: Ej. Tipo1 – archivos del instrumento Espectrobac del laboratorio Caracterización de materiales y del ensayo de análisis de elementos químicos. Y se generalizará si existen tipos de archivos iguales. El sistema se encargará de detectar según los registros configurados por este tipo de archivo, los datos necesarios a extraer y transferir. En la base de datos de la aplicación se tendrá una breve descripción del tipo de archivo ASCII que se importa a la aplicación.
Identificar los datos del archivo para su extracción.	Identificar el formato del archivo importado. El contenido de los archivos puede estar separado por comas, comillas, espacios en blanco o cualquier otro caracter que se considere especial. Tener en cuenta que para un mismo archivo, es posible que un tipo de dato se encuentre delimitado por comillas dobles y otro por comillas sencillas. Tener en cuenta los caracteres especiales “> y <” que pueden ser utilizados para identificar valores aproximados en los datos. Identificar el nombre del operador o usuario y sus delimitadores.
Enviar los datos a los campos de la base de datos.	Los datos extraídos del archivo plano se almacenarán en una base de datos auxiliar y podrán ser seleccionados y enviados en cualquier momento al sistema de base de datos de los laboratorios. La base de datos auxiliar se creará independientemente de las base de datos del Sistema de Información de los Laboratorios. La aplicación no permitirá la modificación o eliminación de datos del archivo original.
Permitir al usuario seleccionar los datos relevantes para transferencia, del conjunto de datos extraídos del archivo plano.	Al menos un dato debe ser seleccionado por el usuario para transferirlo al sistema de información de laboratorios. No a todos los archivos generados se seleccionan de datos para ser transferidos.
Mostrar una previsualización de los datos a enviar.	Los datos ya seleccionados son los que se mostrarán en pantalla.
Transferir los datos a la Base de Datos de los laboratorios.	Para el diseño de la transferencia, es necesario conocer las tablas y los campos que recibirán estos datos en la Base de Datos de Laboratorios.

REQUERIMIENTOS	CONSIDERACIONES
	Los datos a transferir se presentarán en un archivo *.are, que posteriormente se enviará a través del protocolo de transferencia de archivos FTP a un servidor Unix; este servidor contiene un monitor de archivos que se encargará de subir los datos a la base de datos de los laboratorios.

6.2.2 Modelo de proceso de Adquisición de Requerimientos

El siguiente modelo nos brindo una vista del proceso, una secuenciación aproximada y general de las actividades que realizamos y obtuvimos un documento con los requerimientos de la aplicación. Ver figura 4 (Modelo de Proceso de Adquisición de Requerimientos).

Figura 4. Modelo de Proceso de Adquisición de Requerimientos.

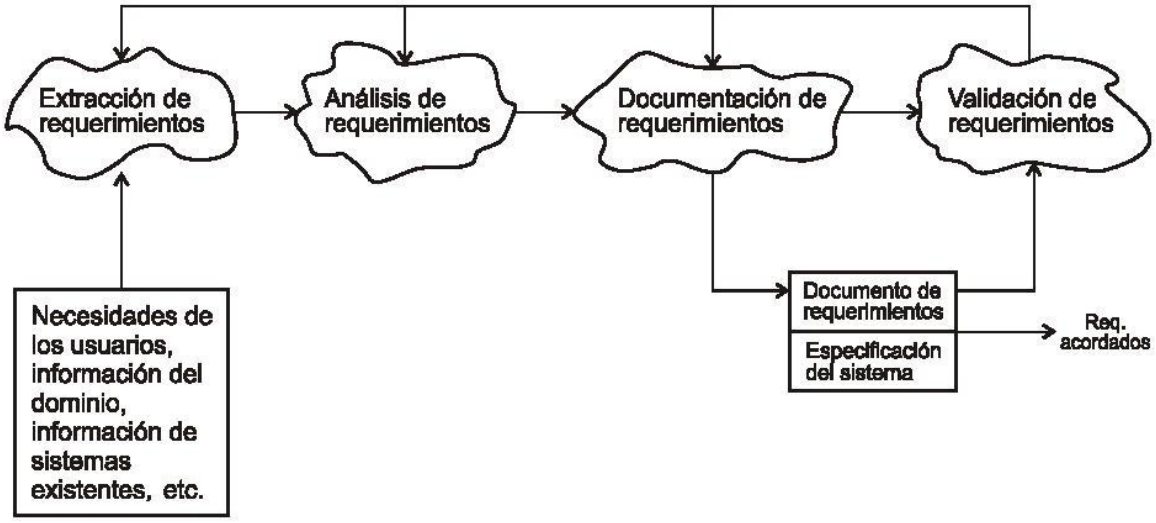


Figura adaptada de DAVYT[27].

[27] DAVYT Dávila, Nicolás, Ingeniería de Requerimientos, Universidad Ort. Uruguay, Facultad de Ingeniería, webs.montevideo.com.uy/nicolasd, como PDF. Requerimientos.pdf, P.8. 2001, web visitada Septiembre de 2004.

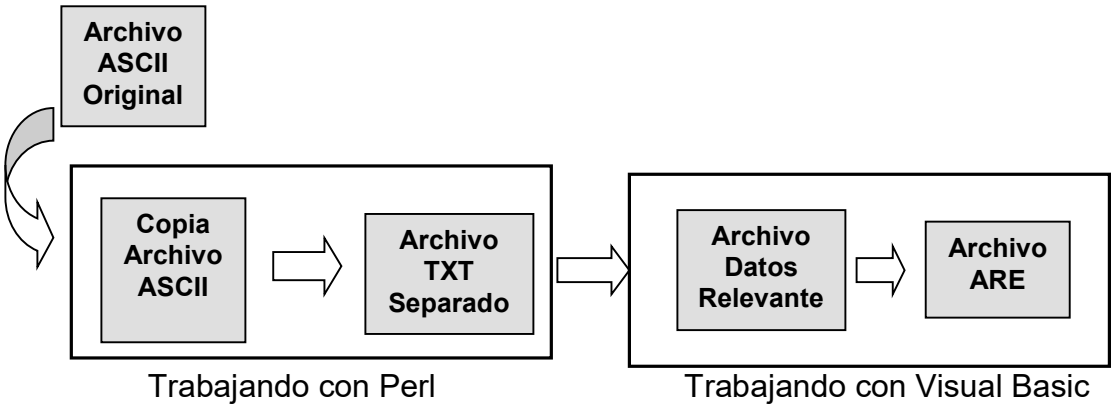
6.3 DISEÑO

El diseño de sistemas se define como el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un sistema, con suficientes detalles como para permitir su interpretación y realización física. En la fase de diseño se determina la arquitectura general del sistema y su comportamiento dinámico, adaptando la especificación realizada en la etapa anterior. En esta fase se establece el comportamiento dinámico del sistema, es decir, cómo debe reaccionar ante los acontecimientos. El resultado obtenido de la etapa de diseño nos facilitó enormemente la implementación de la aplicación, pues proporciona la estructura básica del sistema y cómo los diferentes componentes actúan y se relacionan entre ellos.

Para el diseño de la interfaz de usuario, se tuvo en cuenta la comunicación de la interfaz con otros elementos del sistema (el módulo en Perl para manejo de archivos planos y la base de datos Oracle). Además se contemplaron las consideraciones necesarias en el diseño de la base de datos, para proporcionar una herramienta flexible que facilite su implementación posterior para la transferencia de resultados generados por otros equipos de laboratorio en el Instituto.

El software de transferencia automática (TRANSFER, nombre dado al software en este proyecto) consta de varios programas desarrollados en las herramientas Perl, Visual Basic 6 y base de datos Oracle. Este software procesa los archivos ASCII generados por el equipo del Laboratorio, dando como resultado un nuevo archivo con extensión “*.are” que posteriormente será transferido a través de FTP a un servidor UNIX, donde es interpretado por el monitor ARE del Sistema de Información de Laboratorios.

Figura 5. TRANSFER.



En la figura 5, el programa creado en Perl obtiene una copia del archivo ASCII original, y desglosa los datos según las palabras reservadas y separadores (Archivo Separado) que fueron previamente configurados, en la aplicación, de acuerdo con el tipo de archivo. Estos datos se almacenan en un archivo nuevo (Archivo TXT Relevantes). Posteriormente este archivo es el que toma como entrada el programa de Visual Basic para generar el archivo "*.are".

El diseño del sistema presenta tres grandes modelos, modelo de contexto, modelo de eventos y modelo de información. Estos modelos se crean juntos, en forma iterativa. La veracidad de cada modelo depende de la integridad de los otros dos.

6.3.1 Modelo de Contexto

El modelo de contexto nos define el alcance de este sistema. Como se muestra en el Anexo 1 (Diagrama de Contexto). Contiene un círculo que muestra el sistema propuesto como un gran proceso. Los cuadros que están alrededor muestran a las personas, instrumentos y otros sistemas que tendrán que comunicarse con el nuevo sistema. Las flechas de entradas y salidas muestran el flujo de datos que

estimulan al sistema para ponerlo en acción y las salidas del sistema en forma de una respuesta a su entorno.[28]

6.3.1.1 Propósito del Modelo de Contexto**.** El diagrama de contexto o diagrama de flujo de datos, se ve simple en su primer nivel. (Ver Anexo 1). Tiene sólo una burbuja en el centro que representa el sistema de transferencia de datos "TRANSFER". La notación clásica de diagramas de flujo de datos se usa para mostrar todos los flujos de estímulos que entran al sistema (Archivo ASCII y datos que ingresa el usuario) y sus flujos de respuesta (Archivo ARE) que regresan a su entorno. Los agentes que son externos al sistema (INSTRUMENTO, USUARIO Y SILAB) se muestran como cuadros. Representan a los originadores de flujos de estímulos y/o los destinos de flujo de respuesta.

6.3.1.2 Diagrama de Flujo de Datos. Los diagramas de flujo de datos DFD, proporcionan una indicación de cómo se transforman los datos a medida que se avanzan en el sistema y representan las funciones que transforman el flujo de datos. Los DFD proporcionan información adicional que se usa durante el análisis y nos sirvan como base para el modelado de la información. Los DFD que se muestran en los anexos se presentan en niveles y nos muestran los procesos generales del sistema de transferencia de datos. Anexo 1.

6.3.1.3 Notación de Diagramación de Flujo de Datos o DFD. En el modelo contextual utilizamos la notación de diagramación de flujo de datos clásica (Ver Figura 6. Notación de diagramación de flujo de datos). Los DFD son modelos que muestran la ruta que toman los datos a través de una organización, sin ninguna tendencia a causa de una implementación específica[29].

---

[28] Ruble. Op. cit., p. 55 – 56.

[29] Ibid., p. 56 – 59.

Figura 6. Notación de diagramación de flujo de datos

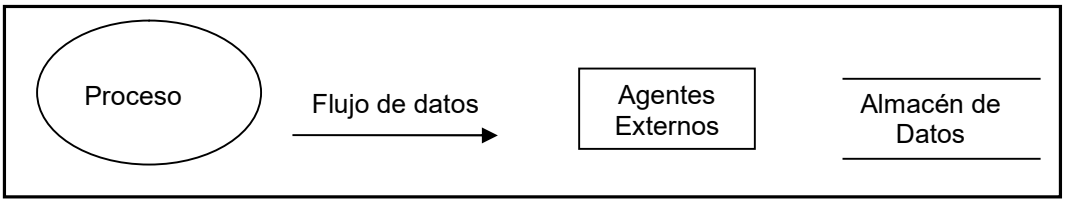


Figura de Ruble [30].

- **Procesos:** Las reglas convencionales de diagramación insisten en que un proceso debe ser nombrado con un verbo seguido por el objeto al que se aplica la acción.

- **Agentes Externos:** Cada parte interesada que está en el ambiente alrededor del sistema y que interactúa con éste se muestra como un rectángulo en el diagrama de contexto. Un agente externo puede ser una persona, un instrumento u otro sistema.

- **Flujo de Datos:** Compuestos por atributos de datos individuales, agrupados en paquetes de información sobre flujo que se transporta de un proceso a otro. Cada vez que uno de estos paquetes es entregado al sistema se requiere que el sistema o el proceso reaccione en una forma predecible. Esta reacción puede incluir la emisión de una respuesta, la cual también es un paquete de información compuesto de atributos individuales.

- **Almacenes de Datos:** Los almacenes de datos son lugares del sistema en donde se encuentran los datos cuando no se utilizan. Se les muestra como líneas paralelas. Realmente se pueden representar como bases de datos, archivos, memorias de computadoras o hasta memoria humana. [31]

---

30 RUBLE, Op. cit., p. 57. Figura 3-2.

31 Ibid., p. 57- 63.

6.3.2 Modelo de Eventos

El propósito del modelo de eventos es describir el comportamiento de un sistema. Esto se logra haciendo un listado de todos los eventos del proyecto ante los cuales está planeado que la aplicación debe responder. Para cada evento de la lista se crea una entrada en el diccionario de eventos, la cual se detalla por medio de una tabla la definición, actividad, respuesta. El diccionario de eventos nos dice la manera en que se espera que la aplicación se comporte cuando sucede el evento.

El modelo de eventos establece la actividad del usuario en relación con el negocio en términos en que el usuario pueda comprender fácilmente. La lista de eventos describe a la aplicación desde la perspectiva del usuario. Para el diseñador de la interfaz, el modelo de eventos proporciona las justificaciones para la navegación y el contenido de las pantallas. Los botones y clic de ratón que son codificados, son una implementación directa de los eventos del negocio en el modelo.

El diccionario de eventos es el lugar principal en donde se descubrirán las mismas políticas y reglas que aparecen en diferentes eventos que conducen a identificar y extraer componentes de software en el diseño interno.

¿Que es un evento?

El modelado de eventos es una forma para determinar todas las cosas que suceden en el mundo real y que deben causar que el sistema entre en acción y haga algo. Para establecer que un evento la sintaxis es sujeto-verbo-objeto. Alguien (sujeto) hace algo (verbo) a algo (objeto). El evento sucede en un momento específico en el ambiente, dentro del agente externo, la ocurrencia del

evento está completamente bajo el control del agente externo. El sistema no puede causar la ocurrencia del evento. [32]

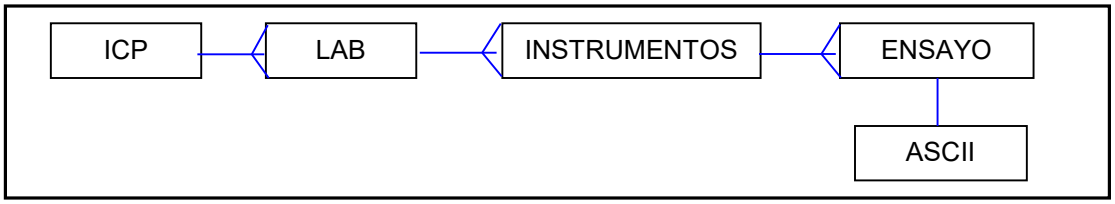
6.3.2.1 Lista de Eventos. La lista de eventos es exactamente eso, una lista de los eventos ante los cuales está planeado que la aplicación debe responder. La lista de eventos describe a cada evento por un nombre con una sintaxis (sujeto-verbo-objeto). [33]

Cada evento necesita ser reconocido en nuestro modelo como un objeto discreto el cual puede estar relacionado con otros objetos más tradicionales del modelo, tales como las entidades, las ubicaciones y los procesos. La lista de eventos también necesita ser ordenada y afinada en varias formas para quesea verdaderamente útil. A continuación presentamos la lista de eventos de nuestra aplicación.

Lista de eventos de nuestra aplicación.

El Instituto Colombiano de Petróleo ICP, tiene varios laboratorios, cada uno de los cuales utiliza instrumentos para realizar ensayos o análisis, cuyos resultados son generados en archivos ASCII. Esto se representa en la Figura 7. (Diagrama generación de archivos ASCII).

Figura 7. Diagrama generación de archivos ASCII



[32] . RUBLE. Op. cit., p. 81.
[33] Ibid., p. 85.

Los datos de los archivos ASCII deben ser finalmente transferidos al Sistema de Laboratorios del ICP, SILAB, y ese es el objetivo principal de este proyecto: construir una interfaz software que permita y facilite esta transferencia, independientemente de la estructura del archivo ASCII empleada por cada instrumento de laboratorio para generar los resultados.

- **El usuario inserta nombre y contraseña para ingresar a la aplicación**

Con base en esta información se validará la existencia del usuario en SILAB (tabla NAI_USERS). Si el usuario existe, se consultará su perfil: coordinador o analista. (NAI_USERS.jobtype). Si el perfil del usuario es coordinador, se habilitarán las opciones para mantenimiento de las tablas de configuración de la Aplicación, de acuerdo con el laboratorio al que pertenezca el usuario.

- **El usuario selecciona el laboratorio, instrumento y ensayo**

El usuario escoge de una lista desplegable, el laboratorio, instrumento y ensayo a los que corresponden los resultados que desea transferir.

- **El usuario selecciona el archivo ASCII a importar**

Prerrequisito: Previamente a este evento, el usuario debe haber colocado en un directorio específico, los archivos ASCII (con extensión .txt o .asc) con resultados pendientes por transferir.

Si no se encuentran archivos en el directorio especifico, el usuario ubicará el archivo, con la ayuda de un browser o explorador de archivos proporcionado por la aplicación.

Ubicados los archivos ASCII a importar, la Aplicación procederá a realizar el cargue de esta información, que consistirá en identificar y extraer del archivo los datos relevantes para la transferencia y subirlos a su base de datos de la aplicación.

Esta identificación y extracción dependerá del laboratorio, instrumento y tipo de ensayo, que son las características que determinan la estructura del archivo ASCII, es decir, cómo están distribuidos en el archivo los resultados de las pruebas. La Aplicación contemplará un diseño flexible que permita la inclusión posterior de nuevas estructuras de archivos ASCII conteniendo resultados de otros tipos de ensayos, a nivel de mantenimiento de tablas, es decir, evitando manipulación del código fuente por el usuario.

El usuario selecciona el tipo de transferencia que desea realizar: directa o con selección previa.

- **Transferencia directa.** La Aplicación transferirá todos los resultados encontrados en el archivo ASCII, sin intervención del usuario.

  - **Transferencia con selección previa.** La Aplicación desplegará en pantalla los resultados encontrados en el archivo y permitirá al usuario realizar un trabajo de selección de los resultados que desea transferir (este es el caso, por ejemplo, de ensayos realizados en el Laboratorio de Caracterización de Materiales).

En la ventana de selección se desplegará un encabezado con información tomada en forma automática del archivo plano. Después del encabezado, desplegará el detalle de los resultados, un resultado en cada línea, y para su selección permitirá marcarlos uno a uno, o marcar o desmarcar todos a la vez. El usuario sólo podrá

seleccionar datos para transferir. En ningún momento tendrá opción de modificar o adicionar información a la obtenida desde el archivo ASCII.

- **El usuario selecciona la opción de transferir los datos a SILAB**

La transferencia consistirá en la creación de un archivo (.are), siguiendo los requerimientos para la interfaz con el sistema de información SILAB, y en su envío al servidor de archivos UNIX mediante FTP. Una vez en el servidor, otro programa que está monitoreando permanentemente el servidor, se encargará de tomar los archivos .are y de subir su información a SILAB.

Si la transferencia es exitosa, serán enviados a un directorio especifico (directorio determinado por el encargado de laboratorio) los archivos ASCII transferidos y serán borrados del directorio en que se encontraban antes de la transferencia, evitando que se procese dos veces el mismo archivo. Se permite el re-procesamiento si se requiere.

Si se establece la conexión con el servidor, se desplegará en pantalla un mensaje de notificación de envío, y se facilitará al usuario continuar con la transferencia de otros resultados. De lo contrario, se permitirá reintentar la conexión hasta que el usuario lo decida. La Aplicación no guardará los archivos .are creados; estos serán almacenados en el servidor UNIX en donde se realizará el monitoreo de los .are.

- **El usuario con perfil de Administrador, hace mantenimiento a las tablas de Configuración de la Aplicación**

Cada laboratorio podrá definir una serie de parámetros (que permitirán a la Aplicación establecer valores por defecto y realizar acciones en forma automática), y las estructuras de los archivos ASCII asociados a cada instrumento y tipo de

ensayo, con base en lo cual se podrá realizar la identificación y extracción automática de los datos de estos archivos.

6.3.2.2 Diccionario de Eventos. La lista de eventos es de muy poco valor para el analista o para el diseñador sin el diccionario de eventos. Las entradas del diccionario de eventos para cada evento definen su importancia en la aplicación y sus componentes. [34]

Partes del diccionario de eventos:

- **Identificador (id):** El identificador del evento puede ser numérico, pero no hay que hacer que ese número sea significativo de alguna manera. El (id) asignado en forma aleatoria permite que se cambie el nombre del evento sin tener que cambiar su identificador.

- **Nombre del evento:** Es una oración clara del evento en palabras que el usuario puede comprender. Está especificada en la sintaxis (sujeto-verbo-objeto) cada vez que es posible.

- **Descripción del evento:** Informa en términos claros y simples, cuáles son las políticas del negocio para el evento. Si los usuarios no leen más que la descripción, deberán ser capaces de captar la esencia de su actividad dentro de la aplicación.

- **Actividad:** La actividad ocurre dentro del sistema. Este es el procesamiento que el sistema debe hacer para convertir la entrada en una respuesta adecuada para el evento. La actividad es una mini especificación del proceso.

---

[34] RUBLE. Op. cit., p. 86.

- **Respuesta:** Identifica los datos requeridos por el usuario para lograr el efecto deseado en el ambiente.

6.3.3 Modelo de Información

El modelo de información contiene el mapa estático de los datos que se deben almacenar en la aplicación. Este modelo influye en el diseño de base de datos e impacta en todos los aspectos de la aplicación. Las técnicas del modelo de información incluyen el diagrama entidad-relación, la definición de atributos y el diagrama de transición de estados. [35]

6.3.3.1 Descripción de entidades relaciones y atributos.

- **Entidades:** Una entidad es un sustantivo, además puede representar una idea del mundo real. Es una cosa que tiene una existencia individual definida en la realidad o en la mente.
  Para la ingeniería de software la definición de una entidad es, una persona, cosa o idea abstracta sobre la que el sistema necesita recordar algo. Las instancias de cada entidad tienen características similares y se comportan de manera parecida. Las entidades son representadas gráficamente como un rectángulo.

- **Relaciones:** Las entidades se asocian constantemente con otras entidades. A estas asociaciones se les llama relaciones. Una relación es una línea que conecta una entidad con otra. Las relaciones piden, muestran, almacenan, devuelven y definen las conexiones relevantes entre entidades. Los nombres de las relaciones son importantes debido a que son capaces de expresar mucho de las políticas y el significado del flujo cuando

---

[35] RUBLE. Op. cit., p. 117.

son nombradas adecuadamente. Dichas relaciones tienen la posibilidad de ser opcionales u obligatorias y la cardinalidad puede ser definida 1-1, 1-M y M-M entre las entidades.

- **Atributos:** El tercer componente principal del modelo de información son los atributos, que representan a todos los elementos de datos del sistema. Cada hecho acerca de una entidad constituye un atributo por separado. Los atributos definen las propiedades de un objeto de datos y pueden usar; primero, para nombrar una ocurrencia del objeto de datos, segundo, describir la ocurrencia, o tercero para hacer referencia a otra ocurrencia en otra tabla. Además, uno o varios atributos se definen como un identificador, es decir, como llave o clave cuando queremos encontrar una instancia del objeto de dato. [36]

6.3.3.2 Diccionario de datos. El modelo de análisis acompaña representaciones de objetos de datos, funciones y control. En cada representación los objetos de datos o elementos de control cumplen con un papel importante. Por ello, es necesario proporcionar un enfoque organizado para representar las características de cada objeto de datos y elementos de control. Esto se realiza con el diccionario de datos. El diccionario de datos es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, salidas, de los componentes de los almacenes. [37] Ver Anexo 5. Diccionario de Datos.

6.3.3.3 Tablas de la Aplicación en Oracle. Ver Anexo 5. Reporte de Entidad sus Atributos. Diccionario de Datos.

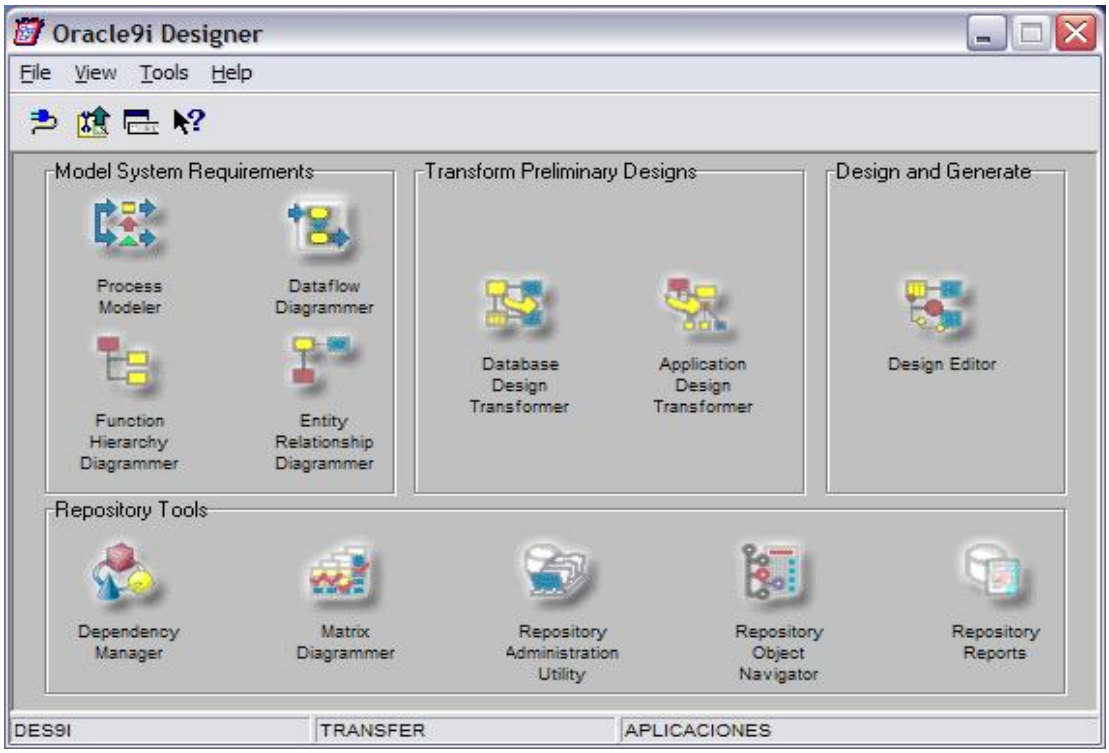
---

[36] RUBLE. Op. cit., p. 121 - 129.
[37] PRESSMAN. Op. cit., p. 215.

6.3.3.4 Diagrama entidad-relación. El diagrama entidad-relación (DER) es el elemento grafico principal del modelo de información. Está compuesto de las entidades acerca de las cuales la aplicación necesita recordar hechos específicos y las relaciones que existen entre estos grupos de hechos. El Diagrama entidad relación del sistema fue asistido bajo la herramienta Oracle Designer 9i. Ver Anexo 3. Diagrama Entidad Relación de Sistema TRANSFER.

6.3.3.5 Oracle9i Designer. Oracle9i Designer es una herramienta de ORACLE que tiene como objetivo soportar el ciclo de vida de un sistema de información, permitiendo la construcción de sistemas de información en forma automatizada bajo el esquema cliente/servidor. Ver Figura 8 Oracle 9i Designer.

Figura 8. Oracle 9i Designer.



Oracle9i Designer es una herramienta que disminuye los tiempos en los desarrollos de las aplicaciones, permitiendo un control sobre los desarrolladores,

unificando estándares de programación y de presentación, ofreciendo confiabilidad para el usuario. Esta herramienta permite analizar las necesidades de una aplicación y diseñar soluciones de sistemas completos, conceptualizando las necesidades de información del negocio y transformándolas en una solución computacional apropiada.

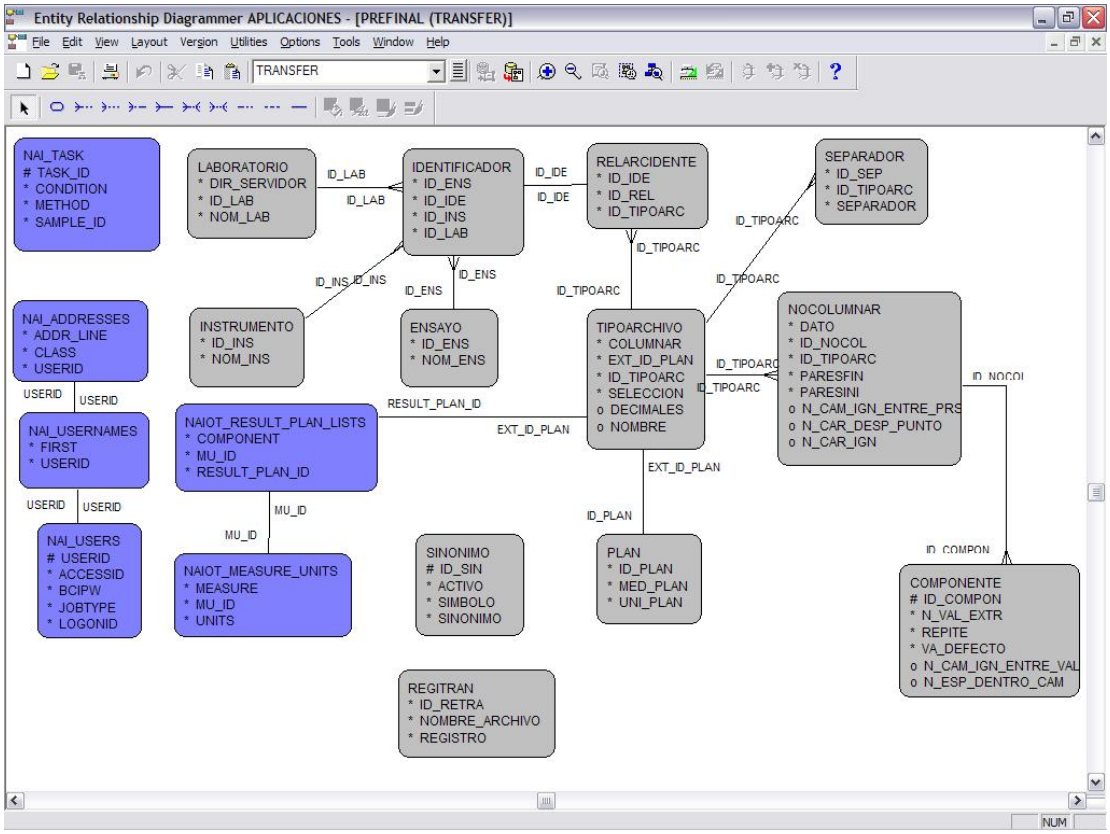
Este diseñador no hace cumplir una metodología específica, sino incluye la ayuda al usuario al desarrollo rápido de aplicaciones, a realizar ingeniería de información y al modelo de procesos.

El ambiente iterativo del diseño y desarrollo proporcionado por Designer apoya al proceso de cambiar una aplicación después de generarlo, capturando los cambios en Designer y regenerando la aplicación mientras que se realizan los cambios. El modelado de los datos en este proyecto fue asistido bajo las siguientes utilidades que nos facilita la herramienta Oracle 9i Designer.

Diseñador de Diagramas Entidad Relación

Esta utilidad se encuentra en el primer nivel de Designer; Modelado de requisitos del sistema. Permite representar los requerimientos de información del sistema gráficamente. El diseñador de diagramas Entidad Relación o Entity Relationship Diagrammer permite definir las entidades con sus respectivos atributos y relaciones. Ver Figura 9. DER TRANSFER.

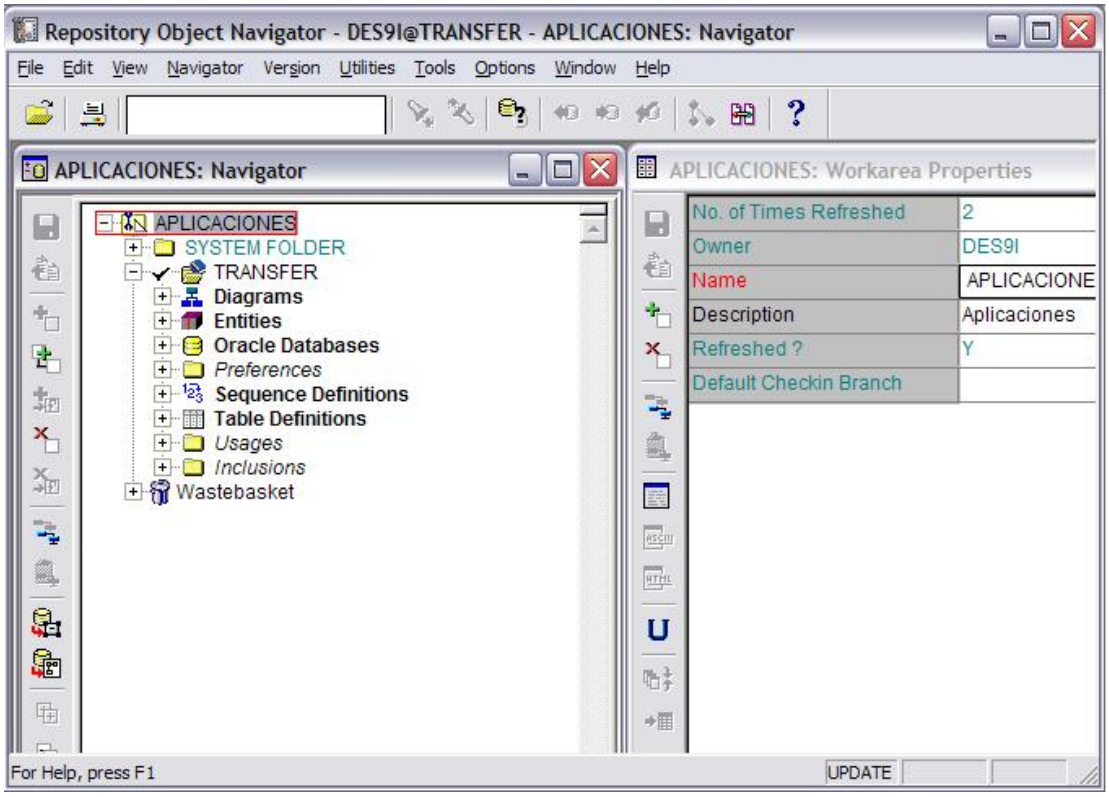
Figura 9. DER TRANSFER.



Repositorio De Objetos

El Repositorio de Objetos o Repository Object Navigator da una apreciación global de los objetos que se utilizan y se pueden utilizar para el modelado de aplicaciones. Es una herramienta universal que se puede usar en el modelado de procesos, modelado de sistemas, diseño de sistemas e implementación. Ver Figura 10. Repositorio De Objetos.

Figura 10. Repositorio de Objetos



SQL*PLUS

Esta herramienta de Oracle Designer permite trabajar con el lenguaje estándar SQL (Lenguaje de consulta estructurado por sus siglas en ingles Structured Query Language), donde se utilizan comandos para acceder los datos de la base de datos de la aplicación [38]. Ver Figura 11. (SQL * PLUS.) El lenguaje Sql consta de sentencias que permiten:

- Consultar datos de la Base de Datos
- Agregar, modificar y remover datos
- Crear, modificar y remover estructuras de datos

---

38 SANGUINO, Sandra. Guia 1 SQL. Universidad Autónoma de Bucaramanga. Agosto 2004. fis.unab.edu.co\docentes\ssanguin como: guia1.pdf. Pagina visitada en Noviembre de 2004. p.1.

- Regular el acceso de datos.

Sentencias de Manipulación de Datos (DML)

- Insert (Seleccionar datos de registros y columnas de una o más tablas).
- Update (Cambiar los datos en una tabla).
- Delete (Borrar los datos de una tabla).
- Commit (Confirmar una transacción).
- Rollback (Deshacer los cambios realizados después de un commit).

Sentencias de Definición de Datos (DDL)

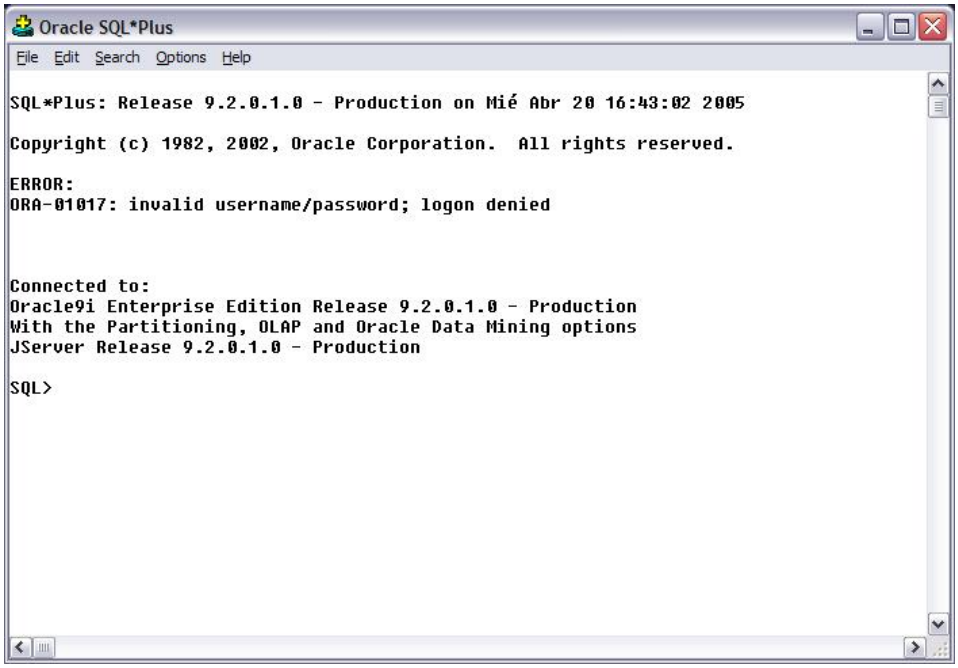
- Create (Crear objetos).
- Alter (Alterar objetos).
- Drop (Eliminar objetos).
- Rename (Renombar objetos).

Esta herramienta reconoce y ejecuta sentencias SQL. El SQL*PLUS no es una extensión del SQL. Las sentencias SQL son almacenadas una a la vez en el buffer SQL y pueden ser editadas usando comandos SQL*PLUS. [39]

---

[39] SANGUINO, Sandra. Guía General. Como pdf: guiag.pdf. Op. cit., p. 1 – 12.

Figura 11. SQL * PLUS.



6.3.3.6 Transformación del Diagrama Entidad Relación a Tablas. Cuando se tiene creado el DER en el diseñador de diagramas Entidad Relación, el siguiente paso es realizar una transformación preliminar de tablas con la herramienta **Database Design Transformer** de Oracle Designer. Ver Figura 12 (Database Design Transformer). Cuando se realiza este proceso Designer genera un reporte de las entidades, atributos, relaciones, llaves principales y foráneas del DER. Ver Anexo 6. (Scripts SQL de Oracle). Una vez generado este archivo se prosigue a generar las tablas físicamente en la base de datos utilizando la herramienta **Design Editor** de Oracle Designer. Ver Figura 13 (**Design Editor**). Esta aplicación genera tres archivo (*.sql, *.tab, *.con) y un DLL (Lenguaje definición de datos). El archivo SQL se ejecuta en el SQL*PLUS creando las tablas en la base de datos o simplemente se corre el archivo DLL, sin necesidad de ingresar al SQL*PLUS ejecutando comandos SQL creando la tabla en la base datos.

Figura 12. Database Design Transformer

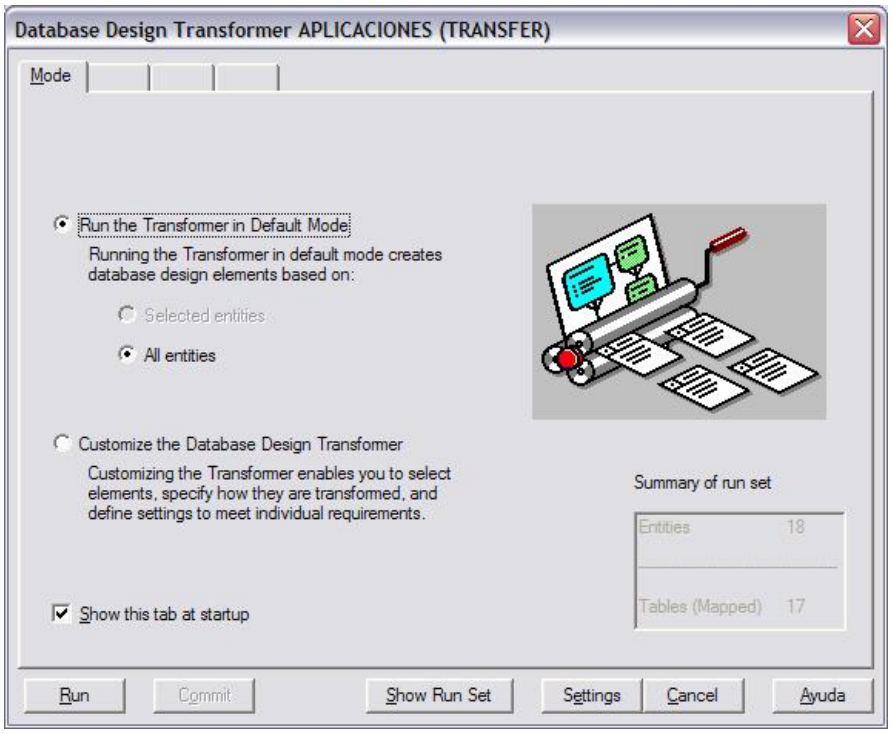
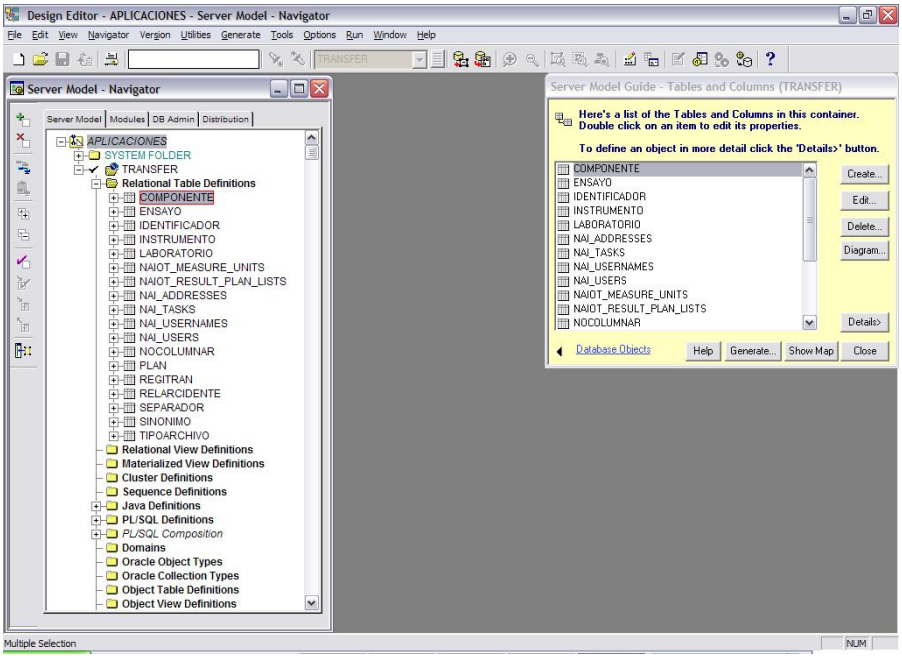


Figura 13. Design Editor.



6.4 GENERACIÓN DE CÓDIGO O CONSTRUCCIÓN

Bajo el lenguaje práctico de extracción y reportes Perl, se trabajo lo siguiente:

- Importar un archivo ASCII desde la aplicación.
- Identificar cadenas de caracteres en el archivo ASCII.
- Importar del archivo los datos necesarios para la creación del archivo "*.are".
- Importar los datos identificados y subirlos a una base de datos Oracle, por medio de conexión establecida con Visual Basic 6.

Bajo el lenguaje de programación Visual Basic 6 se ha desarrollado lo siguiente:

- Creación de la Interfaz de usuario.
- Conexión con Perl para la extracción de datos relevantes del archivo ASCII.
- Conexión con la base de datos Oracle 9i.
- Administración de datos para la configuración de la aplicación según los tipos de archivos ASCII.
- Selección de resultados de archivo ASCII.
- Creación de archivos "*.are", a partir de los resultados seleccionados del ASCII.
- Simulación de transferencia de los archivos "*.are" a través del protocolo FTP.

Para conocer el algoritmo por eventos de la aplicación y las interfaces de la aplicación, ver los Anexos 2 y 4 respectivamente.

6.5 PRUEBAS Y MANTENIMIENTO

La fase de mantenimiento contiene las fases de implantación, prueba, depuración y control de rendimiento, etapas que nos permitieron ajustar nuestro sistema de forma que sus funcionalidades correspondan con los objetivos iniciales.

La primera fase del mantenimiento del sistema consistió en la implantación de nuestra aplicación en el entorno en donde se va a desarrollar. Posteriormente, fue necesario realizar una serie de pruebas para comprobar que no surgieron problemas, en tal caso es recomendable realizar una depuración, finalmente, se debe mantener un control del rendimiento.

En el momento de implantar el sistema la primera cuestión a resolver consiste en la distribución del software. Visual Basic 6 proporciona la herramienta de empaquetado y distribución; Consiste en comprimir nuestros archivos en .CAB, conteniendo toda la información necesaria para que sean fácilmente instalables y, finalmente, distribuir los archivos mediante un archivo Microsoft Installer (Paquete Instalador). [40]

BENEFICIOS DE LA APLICACIÓN

- La selección de los datos podrá ser realizada en el mismo laboratorio y por las mismas personas que participaron en el ensayo, evitando el tratamiento de información por personal del Instituto.

- El registro de la información seleccionada se hará en forma automática, evitando así la digitación de información en el Sistema que suele ocasionar errores en los datos grabados.

---

[40] VAQUERO. Op. cit., p.

- Adicionalmente y como consecuencia de los anteriores beneficios, se podrá ofrecer oportunidad y calidad en la información, agilizando los procesos del laboratorio y de toma de decisiones.

DIFICULTADES EN DESARROLLO DEL PROYECTO

Durante el proceso de desarrollo se nos han presentado las siguientes dificultades:

- Dificultad para identificar correcta y oportunamente los requerimientos, en las entrevistas con el usuario.
- Dificultad para plasmar en un documento los requerimientos identificados.
- Estudio y selección de la herramienta de desarrollo para la manipulación de texto.
- Instalación y conexión de la interfaz de base de datos DBI y el Manejador de la base de datos con MySQL (DBD-mysql) y Oracle (DBD-oracle).
- Problemas técnicos con la conexión de Perl y Visual Basic 6.
- Problemas técnicos con la conexión de Perl y Oracle 9i.
- Inconvenientes al cargar un archivo ASCII a la aplicación cuando cambia su ubicación en el disco duro.

7. CONCLUSIONES

De acuerdo con los objetivos propuestos en el presente proyecto, se pueden presentar las siguientes conclusiones.

- La interacción permanente con el usuario es muy importante para las etapas de análisis y definición de requerimientos. Es necesario conocer el problema, detectar las necesidades, documentar las visitas, identificar los requerimientos, pero todo esto en un trabajo de equipo que requiere retroalimentación permanente del usuario, pues asumir y suponer es algo que definitivamente termina extendiendo considerablemente el tiempo programado para las actividades de diseño y construcción de la Aplicación.

- El acto de crear un buen modelo de contexto proporciona claridad y enfoque a las fronteras y responsabilidades del proyecto, las cuales contribuyen a controlar y medir el impacto de los cambios de alcance conforme se desarrolle el proyecto.

- Para el desarrollo de un futuro proyecto utilizando Designer/2002 hay que tener como factor crítico para el éxito del sistema, el modelo de Entidad-Relación, ya que para nosotros fue parte fundamental sobre el cual realizamos el Sistema de Transferencia (TRANSFER).

- Se debe aprovechar toda la documentación generada de la herramienta Designer/2002. Por esto lo más recomendable es que se documenten todos los objetos del repositorio a través de las etapas de desarrollo de la aplicación.

- Una de las principales razones de implementar en fases en un modelo lineal secuencial es el aprendizaje que se obtiene mientras se toma una parte del

proyecto a través de su ciclo de vida completo. En nuestro caso, nos proporciono experiencia valiosa que agilizó el desarrollo de fases posteriores.

- Un proyecto de desarrollo de Sistemas comprende varios componentes o pasos llevados a cabo durante la etapa del análisis, la cual ayuda a traducir las necesidades del cliente en un modelo de Sistema que utiliza uno más de los siguientes componentes: Software, hardware, personas, base de datos, documentación y procedimientos.

- La selección de la herramienta de desarrollo Perl y Visual Basic 6 fueron escogidas por la facilidad de reconocimiento de texto, el rápido aprendizaje de los lenguajes, por su facilidad de programación en ambientes visuales y su entorno de desarrollo cliente \ servidor.

8. RECOMENDACIONES Y TRABAJOS FUTUROS

- El sistema automático de transferencia de datos TRANSFER, funciona para los archivos ASCII generados por los siguientes laboratorios: Tecnología de Materiales, con el ensayo de composiciones aleaciones metálicas; Cromatografía con los ensayos de destilaciones simuladas y análisis gas de refinerías; y Espectroscopia, con el ensayo de detección de elementos. Está pendiente conocer y configurar la estructura de los archivos ASCII generados por otros laboratorios en donde se quiera implementar el Sistema.

- Poseer conocimiento técnicos de las herramientas de desarrollo de este proyecto para crear nuevas versiones del sistema TRANSFER.

- Capacitación al personal para el manejo del sistema automático de transferencia de datos.

9. REFERENCIAS BIBLIOGRÁFICAS

BABOOM Software. Perl en Español. Expresiones Regulares. 2004. http://perlenespanol.baboonsoftware.com/archives-tut/000072.html. Pagina visitada en Octubre de 2004.

CONCEPCIÓN NOVA, Pedro. Análisis y Diseño de Sistemas. Azua - República Dominicana. 30 de Marzo 2002. http://window.to/concepcion.com.do como archivo .DOC: anaydiseis.doc. Pagina visitada en Agosto de 2004.

DAVYT DÁVILA, Nicolás. Ingeniería de Requerimientos. Universidad Ort Uruguay. Facultad de Ingeniería. webs.montevideo.com.uy/nicolasd. como archivo .PDF: Requerimientos.pdf. Pagina visitada en Julio de 2004.

FROUFE Agustín. Tutorial de Java. Características de Java. 1 de Febrero de 1997. http://www.cica.es/formacion/JavaTut/Intro/carac.html. Pagina visitada en Julio de 2004.

GALUPPO, Fabio. Expresiones Regulares. Articulo publicado originalmente en http://www.universalthread.com/spanish/magazine, traducido en http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art101.asp. Pagina visitada en Octubre de 2004.

GÁLVEZ ROJAS, Sergio. Análisis Lexicográfico. Docente Titular de Escuela Universitaria. Traductores Compiladores e Interpretes. 23 de Junio de 2001. http://www.lcc.uma.es/~galvez/ftp/tci/ como archivo PDF: Tictema2.pdf. Pagina visitada en Agosto de 2004.

ORTIZ M.C. BERNABÉ, Herbert. Lenguaje C++. La Paz, Baja California Sur. 20 de Junio del 2001. http://www.itlp.edu.mx/posgrado/lengprog/c.htm. Pagina visitada en Julio de 2004.

PRESSMAN, Roger. Ingeniería del Software, un enfoque práctico. Madrid. Quinta edición. McGraw-Hill. 2002.

REVELLES, Jorge. Análisis de Léxico. Departamento de Ingeniería de Software. Universidad de Granada. España. 2004. http://giig.ugr.es/~jrevelle/docencia/pl/ como archivo .PDF: Tema02PL.pdf. Visitada en Agosto de 2004.

RODRÍGUEZ, Daniel. Expresiones Regulares - Conceptos Avanzados. Danirc http://www.ibiza-beach.com/. Julio de 2001 como archivo. PDF: bulma-736.pdf. Pagina visitada Octubre de 2004.

RUBLE A., David. Análisis y Diseño Practico para Sistemas Cliente-Servidor con GUI. Prentice Hall. México. 1998.

SANGUINO, Sandra. Guia 1 SQL. Universidad Autónoma de Bucaramanga. Agosto 2004. fis.unab.edu.co\docentes\ssanguin como pdf: guia1.pdf. Pagina visitada en Noviembre de 2004.

SUÁREZ B., Luis. Curso de Visual Basic. www.jrubi.com. 6 de abril 2003. www.telecable.es/personales/jrubi/index.htm?curso.htm. Pagina visitada en Octubre de 2004.

VAQUERO S., Antonio. QUIROZ V., Gerardo. Microsoft Visual Basic 6, Manual del Programador. Microsoft Corporation. Mc Graw Hill. Madrid. 1998.

WYKE R., Allen. Donald B., Thomas. Fundamentos de Programación en Perl. Traducción Gustavo Elías Fonsenca, Osborne – McGraw Hill. Primera Edición Bogotá. 2002.

ANEXOS

ANEXO 1. DIAGRAMA DE CONTEXTO.

Figura 14. Diagrama De Flujo De Datos Nivel 0

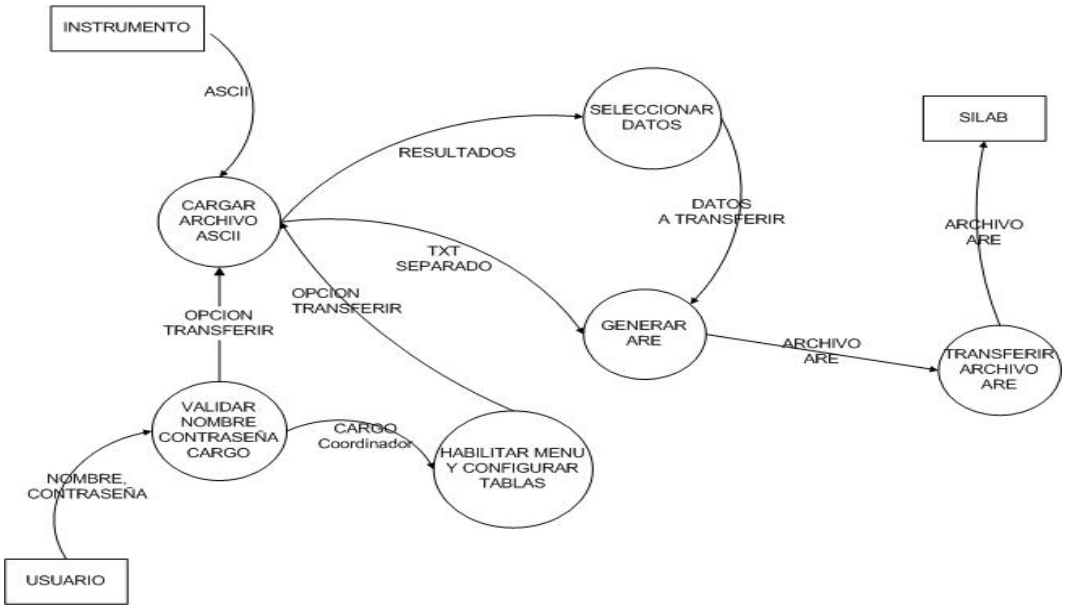


Figura 15. Diagrama De Flujo De Datos: Validación Nombre Y Contraseña

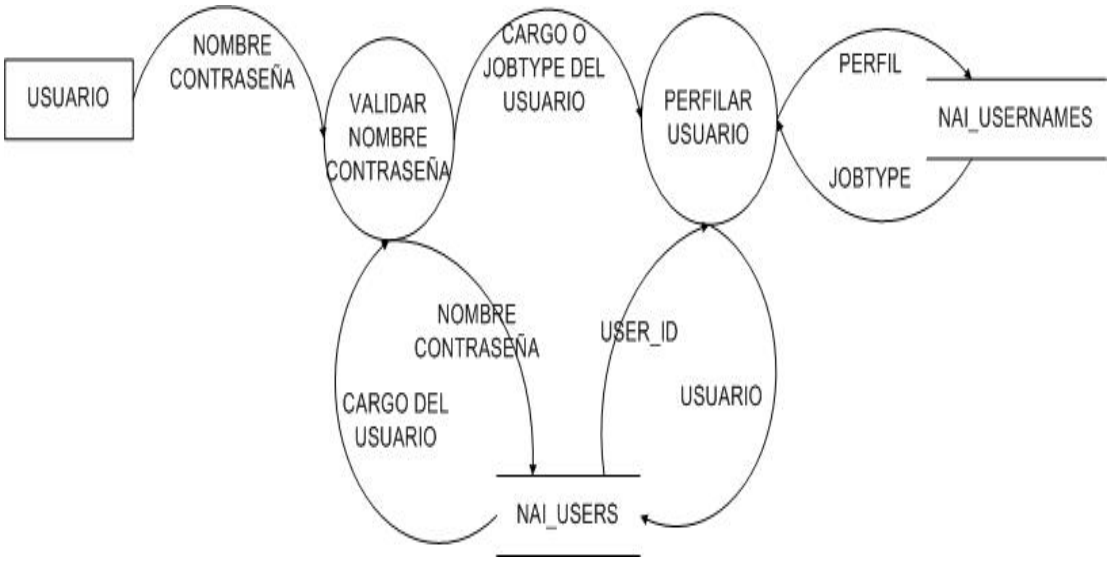


Figura 16. Diagrama De Flujo De Datos: Cargar Archivo

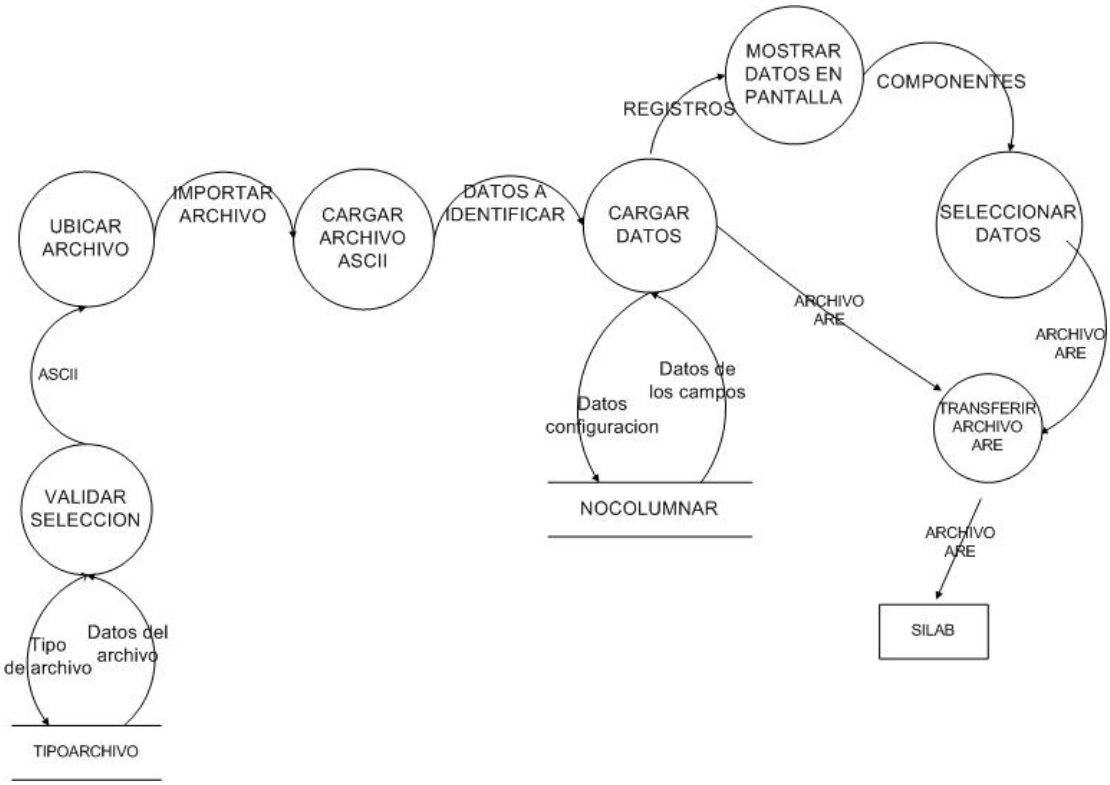


Figura 17. Diagrama De Flujo De Datos: Validar Datos Extraer

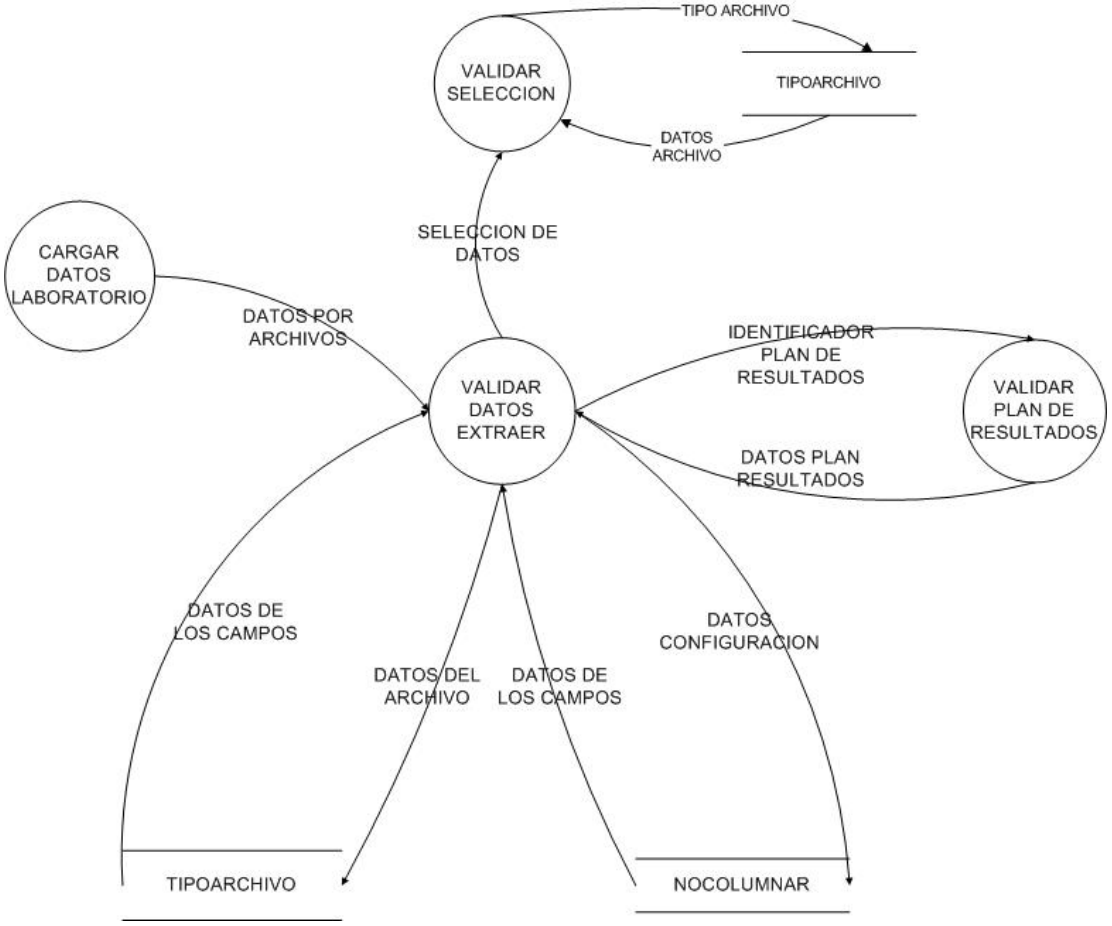
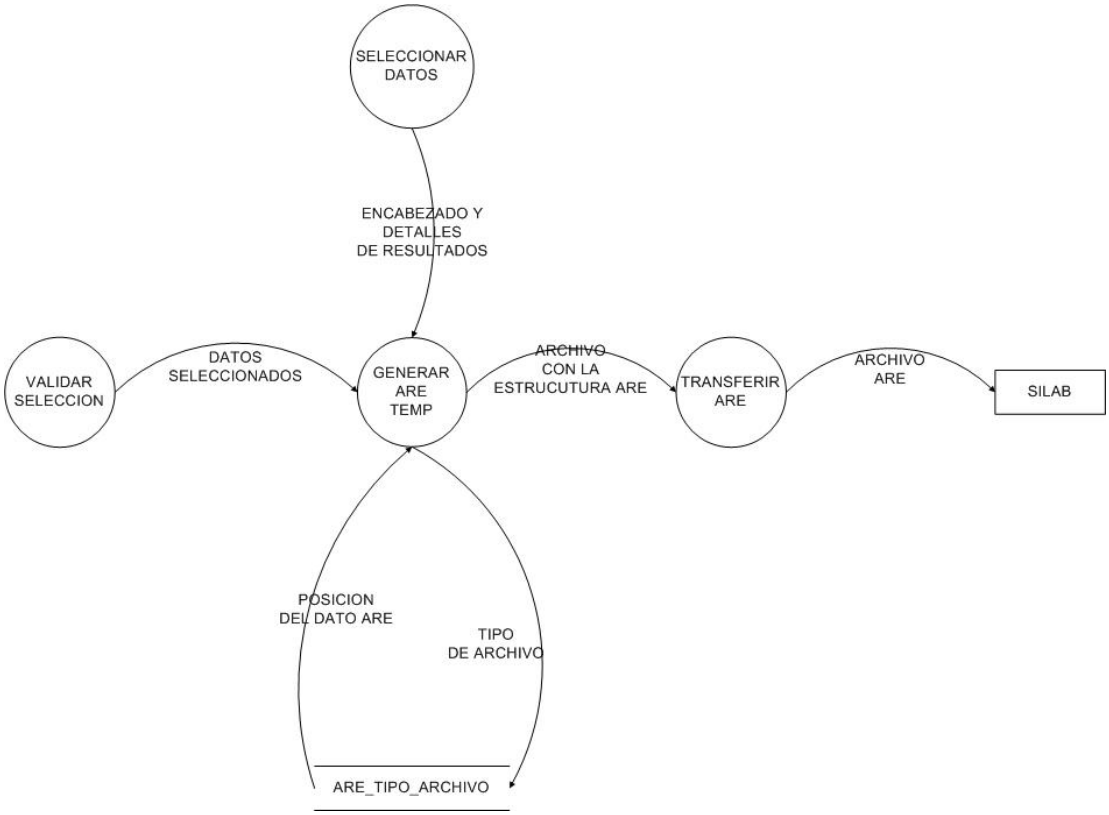


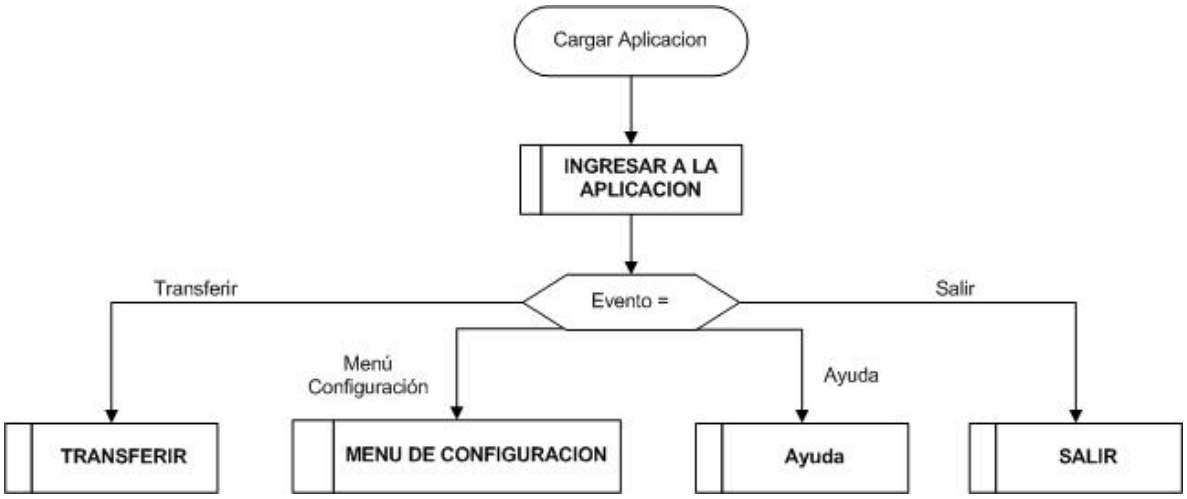
Figura 18. Diagrama De Flujo De Datos: Validar Plan de Resultados

Figura 19. Diagrama De Flujo De Datos: Transferir archivo ARE



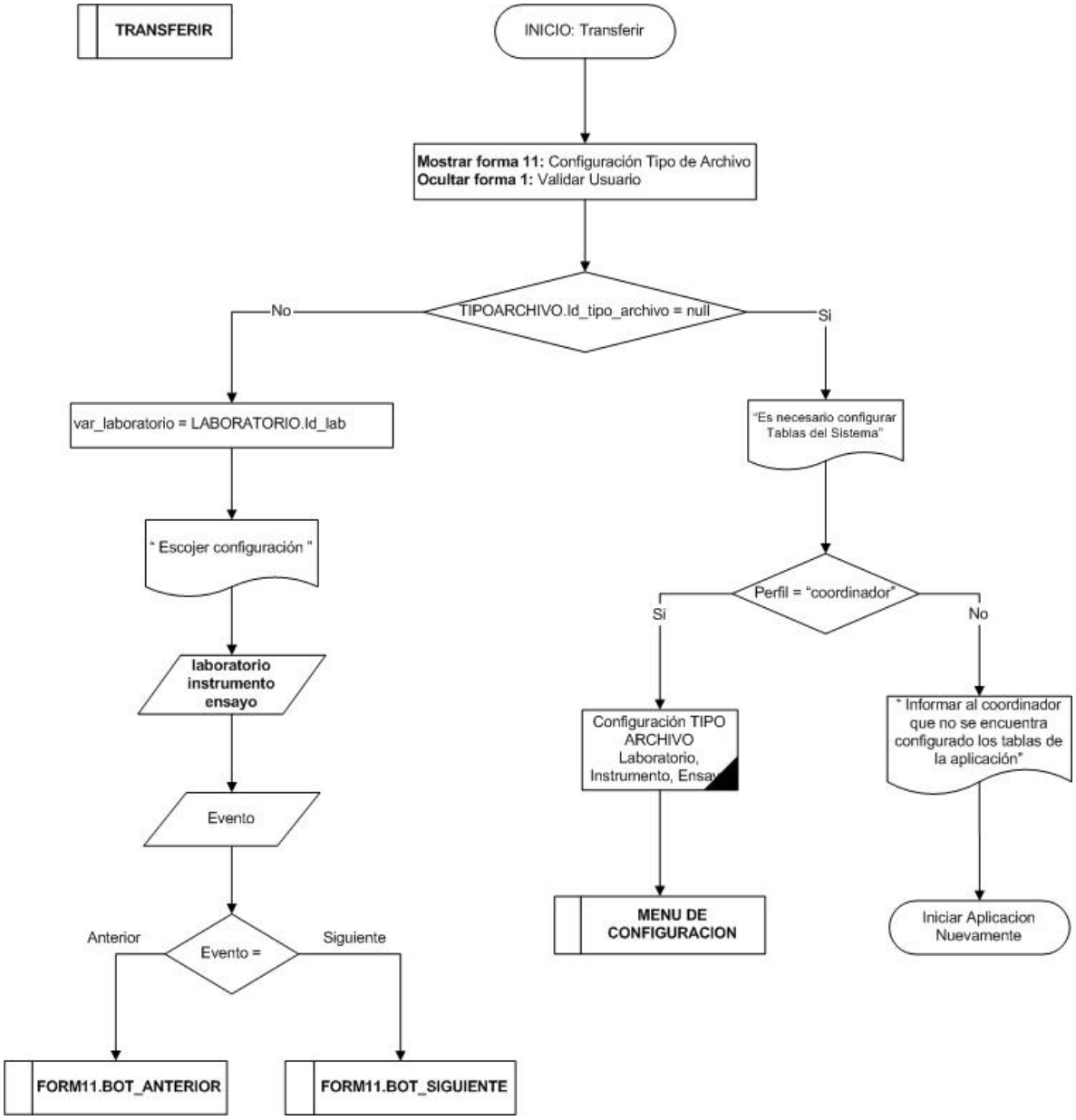
ANEXO 2. ALGORITMO DEL SISTEMA

ALGORITMO GENERAL

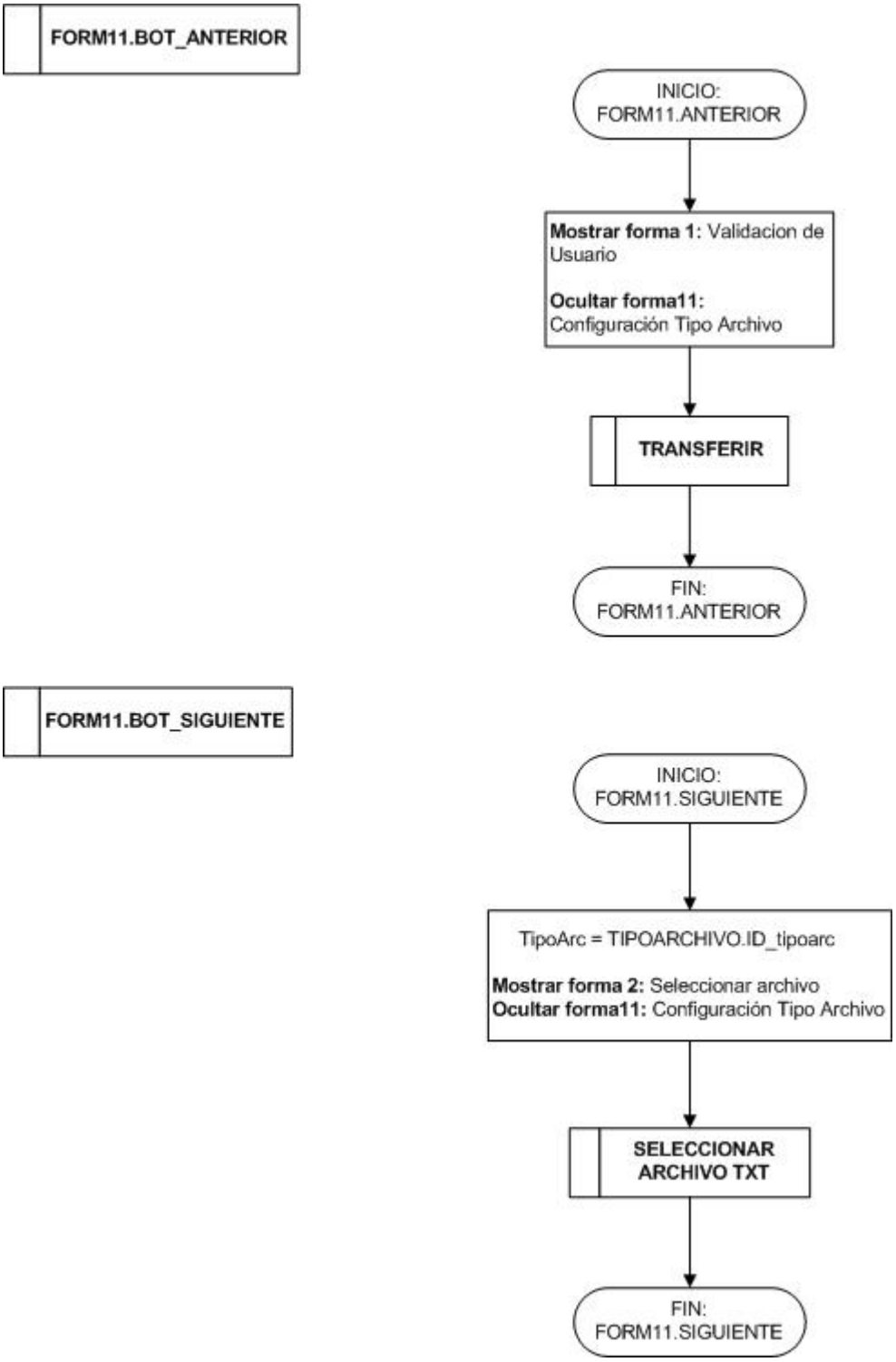


INGRESAR A LA APLICACIÓN

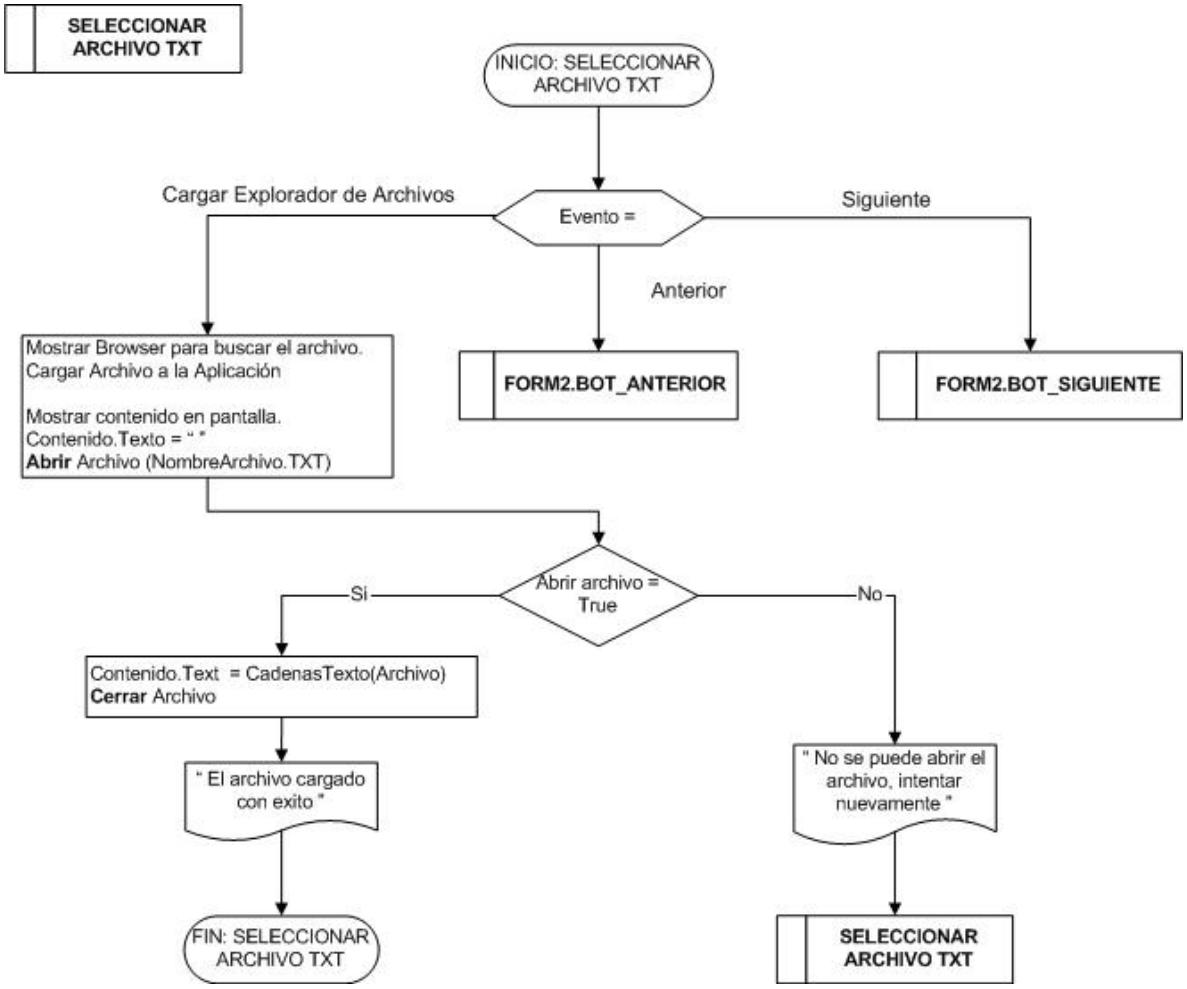
TRANSFERIR DATOS



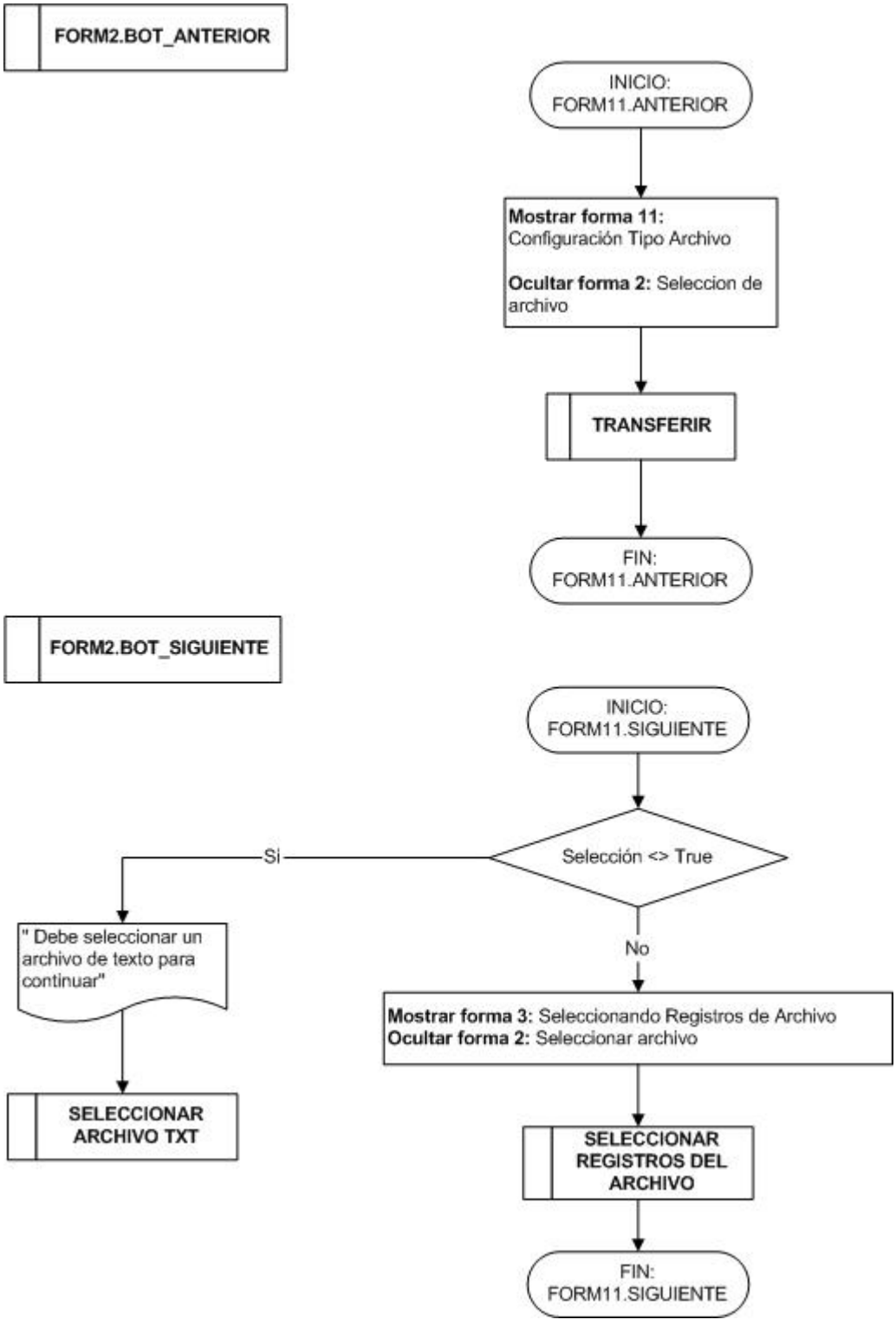
TRANSFERIR DATOS: BOTONES SIGUIENTE – ANTERIOR



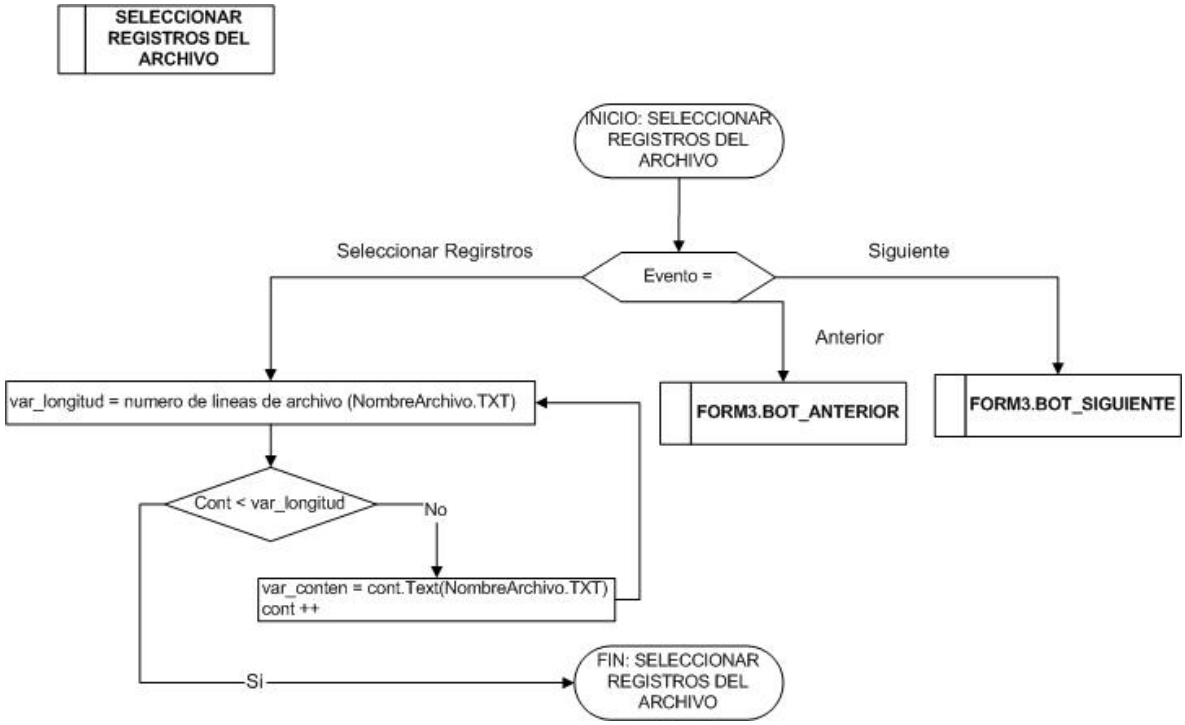
SELECCIONAR ARCHIVO TXT A IMPORTAR



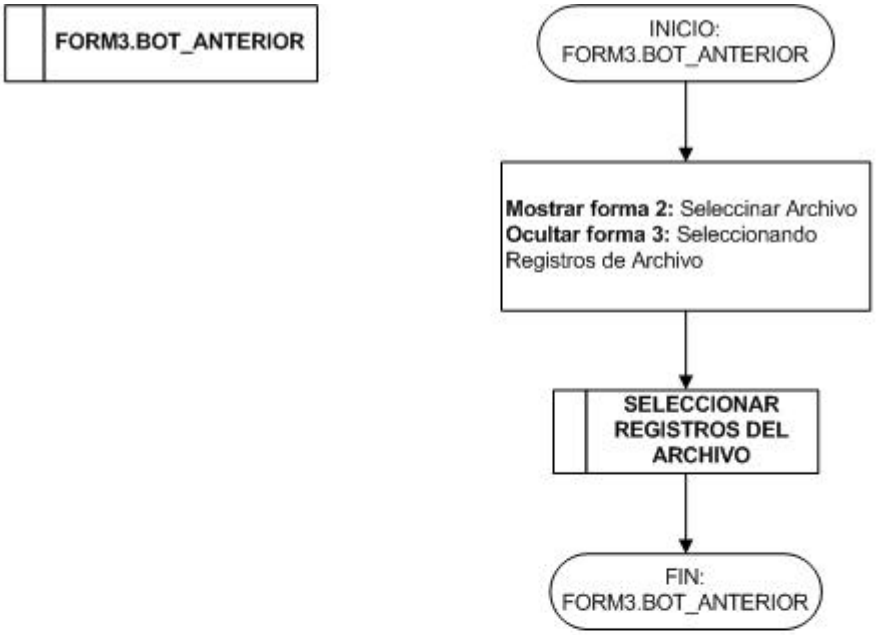
SELECCIONAR ARCHIVO TXT A IMPORTAR: BOTONES SIGUIENTE – ANTERIOR

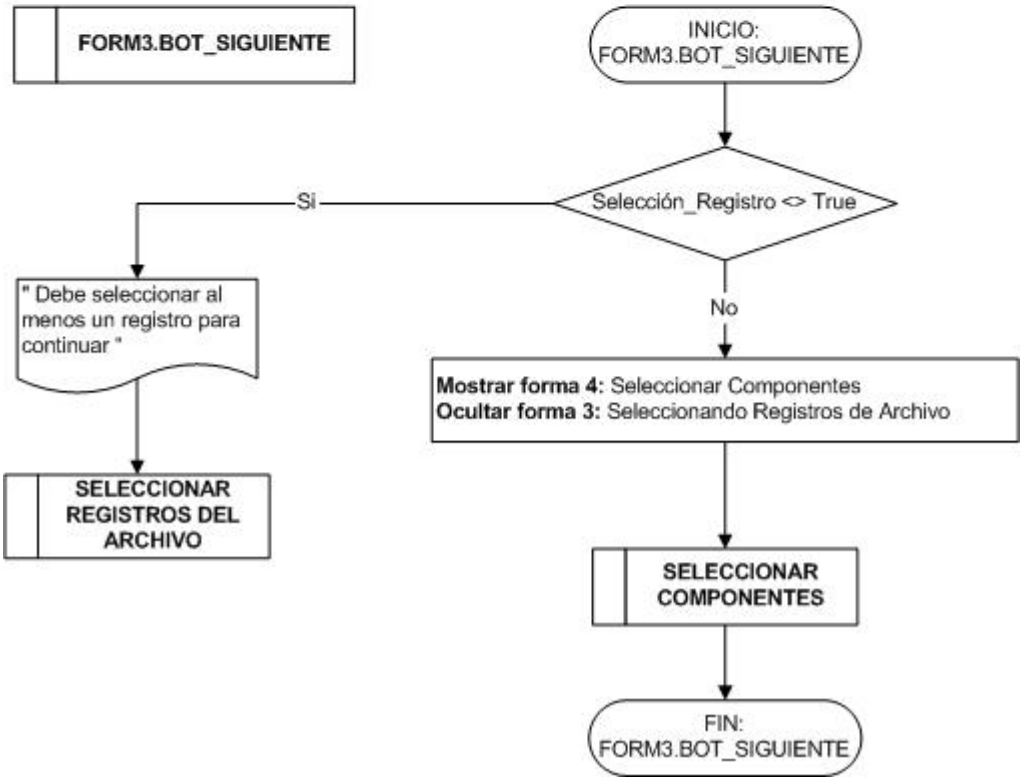


SELECCIÓN DE REGISTROS DEL ARCHIVO TXT A IMPORTAR

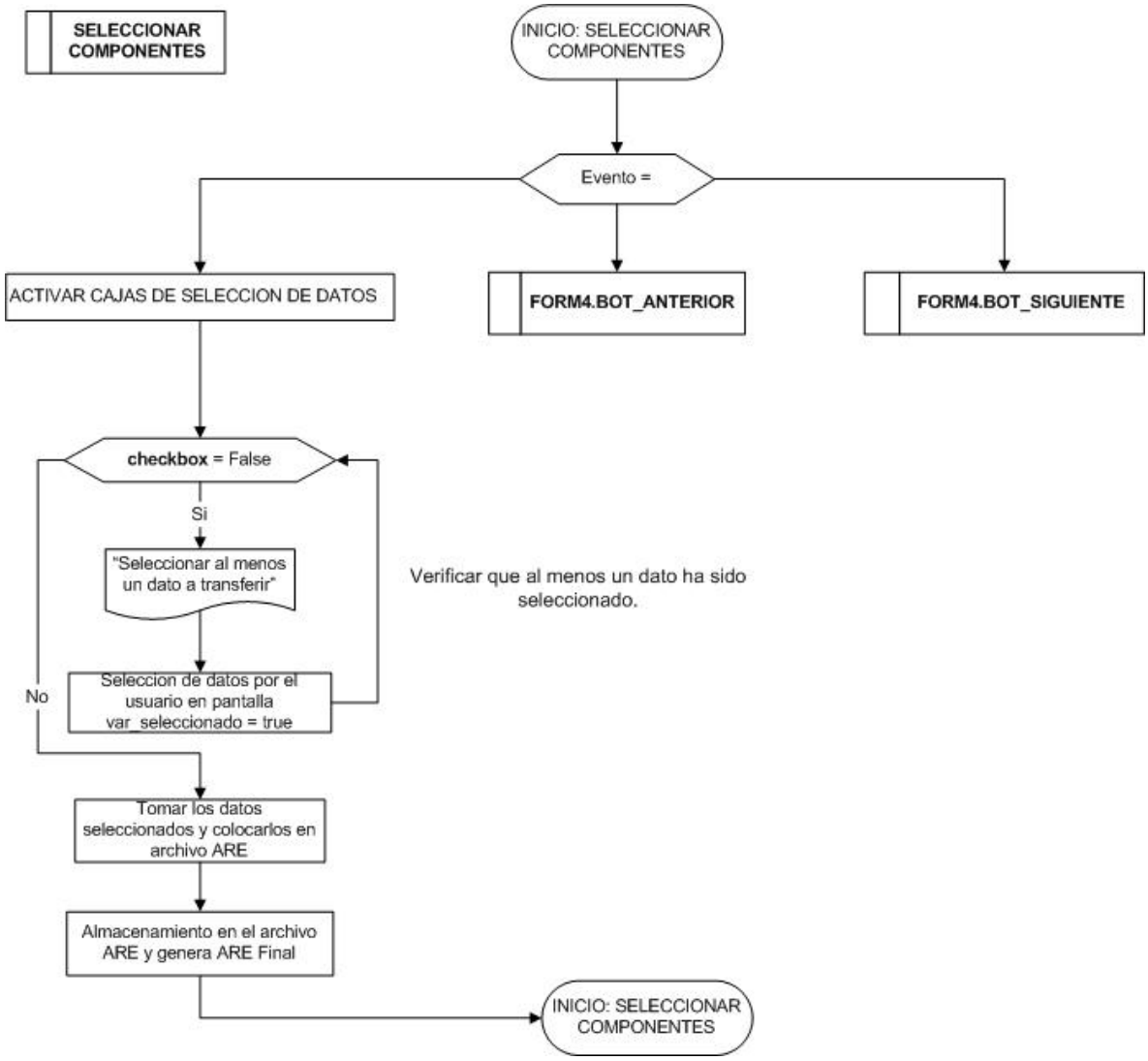


SELECCIONAR REGISTROS DEL ARCHIVO TXT A IMPORTAR: BOTONES SIGUIENTE – ANTERIOR

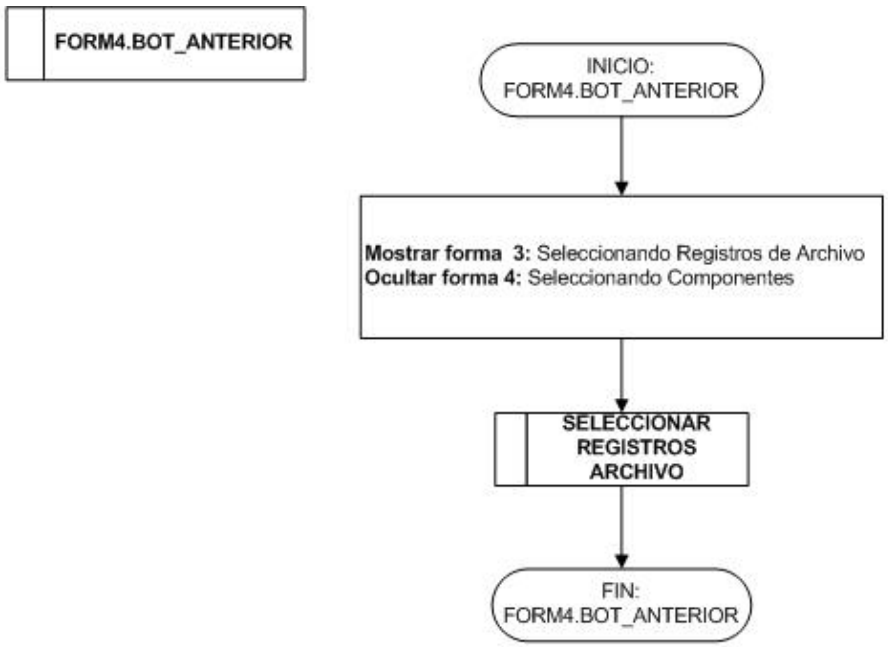


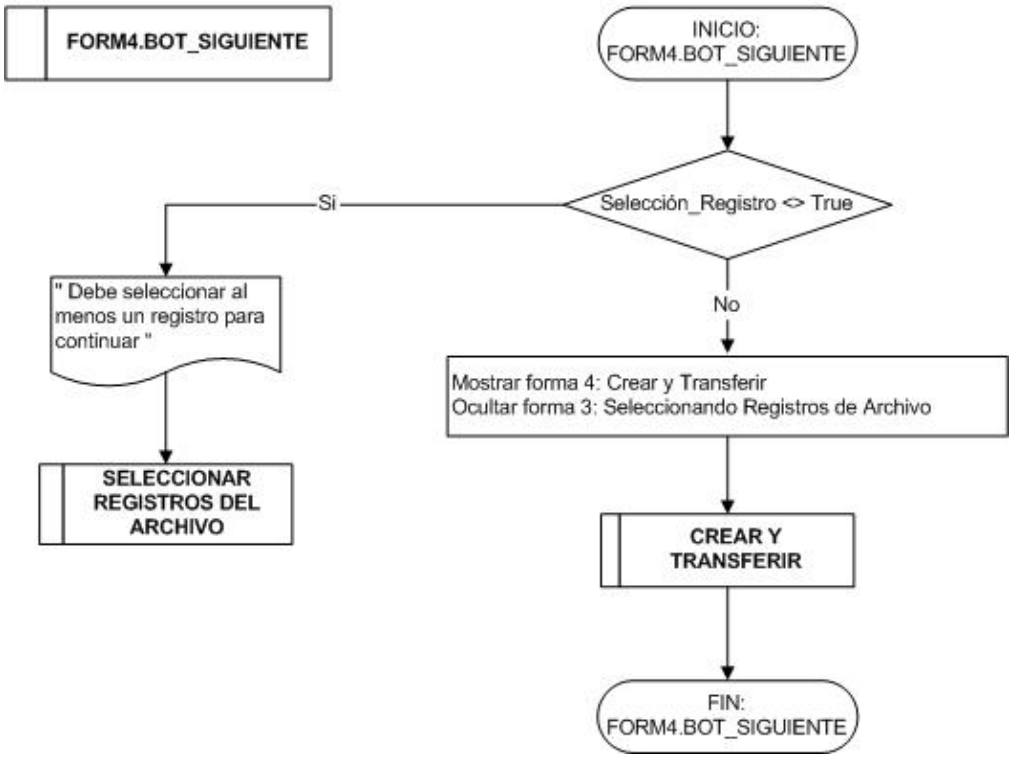


SELECCIONAR COMPONENTES DEL ARCHIVO TXT A IMPORTAR

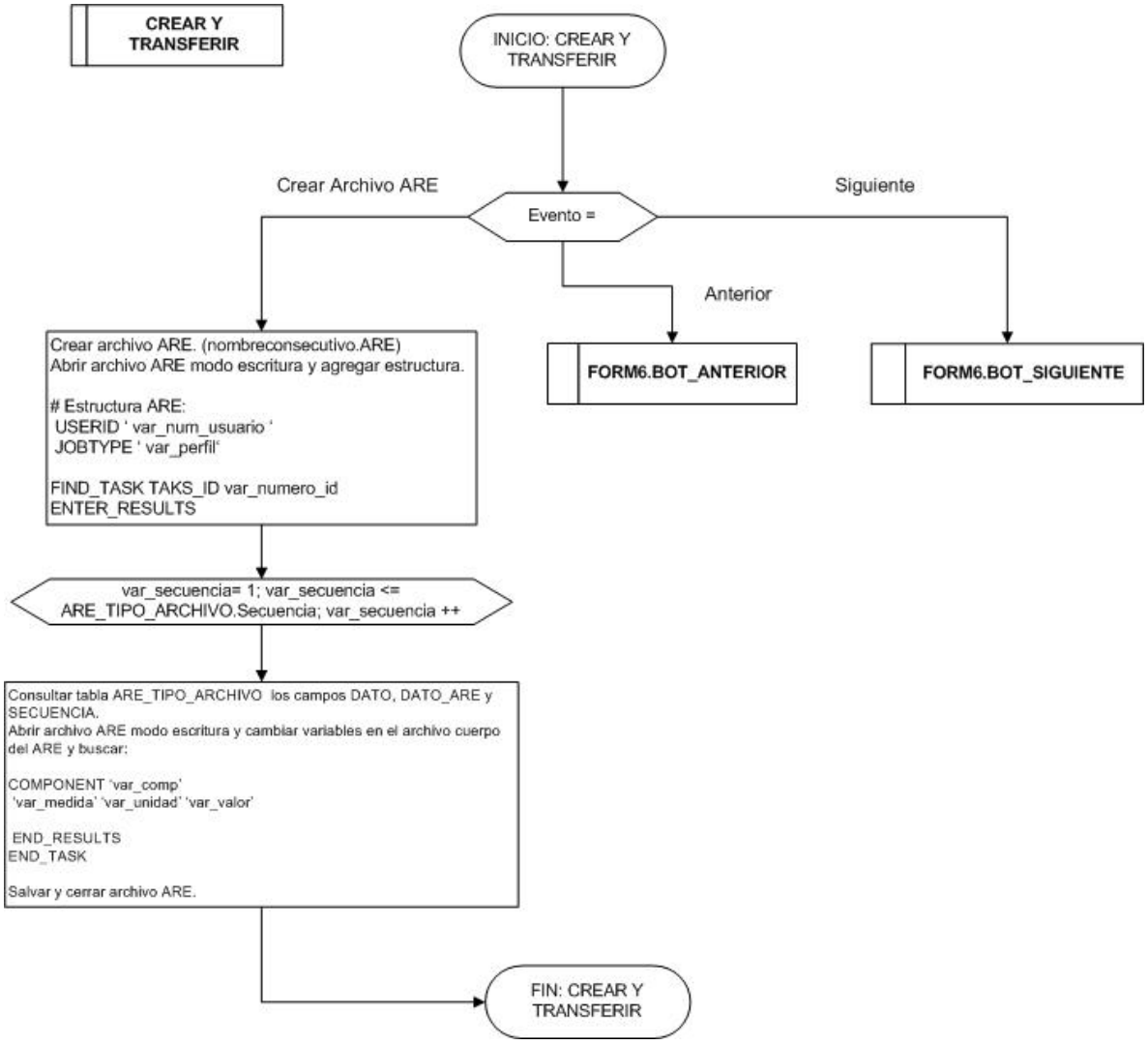


SELECCIONAR COMPONENTES DEL ARCHIVO TXT A IMPORTAR: BOTONES SIGUIENTE – ANTERIOR

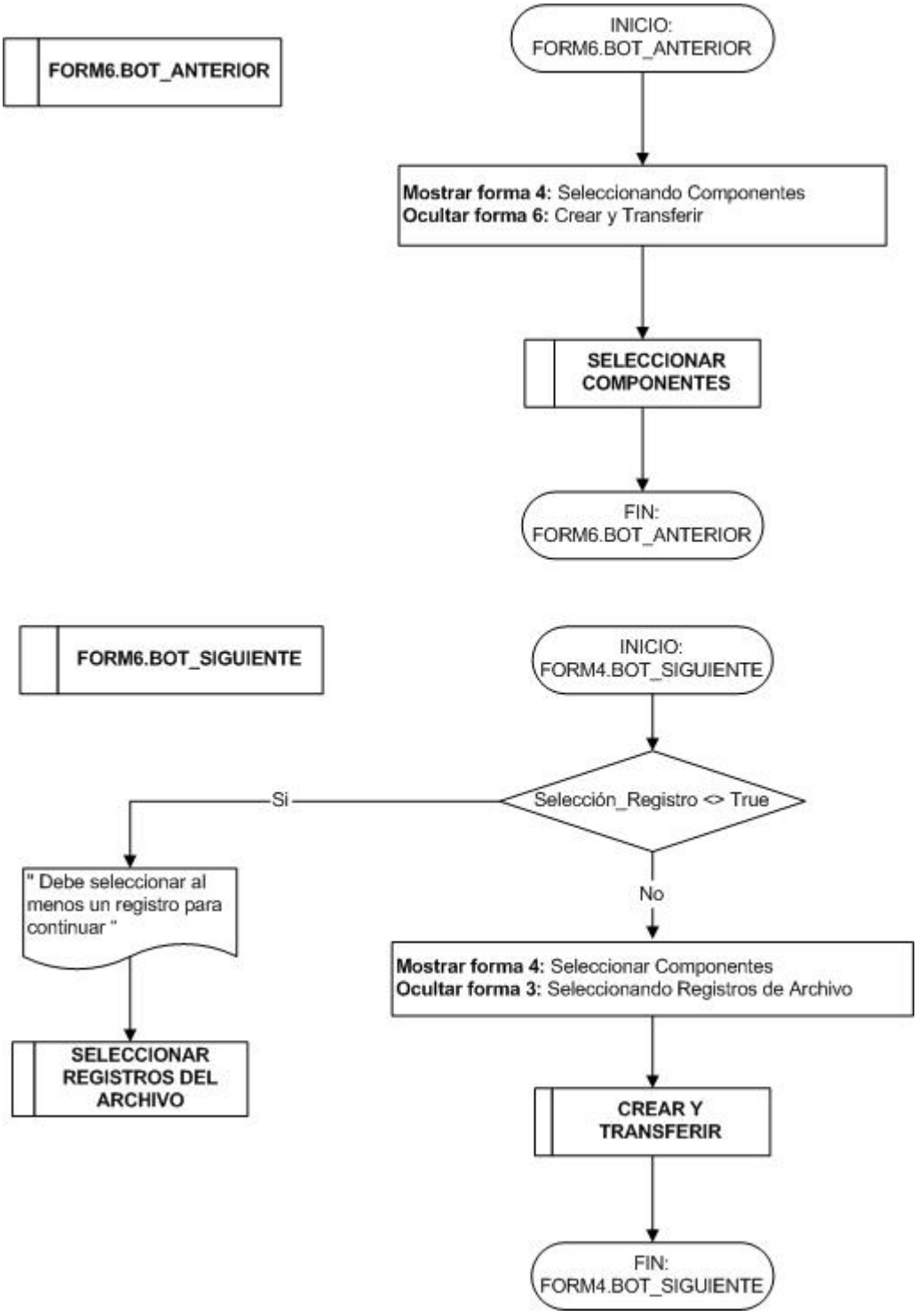




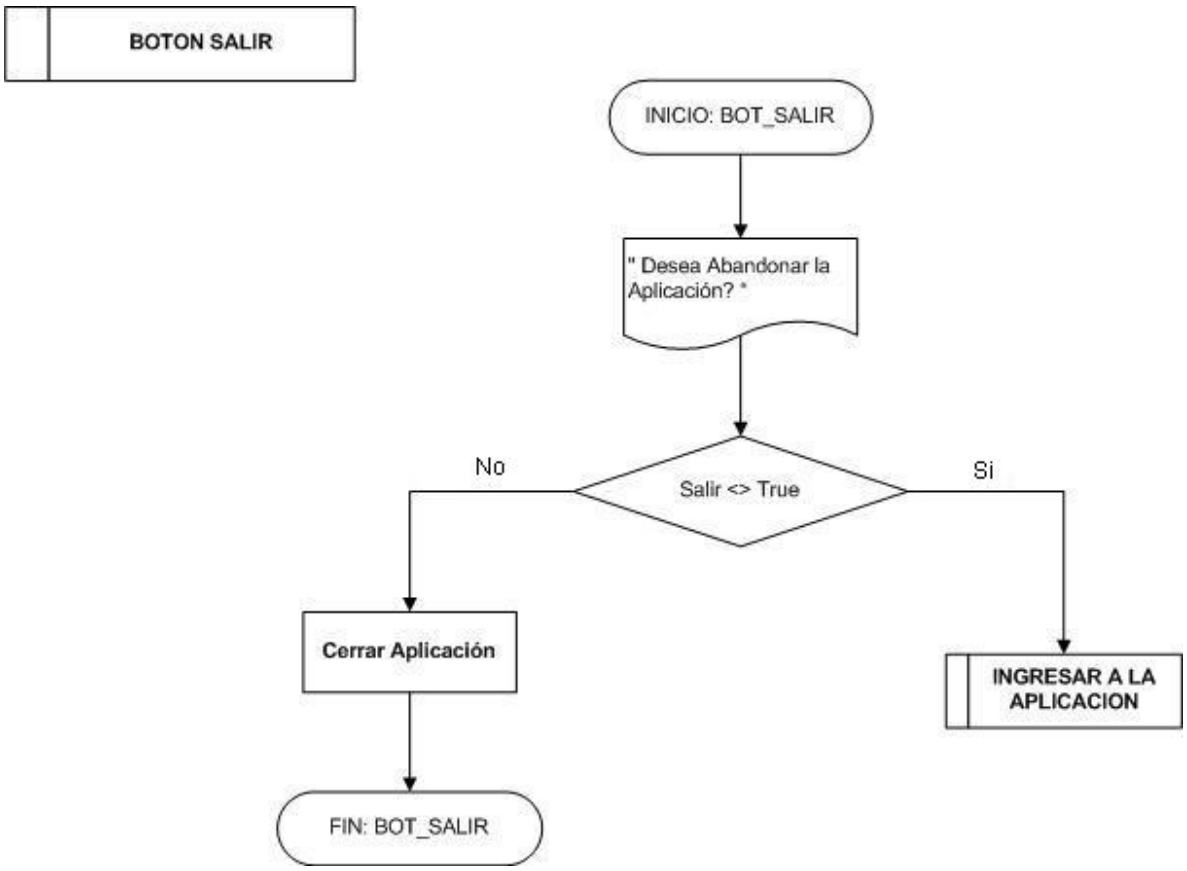
CREAR Y TRANSFERIR ARCHIVO ARE



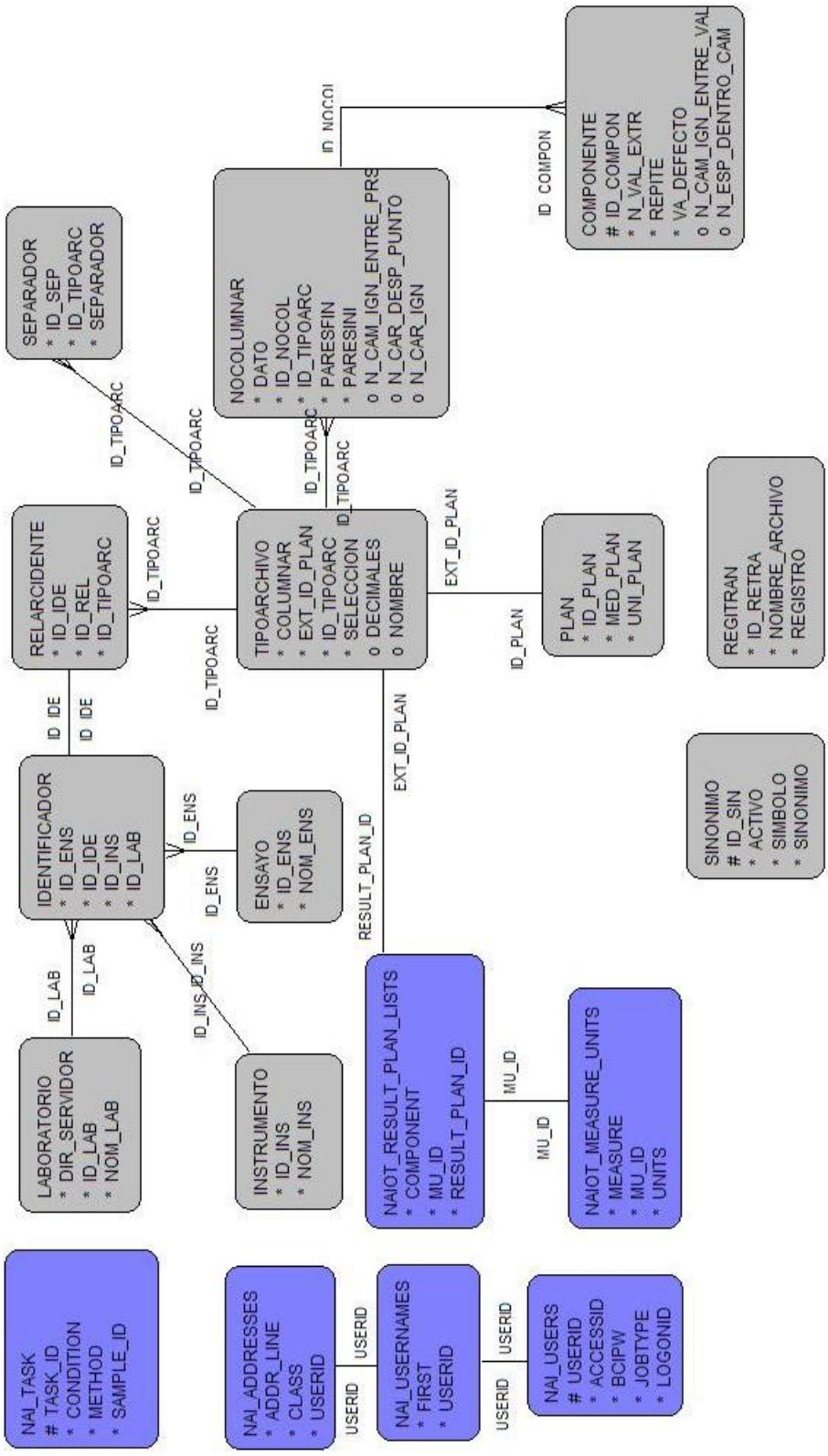
CREAR Y TRANSFERIR ARCHIVO ARE: BOTONES SIGUIENTE - ANTERIOR



SALIR DE LA APLICACIÓN



ANEXO 3. DIAGRAMA ENTIDAD RELACIÓN DEL SISTEMA



ANEXO 4. INTERFAZ DE USUARIO

TRANSFERENCIA AUTOMÁTICA DE DATOS

Figura 20. Cargar archivo ASCII.

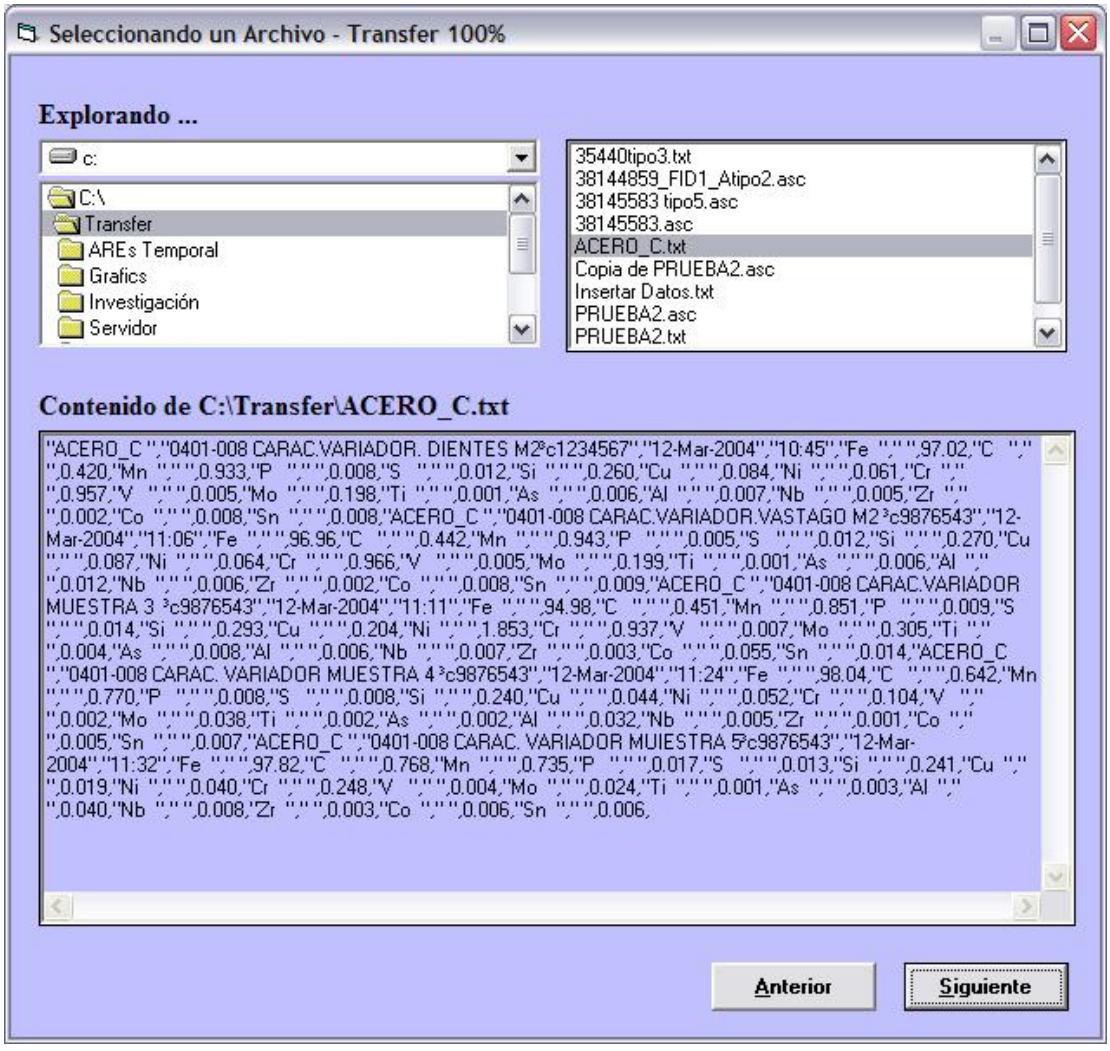


Figura 21. Seleccionando Registros del Archivo.

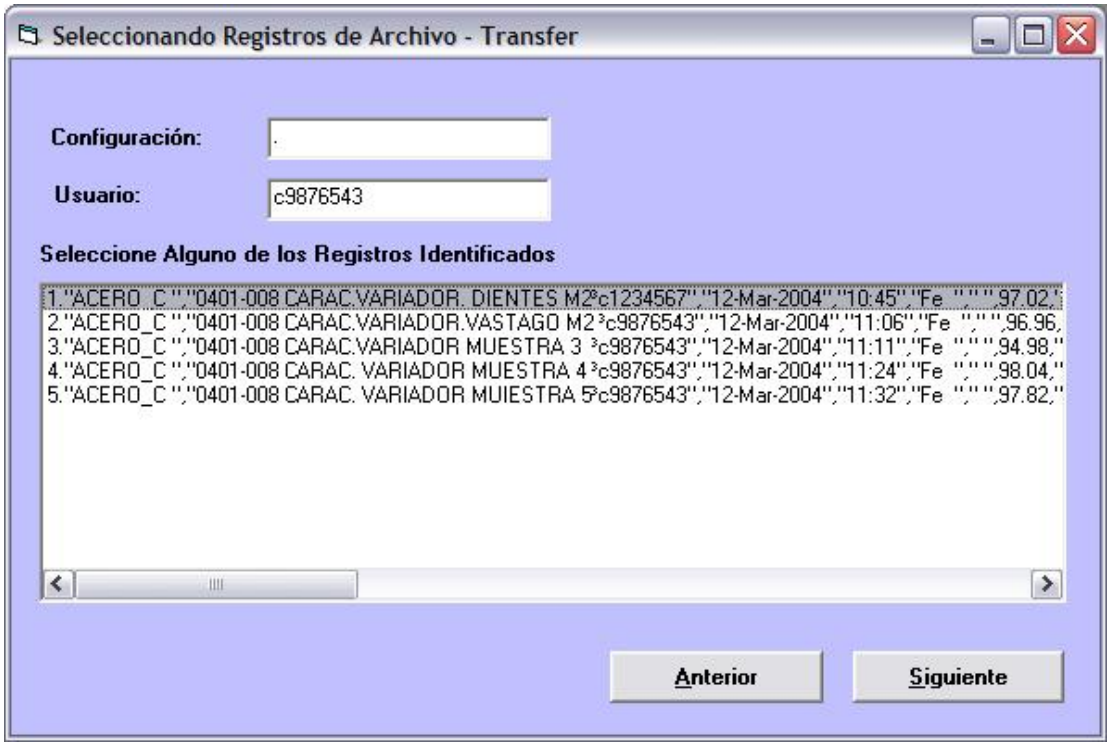


Figura 22. Seleccionando Componentes del Archivo.

Figura 23. Creación y Transferencia de Archivos ARE.

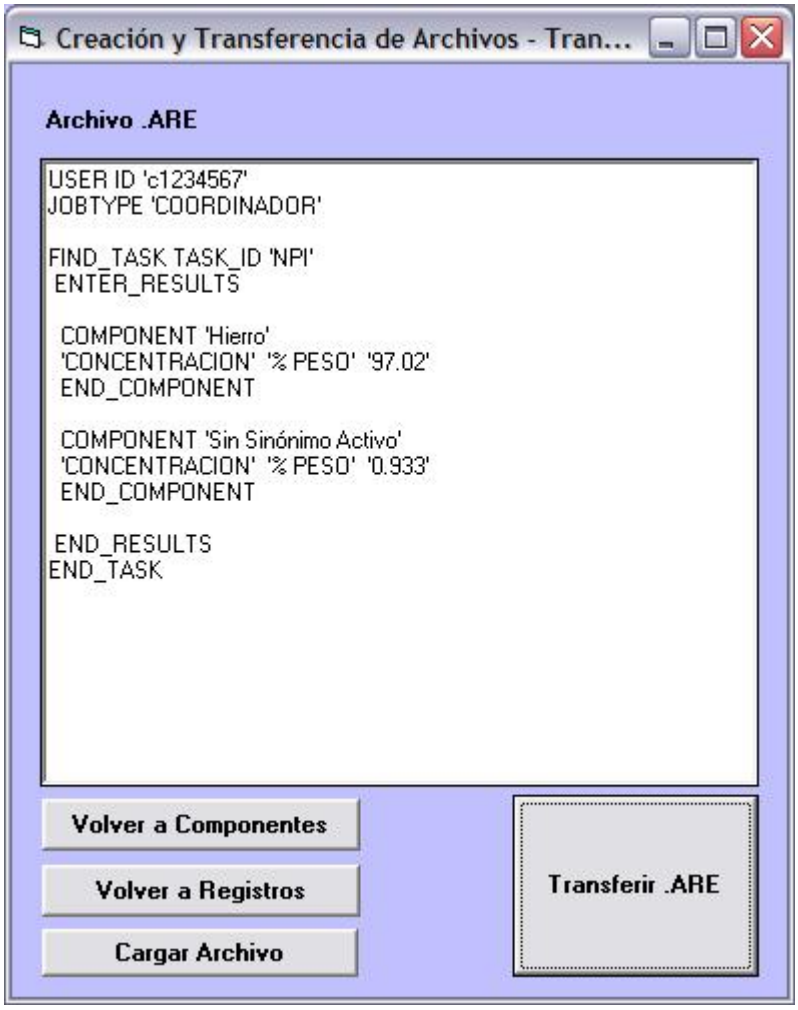
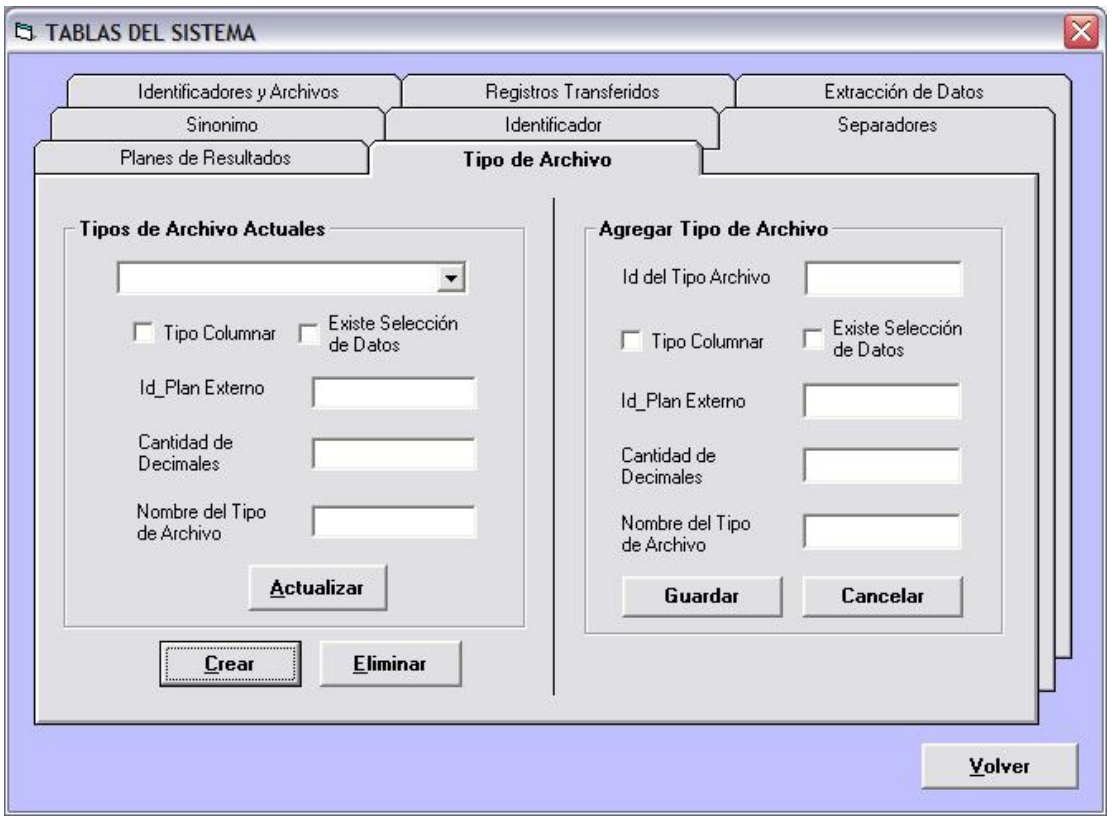


Figura 24. Tablas o Base de Datos del Sistema.



ANEXO 5. REPORTE DE ENTIDAD Y SUS ATRIBUTOS
DICCIONARIO DE DATOS

Archivos generados por Oracle Designer, no modificables:

DescripcionAtributos.pdf

DescripcionEntidades.pdf

+ 19 Hojas: 119

ANEXO 6. SCRIPTS SQL DE CREACION DE TABLAS ORACLE

Tabla Componente

```
PROMPT Creating Table 'COMPONENTE'
CREATE TABLE COMPONENTE
        (N_CAM_IGN_ENTRE_VAL NUMBER(8)
        ,N_VAL_EXTR NUMBER(8) NOT NULL
        ,ID_COMPON NUMBER(8) NOT NULL
        ,N_ESP_DENTRO_CAM NUMBER(8)
        ,REPITE VARCHAR2(50) NOT NULL
        ,VAL_DEFECTO VARCHAR2(50) NOT NULL)
/
ALTER TABLE COMPONENTE ADD CONSTRAINT indicecom
PRIMARY KEY (ID_COMPON) ADD
FOREIGN KEY (ID_COMPON) REFERENCES NOCOLUMNAR (ID_NOCOL);
```

Tabla Ensayo

```
PROMPT Creating Table 'ENSAYO'
CREATE TABLE ENSAYO
        (NOM_ENS VARCHAR2(50) NOT NULL
        ,ID_ENS NUMBER(8) NOT NULL )
/
ALTER TABLE ENSAYO ADD CONSTRAINT indiceens
PRIMARY KEY (id_ens);
```

Tabla Identificador

```
PROMPT Creating Table 'IDENTIFICADOR'
CREATE TABLE IDENTIFICADOR
        (ID_INS NUMBER(8) NOT NULL
        ,ID_LAB NUMBER(8) NOT NULL
        ,ID_IDE NUMBER(24) NOT NULL
        ,ID_ENS NUMBER(8) NOT NULL )
/
ALTER TABLE IDENTIFICADOR ADD CONSTRAINT indiceide
PRIMARY KEY (id_ide) ADD
FOREIGN KEY (id_lab) REFERENCES laboratorio (id_lab) ADD
```

```
FOREIGN KEY (id_ins) REFERENCES instrumento (id_ins) ADD
FOREIGN KEY (id_ens) REFERENCES ensayo (id_ens);
```

Tabla Instrumento

```
PROMPT Creating Table 'INSTRUMENTO'
CREATE TABLE INSTRUMENTO
        (ID_INS NUMBER(8) NOT NULL
        ,NOM_INS VARCHAR2(50) NOT NULL)
/
ALTER TABLE INSTRUMENTO ADD CONSTRAINT indiceins
PRIMARY KEY (id_ins);
```

Tabla Laboratorio

```
PROMPT Creating Table 'LABORATORIO'
CREATE TABLE LABORATORIO
        (NOM_LAB VARCHAR2(50) NOT NULL
        ,ID_LAB NUMBER(8) NOT NULL
        ,DIR_SERVIDOR VARCHAR2(100) NOT NULL)
/
ALTER TABLE LABORATORIO ADD CONSTRAINT indicelab
PRIMARY KEY (id_lab);
```

Tabla Nocolumnar

```
PROMPT Creating Table 'NOCOLUMNAR'
CREATE TABLE NOCOLUMNAR
        (ID_TIPOARC NUMBER(8) NOT NULL
        ,DATO VARCHAR2(50) NOT NULL
        ,ID_NOCOL NUMBER(8) NOT NULL
        ,PARESINI VARCHAR2(50) NOT NULL
        ,PARESFIN VARCHAR2(50) NOT NULL
        ,N_CAR_IGN NUMBER(8)
        ,N_CAM_IGN_ENTRE_PRS NUMBER(8)
        ,N_CAR_DESP_PUNTO NUMBER(8))
/
ALTER TABLE NOCOLUMNAR ADD CONSTRAINT indicenoc
PRIMARY KEY (ID_NOCOL) ADD
FOREIGN KEY (ID_TIPOARC) REFERENCES TIPOARCHIVO (ID_TIPOARC);
```

Tabla Plan

```
PROMPT Creating Table 'PLAN'
CREATE TABLE PLAN
```

```
        (ID_PLAN NUMBER(10) NOT NULL
        ,MED_PLAN VARCHAR2(40) NOT NULL
        ,UNI_PLAN VARCHAR2(40) NOT NULL )
/
ALTER TABLE PLAN ADD CONSTRAINT indicepla
PRIMARY KEY (ID_PLAN);
```

Tabla Regitran

```
PROMPT Creating Table 'REGITRAN'
CREATE TABLE REGITRAN
        (NOMBRE_ARCHIVO VARCHAR2(200) NOT NULL
        ,ID_RETRA NUMBER(8) NOT NULL
        ,REGISTRO NUMBER(8) NOT NULL)
/
ALTER TABLE REGITRAN ADD CONSTRAINT indicereg
PRIMARY KEY (ID_RETRA);
```

Tabla Relarcidente

```
PROMPT Creating Table 'RELARCIDENTE'
CREATE TABLE RELARCIDENTE
        (ID_IDE NUMBER(24) NOT NULL
        ,ID_REL NUMBER(8) NOT NULL
        ,ID_TIPOARC NUMBER(8) NOT NULL )
/
ALTER TABLE RELARCIDENTE ADD CONSTRAINT indicerel
PRIMARY KEY (ID_REL) ADD
FOREIGN KEY (ID_TIPOARC) REFERENCES TIPOARCHIVO (ID_TIPOARC) ADD
FOREIGN KEY (ID_IDE) REFERENCES IDENTIFICADOR (ID_IDE);
```

Tabla Separador

```
PROMPT Creating Table 'SEPARADOR'
CREATE TABLE SEPARADOR
        (ID_SEP NUMBER(8) NOT NULL
        ,SEPARADOR VARCHAR2(50) NOT NULL
        ,ID_TIPOARC NUMBER(8) NOT NULL)
/
ALTER TABLE SEPARADOR ADD CONSTRAINT indicesep
PRIMARY KEY (ID_SEP) ADD
FOREIGN KEY (ID_TIPOARC) REFERENCES TIPOARCHIVO (ID_TIPOARC);
```

Tabla Sinonimo

```
PROMPT Creating Table 'SINONIMO'
CREATE TABLE SINONIMO
        (SIMBOLO VARCHAR2(50) NOT NULL
        ,ID_SIN NUMBER(8) NOT NULL
        ,ACTIVO NUMBER(2) NOT NULL
        ,SINONIMO VARCHAR2(50) NOT NULL)
/
ALTER TABLE SINONIMO ADD CONSTRAINT indicesin
PRIMARY KEY (ID_SIN);
```

Tabla Tipoarchivo

```
PROMPT Creating Table 'TIPOARCHIVO'
CREATE TABLE TIPOARCHIVO
        (COLUMNAR NUMBER(2) NOT NULL
        ,EXT_ID_PLAN NUMBER(10) NOT NULL
        ,ID_TIPOARC NUMBER(8) NOT NULL
        ,NOMBRE VARCHAR2(50)
        ,SELECCION NUMBER(2) NOT NULL
        ,DECIMALES NUMBER(8))
/
ALTER TABLE TIPOARCHIVO ADD CONSTRAINT indicetip
PRIMARY KEY (ID_TIPOARC) ADD
FOREIGN KEY (EXT_ID_PLAN) REFERENCES PLAN (id_plan);
```

ANEXO 7. DICCIONARIO DE EVENTOS

ID del evento	01
Nombre del evento	El usuario inserta nombre y contraseña para ingresar a la aplicación
Descripción	Por medio de tablas de la bases de datos de los laboratorios, se consultará y validará la existencia del usuario para poder ingresar a la aplicación. Si existe el usuario se consultará su perfil, ya sea coordinador del laboratorio o analista.
Actividad	**Si** (Nombre_usuario **and** contraseña = True) **entonces** **Si** (cargo_usuario = perfil) **entonces** Habilitar menú: Mantenimiento Base Datos Aplicación. **Imprimir** "Bienvenido usuario " & perfil **Si no** No Habilitar menú: Mantenimiento Base Datos aplicación. perfil = cargo_usuario; **Fin Si** **Si no** **Imprimir** “Error al ingresar al sistema, nombre o contraseña incorrectos" **Fin si**
Respuesta	Acceder al sistema si el nombre de usuario y contraseña son correctas, de otro modo no ingresa a la aplicación.

ID del evento	02
Nombre del evento	El usuario selecciona el Laboratorio, instrumento y ensayo
Descripción	El usuario escoge de una lista desplegable, el instrumento y ensayo a los que corresponden los resultados que desea transferir.
Actividad	**Leer** Laboratorio **Consulta SQL:** "select nom_ins from instrumento, laboratorio, identificador where instrumento.id_ins = identificador.id_ins and identificador.id_lab = laboratorio.id_lab and laboratorio.nom_lab = '" **Leer** Instrumento, Ensayo **Consulta SQL:** cad = "select nom_ens from instrumento, ensayo, laboratorio, identificador where identificador.id_ens = ensayo.id_ens and identificador.id_lab = laboratorio.id_lab and identificador.id_ins = instrumento.id_ins " **Si (**Selección_Lab = null **) entonces** **Imprimir** "Debe seleccionar un Laboratorio" **Fin Si** **Si (**Selección_Ins = null **) entonces** **Imprimir** "Debe seleccionar un Instrumento" **Fin Si** **Si (**Selección_Ens = null **) entonces** **Imprimir** "Debe seleccionar un Ensayo" **Fin Si** **Validar Combinación:** **Si(**Selección_Lab & Selección_Ins & Selección_Ens = No_Null**)** **Entonces Consultar:** **Si** (TipoArc = (Selección_Lab & Selección_Ins & Selección_Ens)) **Imprimir** "Tipo de Archivo: " & TipoArc ” **Si_no** **Imprimir** "Esta combinación no ha sido creada en el sistema" **Fin si** **Fin si**
Respuesta	Tipo Archivo con el que se trabajara en la aplicación.

ID del evento	03
Nombre del evento	El usuario selecciona la opción de importar archivos ASCII
Descripción	El usuario debe haber colocado en un directorio específico los archivos ASCII con resultados pendientes por transferir. Ubicados los archivos ASCII a importar, la Aplicación procederá a realizar el cargue de esta información, que consistirá en identificar y extraer del archivo los datos relevantes para la transferencia y subirlos a la base de datos.
Actividad	**Mostrar** Browser **NombreArchivo =** Ruta & NombreArchivo **Si (**NombreArchivo (Contenido.text <> “”)**)** **Entonces** **Abrir (**NombreArchivo**)** **Hacer** Contenido.Text = Contenido.Text & cadenas **Fin_Hacer** **CopiarArchivo (**General.copiar_archivo (NombreArchivo)) **MostrarPantalla** ContenidoArchivo **Si_no** **Imprimir** "El archivo esta vació" **Fin si**
Respuesta	Importar archivo en la aplicación o mensaje de error al importar el archivo.

ID del evento	04
Nombre del evento	El usuario selecciona el tipo de transferencia que desea realizar: directa o con selección previa.
Descripción	Dependiendo del tipo de archivo el usuario podrá seleccionar que tipo de transferencia puede realizar, ya sea que no se tenga que seleccionar datos del archivo y solo transferirlo de forma automática o deba realizar una selección previa de los datos del archivo seleccionado.
Actividad	**Si** (Boton_Seleccion = Activo) **entonces** **Activar** Pantalla de Preseleccion de datos. **Si_no** **Si** (Boton_tranferir = Activo) **entonces** **Activar** Transferir_Automatica_Archivo. **Fin Si** **Fin Si**
Respuesta	Preseleccion de los datos a transferir o Transferencia automática de los datos.

ID del evento	05
Nombre del evento	Transferencia directa.
Descripción	La Aplicación transfiere todos los resultados encontrados en el archivo ASCII, sin intervención del usuario.
Actividad	**Si (**TipoArc = TransferenciaDirecta**) Entonces** **Cargar Modulos (**TipoArc**)** **Extraer Datos (**ConsultasTablaExtraerDatos**)** General.mandar_separador General.mandar_nocolumnar Generar Archivos (CopiaArchivo,Separado,Relevantes) Generar Archivo ARE. **Fin Modulo** **Mostrar Pantalla (**Previsualización Archivo ARE**)** **Fin Si.**
Respuesta	Generar archivo ARE y se activa ventana de transferir el archivo a un servidor de archivos.

ID del evento	06
Nombre del evento	Transferencia con selección previa.
Descripción	La Aplicación desplegará en pantalla los resultados encontrados en el archivo y permitirá al usuario realizar un trabajo de selección de los resultados que desea transferir.
Actividad	**Si (**TipoArc = TransferenciaSeleccion**) Entonces** **Cargar Modulos (**TipoArc**)** **Extraer Datos (**ConsultasTablaExtraerDatos**)** General.mandar_separador General.mandar_nocolumnar Generar Archivos (CopiaArchivo,Separado,Relevantes) **Fin Modulo** **Mostrar Pantalla (**Registros Archivo (TipoArc)**)** **SeleccionarRegistros** **Si (**RegSeleccionado <> Null) **Entonces** **Mostrar Pantalla (Componentes (**TipoArc**))** **Si_no** **Imprimir** "Debe seleccionar un registro antes de continuar." **Fin si** **Si (**ComponSeleccionado = Null) **Entonces** **Imprimir "**No ha seleccionado elementos o componentes" **Si_no** **Generar Archivo ARE.** **Fin Si.** **Fin Si.**
Respuesta	Generar archivo ARE y se activa ventana de transferir el archivo a un servidor de archivos.

ID del evento	07
Nombre del evento	El usuario selecciona la opción de transferir los datos a SILAB.
Descripción	La transferencia consistirá en la creación de un archivo (.are), siguiendo los lineamientos determinados por el proveedor de SILAB, y en su envío al servidor de archivos UNIX mediante FTP. Una vez en el servidor, otro programa que está monitoreando permanentemente el servidor, se encargará de tomar los archivos .are y de subir su información a SILAB.
Actividad	**Mostrar Pantalla (**Previsualización Archivo ARE**)** **Cargar Modulo (**Conexión Servidor**)** **Si (**ConexionS <> Null**) Entonces** ConexionS.direcServidor **=** Dirservidor **Transferir (**Archivo ARE**)** **Si (**Transferir = True**) Entonces** **CopiarArchivo (**Copiar ARE Directorio Transferidos**)** **Imprimir** "El archivo enviado satisfactoriamente". **Fin_si** **Si_no** **Imprimir** "Problemas al conectarse con el Servidor". **Fin_si** **Fin modulo**
Respuesta	Transferir archivo ARE a servidor de archivo, de otro modo respuesta de acerca de la transferencia.

ID del evento	08
Nombre del evento	El usuario con perfil de Administrador, hace mantenimiento a las tablas de Configuración de la Aplicación
Descripción	Cada laboratorio podrá definir una serie de parámetros que permitirán a la Aplicación realizar acciones en forma automática, y las estructuras de los archivos ASCII asociados a cada instrumento y tipo de ensayo, con base en lo cual se podrá realizar la identificación y extracción automática de los datos de estos archivos.
Actividad	// Configuración de TABLAS de la APLICACIÓN. **Presentar** Tablas de mantenimiento de la Aplicación. EXTRACCION_DATOS REGISTROS_TRANSFERIDOS IDENTIFICADOR Y ARCHIVOS TIPO_ARCHIVOS PLAN_RESULTADOS SEPARADOR IDENTIFICADOR SINONIMO **Leer** OpcionTabla **Sea el Caso** OpcionTabla = TABLASAPLICACION **entonces** **Mostrar (**TABLAS de Mantenimiento) **Llenar Datos (**NuevosRegistros**)** **ActualizarRegistros (**ActualizarBaseDatos**)** **Fin_Caso**
Respuesta	Administración de tablas de configuración de la aplicación.