MODULO DE CONTROL DE PEDIDOS Y REMISIONES DE MERCANCIA DE LA EMPRESA INDUSTRIAS ROMAN LTDA (ERLASS)

YAZMIN TINJACA LANCHEROS
CINDY JOHANA MEJIA CALDERON

CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS. UNIMINUTO
FACULTAD DE INGENIERÍA
TECNOLOGÍA EN INFORMÁTICA
SOACHA, CUNDINAMARCA
2010

MODULO DE CONTROL DE PEDIDOS Y REMISIONES DE MERCANCIA DE LA EMPRESA INDUSTRIAS ROMAN LTDA (ERLASS)

YAZMIN TINJACA LANCHEROS ID: 000066121
CINDY JOHANA MEJIA CALDERON ID: 000066546

Trabajo de grado para optar el titulo Tecnología en Informática

MAURICIO BERMUDEZ
Director de Proyectos

CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS. UNIMINUTO
FACULTAD DE INGENIERÍA
TECNOLOGÍA EN INFORMÁTICA
SOACHA, CUNDINAMARCA
2010

Nota de Aceptación

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Soacha, Cundinamarca 11 Mayo 2010

DEDICATORIA

A Dios por darnos fortaleza y permitir este momento;
A nuestros padres Ana Lucía Lancheros, Pablo Tinjacá,
Susana Calderón, Néstor Mejía,
por ser el apoyo y la guía en todo momento;
A UNIMINUTO por aportarnos conocimiento;
A nuestros amigos por animarnos en los momentos difíciles.

AGRADECIMIENTOS

A Mauricio Román, Ingeniero de Sistemas, Subgerente de INDUSTRIAS ROMAN LTDA.

A Mauricio Ruiz, Ingeniero de Sistemas, Docente UNIMINUTO Regional Soacha 2009.

A Julián Romero, Ingeniero de Sistemas, Asesor de proyecto UNIMINUTO Regional Soacha 2010.

.

A Ricardo Bernal, Ingeniero de Sistemas, Docente UNIMINUTO Regional Soacha 2010.

A Carlos Charry, Ingeniero de Sistemas, Docente UNIMINUTO Regional Soacha 2010.

A Mauricio Bermúdez, Ingeniero de Sistemas, Coordinador de Tecnología en Informática UNIMINUTO Regional Soacha.

CONTENIDO

INTRODUCCION

Los diseñadores de sistemas continuamente buscan métodos mejorados de desarrollo para poder comprender y comunicar los requisitos que la aplicación debería cumplir; décadas atrás, este tipo de soluciones se describían textualmente y se desarrollaban en un modelo estructurado basado en un análisis de suceso-respuesta, en donde el objetivo principal era enlazar los datos con los procesos, descuidando la seguridad y excediendo la cantidad de código.

Desde la década de 1980 empezaron a aparecer modelos de análisis, diseño y codificación en donde se definían objetos y se reducía al mínimo el código derivado de los datos. Para apoyar la codificación orientada a objetos el análisis y el diseño previos a este se deben centrar en la identificación de objetos, la organización jerarquizada de clases, la reutilización de clases y la construcción de marcos estructurales a partir de librerías de clases.

Al analizar los requisitos se obtienen los procesos que la aplicación debe realizar, esto se documenta con distintos diagramas, luego se definen las clases, los objetos y los atributos que van a ser parte del código de la aplicación y a partir de esto se estructuran marcos donde soportará la interacción con el usuario. Para facilitar la aplicación de la programación orientada a objetos se suele hacer en una arquitectura en capas

En este trabajo se describe la arquitectura que se sigue en el desarrollo de ERLASS, las cuales son: capa de presentación, capa de almacenamiento y capa de negocio

La capa de presentación se refiere a la interfaz grafica, es decir a parte del software que puede verse. Por medio de esta el usuario interactúa con la

aplicación; por ello es necesario que tenga una presentación estética y ordenada. El lenguaje de marcado mas utilizado actualmente para ello es HTML, este permite la fácil integración de texto ordenado y junto con CSS (hojas de estilo en cascada) logran crear un ambiente visual atractivo para el usuario de la aplicación.

La capa de almacenamiento es aquella donde todos los datos son guardados para su posterior utilización. Con la utilización de MySQL se puede gestionar los datos a través de consultas que se integran fácilmente a la capa de negocio.

La capa de negocio permite la conexión de las dos anteriores capas además del tratamiento de los datos para ser presentados al usuario. El desarrollo en JAVA (lenguaje de programación) permite que esta capa obtenga los beneficios de la programación orientada a objetos, permitiendo así mejor seguridad al momento de la transferencia de datos en un ambiente web.

1. TEMA:

- **INFORMATICA:**

Aplicación de conocimientos y técnicas en el tratamiento de datos y transferencia de datos para satisfacer las necesidades del cliente.

2. PROBLEMA:

La gran mayoría de empresas manufactureras crean productos de consumo masivo para acceder a grandes mercados, por ello es necesario realizar un control exhaustivo de los clientes y de las mercancías solicitadas regularmente por estos. Dicho control se hace para disminuir los costos de la producción y aumentar las ganancias de la empresa.

¿Como automatizar y gestionar las pedidos y remisiones de mercancía de la empresa Industrias Román LTDA.?

3. OBJETIVOS

3.1 OBJETIVO GENERAL:

Construir modulo de control para gestionar y automatizar los pedidos y las remisiones de mercancía de la empresa Industrias Román LTDA.

3.2 OBJETIVOS ESPECIFICOS:

Reconocer la actividad social de la empresa Industrias Román LTDA y todos los procesos que involucran al proyecto.

Analizar el sistema actual encontrando sus falencias y así desarrollar el sistema propuesto.

Diseñar capa de datos, capa de presentación y Construir capa de negocio Orientado a Objetos

Construir capa de datos, capa de presentación y Construir capa de negocio Orientado a Objetos

Realizar documentación: manual técnico y manual de usuario

4. MARCO REFERENCIAL:

4.1 MARCO TEORICO:

- La empresa Industrias Román LTDA proveerá toda la información interna y los datos relacionados con los pedidos y remisiones de mercancía, necesarios para el desarrollo del modulo ERLASS; además los permisos para el ingreso al modulo actual.

Para el desarrollo de las aplicaciones actualmente se están utilizando los patrones de diseño y arquitectura de software basadas en modelos de construcción de los cuales hay 23 esquemas reconocidos en el mercado. Uno de estos esquemas es el modelo vista controlador (MVC).

Según Deitel y Deitel en su libro Como Programar en Java “los patrones de diseño describen estrategias comprobadas para crear sistemas confiables de software orientada a objetos” (Deitel y Deitel, 2004, 655) y donde “la arquitectura MVC ayuda a crear sistemas confiables y fáciles de modificar” (Ibid, 2004, 656), el divide el MVC en tres partes:

- El modelo que mantiene los datos y la lógica de los programas.
- La vista que proporciona una presentación visual del modelo
- El controlador que procesa la entrada del usuario y hace modificaciones al modulo.

Para estructurar y visualizar el comportamiento de cada una de las partes del MVC se utiliza el lenguaje de modelado unificado (UML), con el cual se realizan diagramas donde se pueden visualizar los procesos internos de las aplicaciones , la estructura de la base de datos BD, y la estructura de la interfaz grafica GUI.

Roger Preesman en su libro Ingeniería del software un enfoque practico, se refiere a la utilización del lenguaje de modelado de datos diciendo: “UML permite a un ingeniero del software expresar un modelo de análisis utilizando una notación de modelado con unas reglas sintácticas, semánticas y prácticas.” (Preesman, 2002, 362)

La utilización del modelado de datos a través de UML en el desarrollo de cualquier aplicación de software tiene el propósito de darle un enfoque Orientado a objetos; esto quiere decir que se pueden aprovechar sus beneficios, entre ellos:

- Menor tiempo en el desarrollo de la aplicación.
- Menor gasto de recursos
- Menos requerimientos (en cuanto a memoria para la ejecución de la aplicación)
- Mayor aprovechamiento del código de fuente (ya que se puede volver a utilizar)

Pressman muestra los beneficios de un DOO (Diseño Orientado a Objetos) asegurando que: “A diferencia de los métodos de diseño de software convencionales, el DOO proporciona un diseño que alcanza diferentes niveles de modularidad. La mayoría de los componentes de un sistema, están organizados en subsistemas, un «módulo» a nivel del sistema. Los datos y las operaciones que manipulan los datos se encapsulan en objetos –una forma modular que es el bloque de construcción de un sistema OO (orientado a objetos) -. Además, el DOO debe describir la organización específica de los datos de los atributos y el detalle procedural de cada operación. Estos representan datos y piezas algorítmicas de un sistema OO y son los que contribuyen a la modularidad global.”

La utilización de modelos y diagramas UML tal como lo son los diagramas de casos de uso y de clases son fundamentales para la creación de aplicaciones orientadas a objetos, para Benet Campderrich este es uno de los primeros del análisis es traducir los requisitos que se expresan por el cliente y expresarlos en un lenguaje mas formal, el segundo seria la identificación de clases fundamentales las cuales son base para la implementación del software. (Campderrich, 2003, 146)

Para que ERLASS (Modulo de pedidos y remisiones) este vanguardia de la tecnología es necesario que sea un sistema OO (orientado a objetos), por ello es necesario que el lenguaje de programación a utilizar para la capa de negocio o capa de procesamiento permita crear objetos a partir de clases.

El lenguaje más adecuado para ello es Java, ya que es un lenguaje por naturaleza orientado a objetos. La empresa Sun Microsystems en la página web de Java se refiere a las ventajas de su creación:

“Java ha sido probado, mejorado, ampliado y probado por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo. Gracias a su versatilidad, eficiencia y portabilidad.” Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma (Multiplataforma)
- Crear programas para que funcionen en un navegador web y en servicios web (Aplicaciones orientadas a la web)
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados

- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital". (Sun Microsystems, 2009)

En la capa de presentación ERLASS esta desarrollado en el lenguaje de marcado HyperText Markup Language (Lenguaje de Marcado de Hipertexto) mas conocido por sus siglas HTML, que en conjunto con CCS (hojas de estilos en cascadas) permiten un diseño visualmente agradable y de muy fácil acceso.

4.2 MARCO CONCEPTUAL:

¿QUE ES?

Es un software que nos permite controlar los pedidos y remisiones de la mercancía de la empresa Industrias Román LTDA.

- ¿COMO ES?

Erlass (Modulo de pedido y remisiones) se compone por: capa de almacenamiento desarrollada en Mysql porque es una estructura donde la información está relacionada entre si, por tanto es recomendable para manejar numerosos registros (datos almacenados en memoria física, RAM o ROM), capa de diseño en Seudo-lenguaje HTML, capa de procesamiento en lenguaje de programación Java porque nos permite trabajar estructuras orientadas a objetos OO y posee una licencia tanto de desarrollo como de utilización libres.

-**¿PARA QUE ES?**

Erlass (Modulo de pedido y remisiones) se desarrollo para capturar los datos de los pedidos y las remisiones de la empresa Industrias Román LTDA tales como los

datos del proveedor, el numero de pedido, la cantidad, el detalle, el valor total de la mercancía al igual que el respectivo numero de remisión, cantidad, y descripción del pedido y remisión; de igual forma Erlass procesa los datos y los compara obteniendo información que será remitida a los departamentos de producción y de contabilidad.

Por otra parte el modulo de pedidos y remisiones estará en la capacidad de gestionar la información y presentarla en estadísticas útiles para la toma de decisiones gerenciales.

-**¿POR QUE?**

El nuevo modulo de pedidos y remisiones soluciona las necesidades del cliente en cuanto a la GUI porque su distribución y diseño será agradable para los usuarios y de fácil manejo permitiendo una captura de datos mas fluida y con mayor precisión, además de proporcionarle a la empresa Industrias Román ltda. Una herramienta de software que gestione la información de los pedidos y las remisiones, arrojando resultados que le permitan al área administrativa tomar decisiones trascendentales para el crecimiento de su mercado.

4.3 MARCO ORGANIZACIONAL:

*** QUIENES SON:**

- Industrias Román LTDA es una empresa dedicada a la fabricación de muebles metálicos y partes para bicicletas con 34 años de actividad y 50 empleados, esta empresa esta constituida bajo una sociedad limitada con el NIT 800.211.555-3.

MISIÓN DE LA EMPRESA

Industrias Román Ltda., es una empresa Colombiana, dedicada a contribuir con la productividad y rentabilidad de la industria metalmecánica en general a través de

la fabricación y comercialización de piezas metálicas, destacándose en los acabados de pintura y galvánicos, siendo sus principales fortalezas el asesoramiento a sus clientes en la etapa de diseño y desarrollo de nuevos productos, la gestión de calidad en todos sus procesos y el respeto al medio ambiente. (Román, 2009, 35)

VISIÓN DE LA EMPRESA

Industrias Román Ltda., será una de las cinco empresas destacadas del sector Metalmecánica en el distrito capital por ofrecer soluciones integrales a sus clientes, destacándose por la calidad de sus productos, el cumplimiento en sus acuerdos comerciales y precios competitivos. (Ibídem, 2009)

* **DONDE ESTA:**

- La empresa Industrias Román LTDA se encuentra ubicada en la Avenida 1 mayo # 29-28 sur Bogota D.C.

*** UBICACIÓN DEL SOFTWARE EN LA EMPRESA:**

- Se encuentra ubicado en el departamento de Contabilidad y producción de la empresa.

*** RESEÑA HISTORICA:**

- Industrias Román Ltda., se fundó en el barrio 7 de agosto de Bogotá en 1974. Inicialmente estaba dedicada al mantenimiento técnico de las bicicletas. Poco después se empezó la fabricación de los tenedores y cañas de galápago. Con el tiempo se empezaron a fabricar los marcos y otros elementos. En 1981 se trasladó a donde hoy se encuentra (Barrio Santander), ya tenía 20 trabajadores.

Se amplió la variedad de modelos de marcos, se abrió un punto de venta allí Mismo. Las ventas ya eran a nivel nacional. La apertura económica durante el Gobierno de Cesar Gaviria condujo la empresa al borde de la quiebra, debido al

ingreso a nuestro país de productos similares a precios muy bajos. Fue en esa época en que se decidió incursionar en otros subsectores de la metalmecánica.

Los muebles metálicos tipo exportación fueron escogidos como nueva línea, esta se ha desarrollado al punto que hoy en día se ha convertido en la principal actividad de la empresa. Hoy Industrias Román es reconocida a nivel Distrito Capital como una de las más completas en servicios y calidad en este ramo. (Román., 2009, 36)

5. ALCANCES

5.1 FINALIDAD DE LA APLICACION

El modulo de control de pedidos y remisiones de la empresa Industrias Román LTDA se encuentra realizado en varias fases:

- Fase de Investigación: Se realizo visita a la empresa para reunir información acerca de su actividad económica sus necesidades a suplir con la aplicación, para ello se contó con la asesoría del DR. Mauricio Román Castañeda subgerente general.
- Fase de Análisis: Se analizo el sistema actual y determinamos sus falencias para darles una solución.
- Fase de Diseño: Basadas en el análisis del sistema actual se diseño el nuevo modulo de control de pedidos y remisiones de la empresa Industrias Román LTDA de acuerdo a las especificaciones del software implementado en la empresa y las necesidades del cliente.
- Fase de Desarrollo: Se desarrollo un prototipo donde se pueden ver el funcionamiento del modulo a implementar.

5.2 RECURSOS:

- Recursos físicos:

Un computador cuyos requisitos mínimos son:

HARDWARE

- Procesador Dual 1 GHz

- Memoria RAM de 512 MB
- Unidad lectora de CD
- Disco Duro de 80 GB
- Puertos USB
- Teclado
- Mouse
- Monitor de 15”

SOFTWARE

- Sistema operativo Microsoft Windows XP Profesional licenciado
- Versión del servidor: 5.0.51b Mysql
- Netbeans 6.8 IDE
- Dreamweaver cs4 y 8
- Microsoft Internet Explorer 7 ó equivalente licenciado.
- Herramienta IDE de desarrollo open source.

- Recursos Humanos:

-Dos programadores (Yazmín Tinjacá y Cindy Mejía).
-Dos asesores de proyecto (Universidad y Empresa).

5.3 ESPACIO:

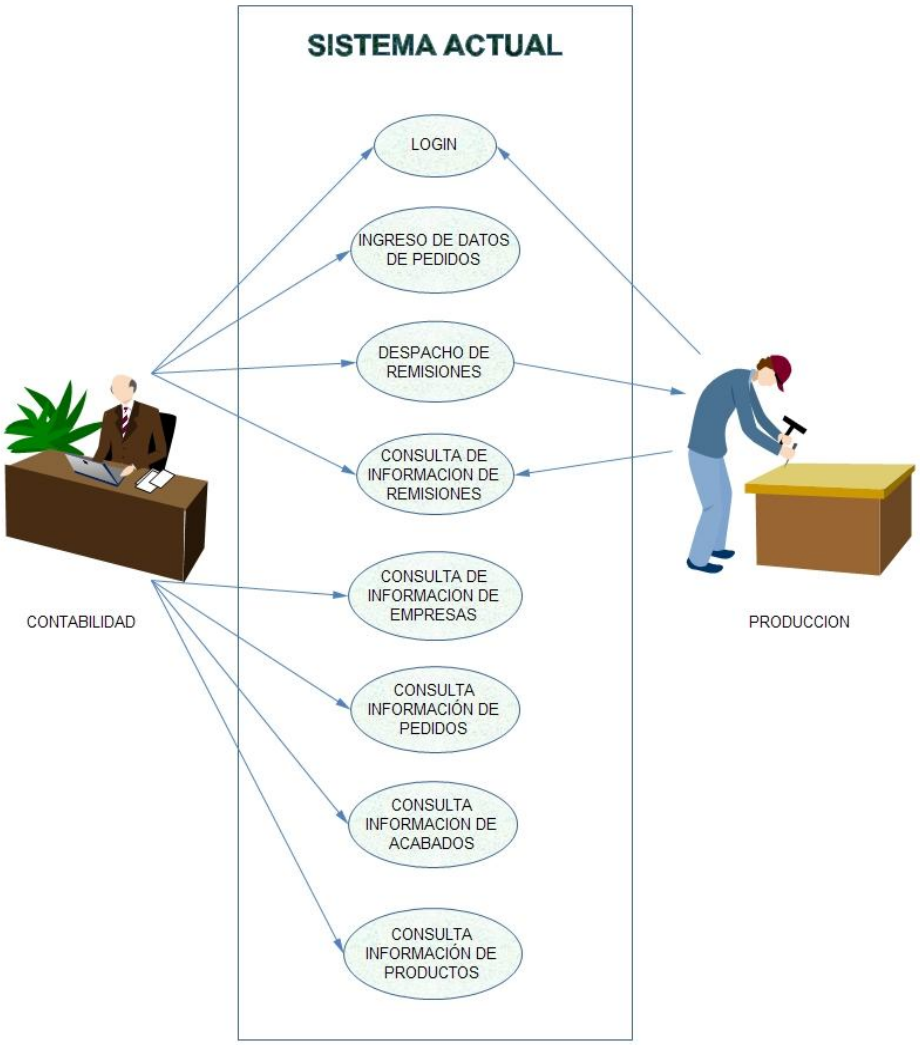
Una oficina en la empresa Industrias Román LTDA con un escritorio de 2 sillas

5.4 COSTOS:

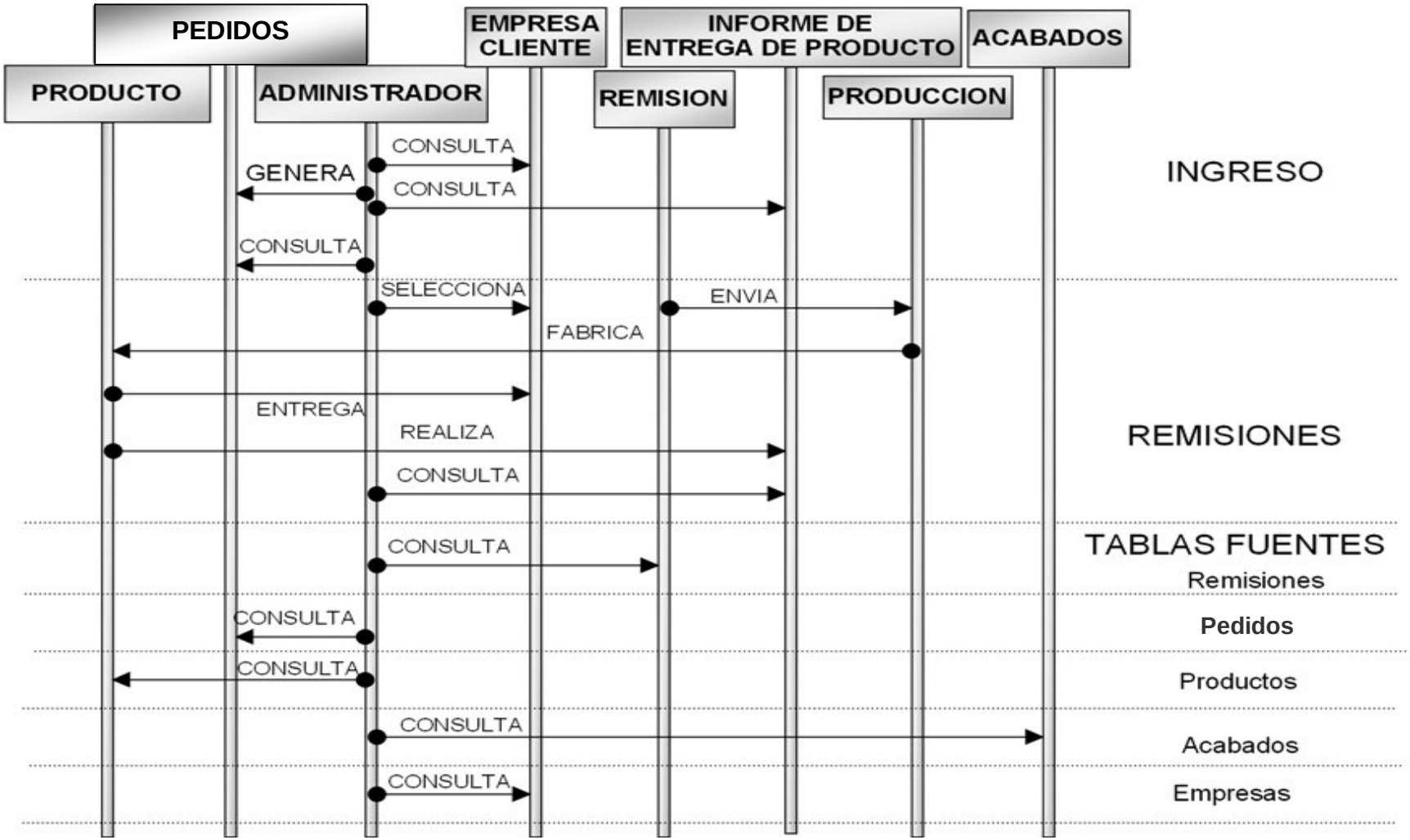
El valor aproximado de los recursos es de $5.00.000 de pesos

6. SISTEMA ACTUAL

6.1 DIAGRAMA DE CASOS DE USO

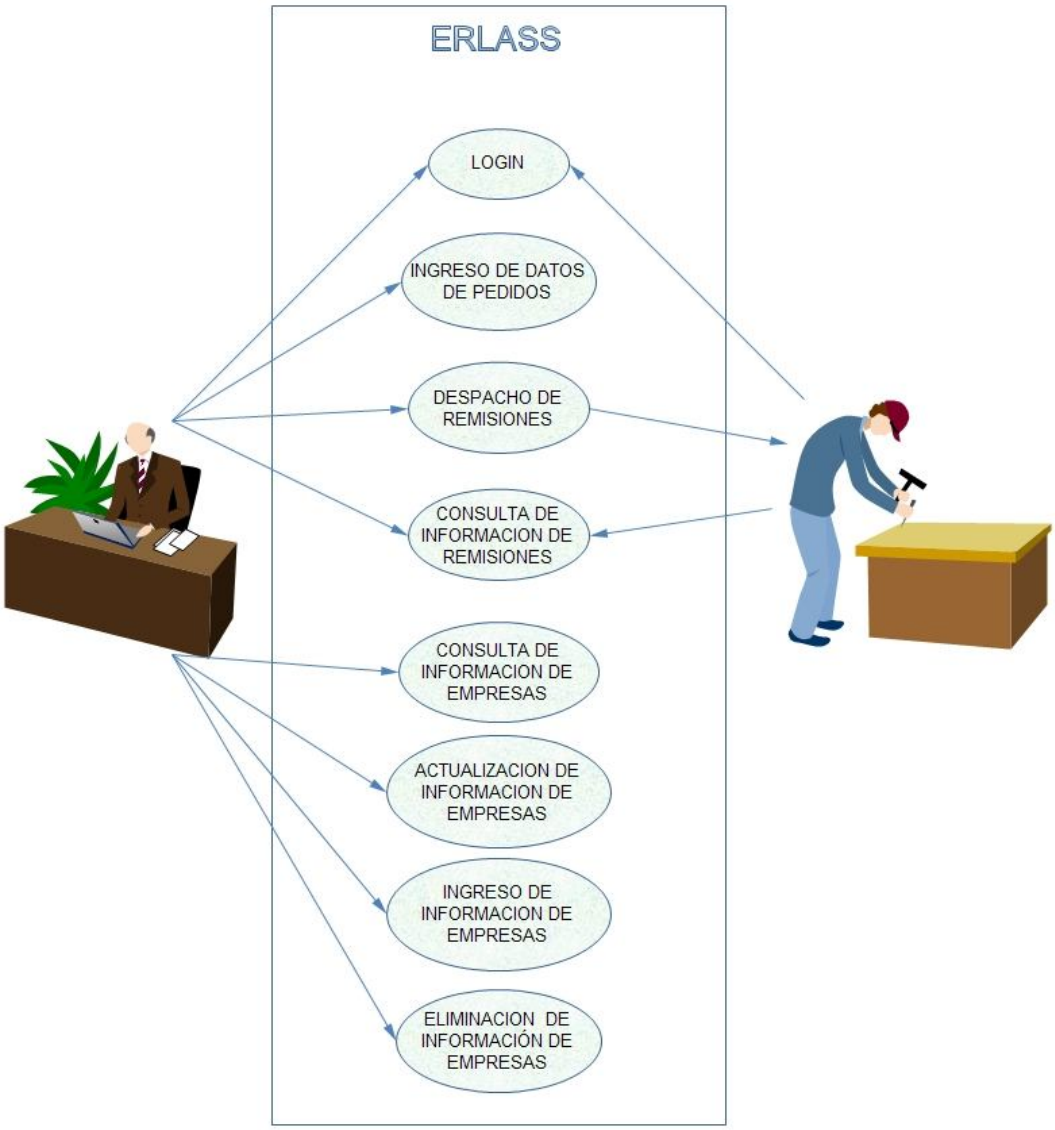


6.2 DIAGRAMA SECUENCIAL

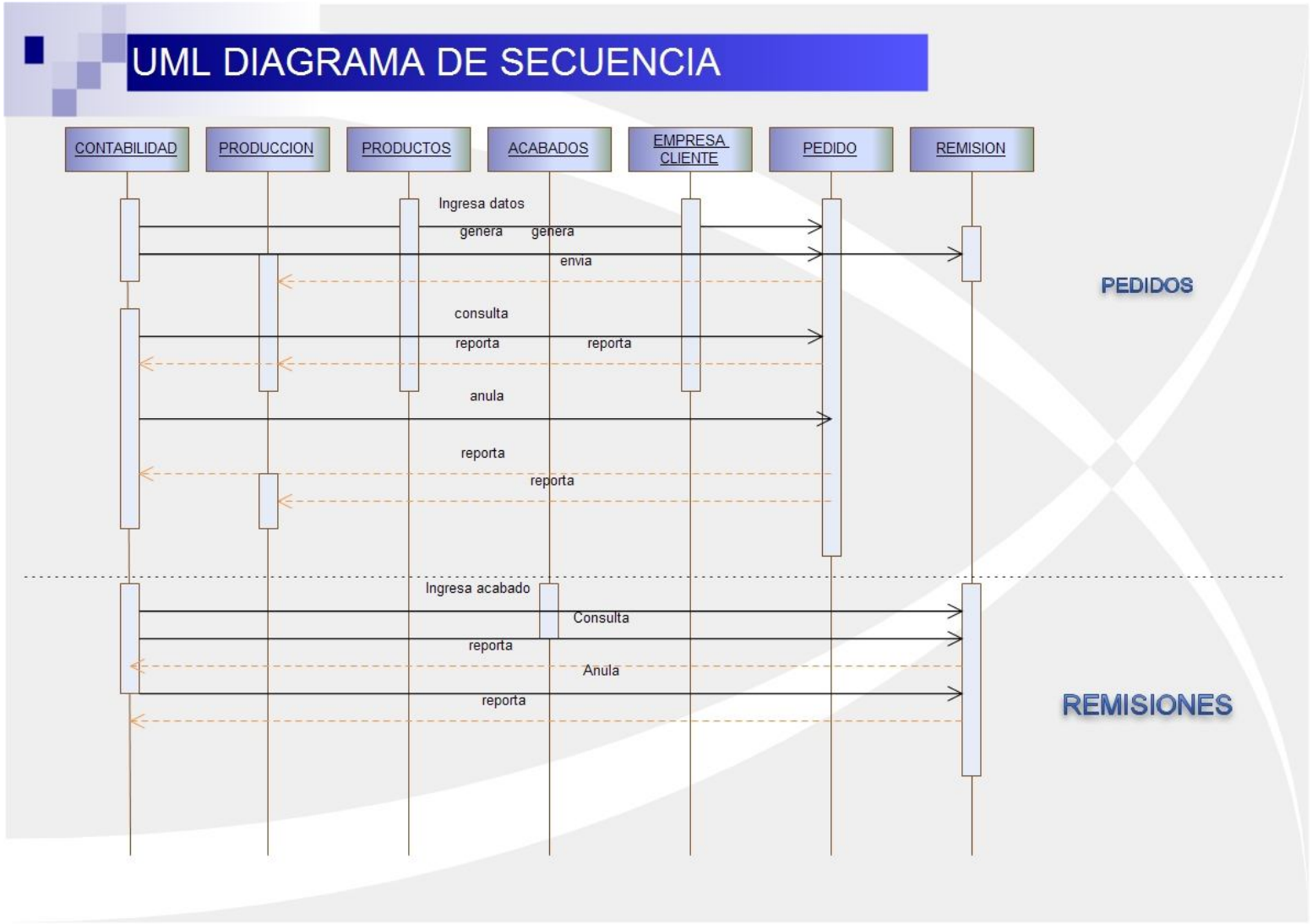


7. SISTEMA PROPUESTO

7.1 DIAGRAMA DE CASOS DE USO



7.2 DIAGRAMA SECUENCIAL



8. CRONOGRAMA:

Id	TAREA	DURACION	INICIO	FIN	PREDECESORAS	RECURSOS
1	ANTEPROYECTO	40 días	14/08/2009 9:00	08/10/2009 19:00		Cindy Mejía y Yazmin Tirjaca
2	tema	1 día	14/08/2009 9:00	14/08/2009 19:00		Cindy Mejía y Yazmin Tirjaca
3	problema	2 días	17/08/2009 9:00	18/08/2009 19:00	2	Cindy Mejía y Yazmin Tirjaca
4	Titulo	2 días	19/08/2009 9:00	20/08/2009 19:00	3	Cindy Mejía y Yazmin Tirjaca
5	objetivos	4 días	21/08/2009 9:00	26/08/2009 19:00	4	Cindy Mejía y Yazmin Tirjaca
6	justificación	8 días	27/08/2009 9:00	07/09/2009 19:00	5	Cindy Mejía y Yazmin Tirjaca
7	alcance	8 días	08/09/2009 9:00	17/09/2009 19:00	6	Cindy Mejía y Yazmin Tirjaca
8	Marco Referencial	8 días	18/09/2009 9:00	29/09/2009 19:00	7	Cindy Mejía y Yazmin Tirjaca
9	metodología	7 días	30/09/2009 9:00	08/10/2009 19:00	8	Cindy Mejía y Yazmin Tirjaca
10	PROYECTO	42 días	08/10/2009 9:00	04/12/2009 19:00		Cindy Mejía y Yazmin Tirjaca
11	requisitos	7 días	08/10/2009 9:00	16/10/2009 19:00		Cindy Mejía y Yazmin Tirjaca
12	análisis	13 días	19/10/2009 9:00	04/11/2009 19:00	11	Cindy Mejía y Yazmin Tirjaca
13	diseño	22 días	05/11/2009 9:00	04/12/2009 19:00	12	Cindy Mejía y Yazmin Tirjaca
14	asesoria1	1 día	28/10/2009 9:00	28/10/2009 19:00		Cindy Mejía y Yazmin Tirjaca
15	asesoría2	1 día	04/11/2009 9:00	04/11/2009 19:00		Cindy Mejía y Yazmin Tirjaca
16	asesoría3	1 día	11/11/2009 9:00	11/11/2009 19:00		Cindy Mejía y Yazmin Tirjaca

17	asesoría 4	1 día	18/11/2009 9:00	18/11/2009 19:00		Cindy Mejía y Yazmin Tinjaca
18	asesoría 5	1 día	25/11/2009 9:00	25/11/2009 19:00		Cindy Mejía y Yazmin Tinjaca
19	asesoría 6	1 día	02/12/2009 9:00	02/12/2009 19:00		Cindy Mejía y Yazmin Tinjaca

9. ARQUITECTURA DE LA SOLUCION DE SOFTWARE

9.1 MODELO

Para la construcción de este módulo se ha elegido dentro de los 23 modelos conocidos de desarrollo el patrón MVC (Modelo Vista Controlador), en donde los procesos se esquematizan después de un análisis de los requerimientos que debe cumplir el software y antes de la codificación de la nueva solución.

Utilizando el modelado UML que permite desarrollar aplicaciones Orientadas a Objetos OO se le dan consistencia las capas de almacenamiento (base de datos), y a la capa de presentación (interfaz grafica).

Para la capa de almacenamiento se aplican una serie de reglas y pasos que parten desde la primera forma normal (una tabla general con registros, atributos simples y una llave principal) siguiendo con la segmentación de los datos y finalizando en la quinta forma normal (Definiendo llaves foráneas y dependencias de las tablas resultantes de la segmentación.) Esto se hace con el fin de eliminar la redundancia y proteger la integridad de los datos

Para la capa de presentación se realiza un bosquejo de la estructura gráfica que va ha ser vista por el usuario basándose en las tablas finales de la base de datos resultantes de la quinta forma normal del diseño de la capa de almacenamiento.

10. ANALISIS DE LA SOLUCION DE SOFTWARE

10.1 ESTRUCTURA DE ALMACENAMIENTO

PEDIDO												
pedido		empresa				producto	detalle	producto	detalle	pedido		
Nº pedido	**Fecha**	**Razón Social**	**NIT**	**Dirección**	**Teléfono**	**Descripción**	Cantidad	Valor Unitario	Valor Total	Subtotal	IVA	Gran Total
31981	06/11/2009	Solinoff Corp S.A	800.134.773-2	Carrera 65 Nº 10-68	4463822	Tubo Eliptico 25 * 48 * 50 Base Albert	150	$6.000	$900.000	$900.000	144.000	$1.044.000
31968	06/11/2009	Solinoff Corp S.A	800.134.773-2	Carrera 65 Nº 10-68	4463822	Cromado de Base Albert	12	$25.680	$308.160	$359.520	$57.523,20	$417.043,20
31968	06/11/2009	Solinoff Corp S.A	800.134.773-2	Carrera 65 Nº 10-78	4463822	Cromado Base Trazzo * 70	2	$25.680	$51.360	$359.520	$57.523,20	$417.043,20
31987	06/11/2009	Solinoff Corp S.A	800.134.773-2	Carrera 65 Nº 10-78	4463822	Cromado de Base Albert	3	$25.680	$77.040	$89.640	$14.342,40	$103.982,40
31987	06/11/2009	Solinoff Corp S.A	800.134.773-2	Carrera 65 Nº 10-78	4463822	Cromado Tubo Union Superficie 4”	3	$4200	$12.600	$89.640	$14.342,40	$103.982,40
3154	30/10/2009	Organi.k S.A	800.228.372-7	Autpta escripc Km 14		Base Poltrona Segovia Sencilla Cromo	1	$93.882	$93.882	$351.480	$56.237	$407.722
3154	30/10/2009	Organi.k S.A	800.228.372-7	Autpta escripc Km 14		Base Poltrona Segovia Sencilla Gris	1	$53.810	$53.810	$351.480	$56.237	$407.722
3154	30/10/2009	Organi.k S.A	800.228.372-7	Autpta escripc Km 14		Base Poltrona Segovia Doble Gris	1	$60.680	$60.680	$351.480	$56.237	$407.722

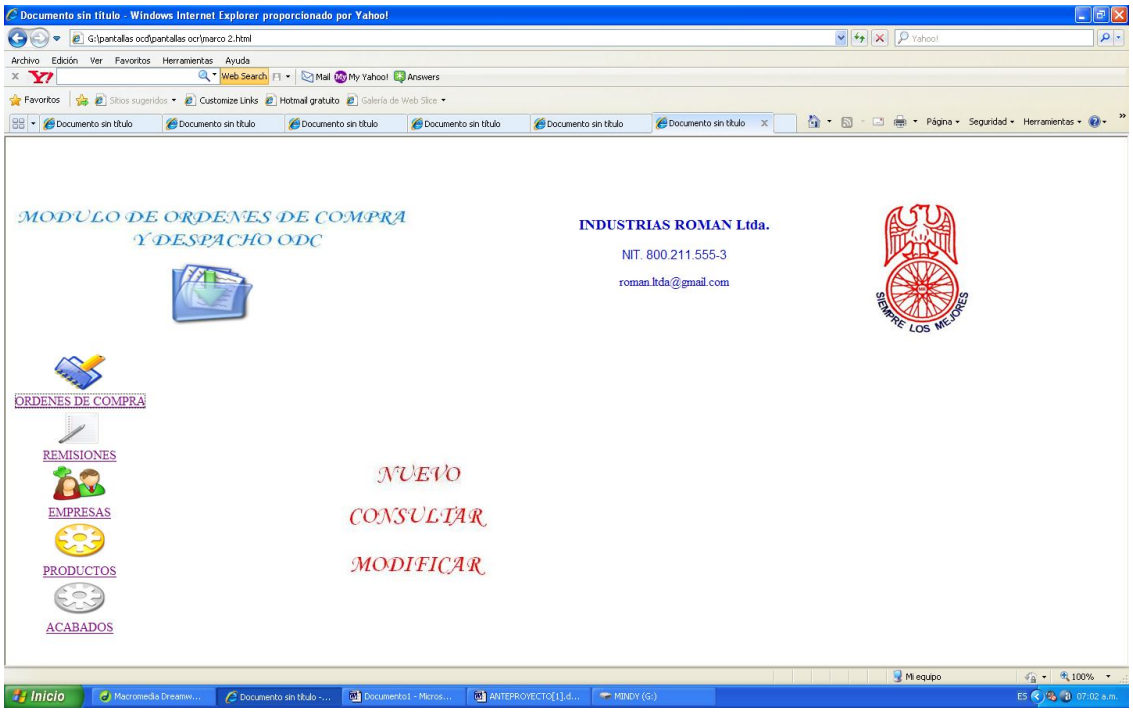
3154	30/10/2009	Organi.k SA	800.228.372-7	Autpta escripc Km14		Base Poltrona Segovia Triple Cromo	1	$143.113	$143.113	$351.480	$56.237	$407.722
3170	09/11/2009	Organi.k SA	800.228.372-7	Autpta escripc Km14		Base Poltrona Segovia Triple Cromo	1	$143.113	$143.113	$674.347	$107.895,52	$782.242,52
3170	09/11/2009	Organi.k SA	800.228.372-7	Autpta escripc Km14		Base Poltrona Segovia Semilla Cromo	4	$93.882	$375.528	$674.347	$107.895,52	$782.242,52
3170	09/11/2009	Organi.k SA	800.228.372-7	Autpta escripc Km14		Estructura Mesa Segovia Semilla de 60*60 Cromo	2	$77.853	$155.708	$674.347	$107.895,52	$782.242,52

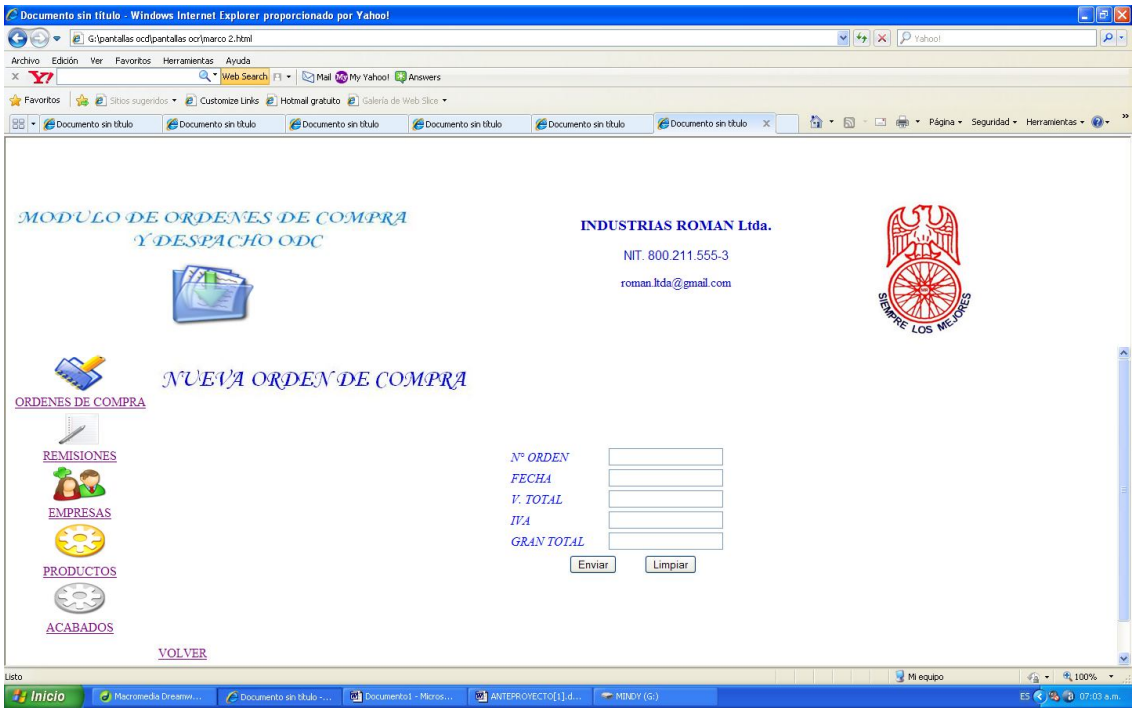
REMISIONES										
emisión		empresa				detalle		producto	Acabado	detalle
Nº Remision	**Fecha ingreso**	**Razon social**	**NIT**	**Direccion**	**Telefono**	**Nº Orden Compra**	**Fecha Despacho**	**Descripcion**	**Acabado**	**Cantidad**
13150	16/11/2009	Solinoff Corp SA	800.134.773-2	Carrera 65 Nº 10-68	4463822	31981	16/11/2009	Tubo Eliptico 25*48*50 Base Albert	Cromo	150
13151	16/11/2009	Solinoff Corp SA	800.134.773-2	Carrera 65 Nº 10-68	4463822	31986	16/11/2009	Cromado de Base Albert	Cromo	12
13152	16/11/2009	Solinoff Corp SA	800.134.773-2	Carrera 65 Nº 10-68	4463822	31986	16/11/2009	Cromado Base Trazzo*70	Cromo	2
13152	16/11/2009	Solinoff Corp SA	800.134.773-2	Carrera 65 Nº 10-68	4463822	31987	16/11/2009	Cromado de Base Albert	Cromo	3
13152	16/11/2009	Solinoff	800.134.77	Carrera 65	4463822	31987	16/11/200	Cromado Tubo	Cromo	3

		Corp SA	3-2	Nº 1068			9	Union Superficie 4'		
13146	05/11/2009	Organi. k SA	800228837 2-7	Autpta escripc Km 14		3154	05/11/2009	Base Poltrona Segovia Sandilla Cromo	Cromo	1
13147	05/11/2009	Organi. k SA	800228837 2-7	Autpta escripc Km 14		3154	05/11/2009	Base Poltrona Segovia Sandilla Gris	Cromo	1
13147	05/11/2009	Organi. k SA	800228837 2-7	Autpta escripc Km 14		3154	05/11/2009	Base Poltrona Segovia Doble Gris	Aluminio	1
13147	05/11/2009	Organi. k SA	800228837 2-7	Autpta escripc Km 14		3154	05/11/2009	Base Poltrona Segovia Triple Cromo	Aluminio	1
13149	11/11/2009	Organi. k SA	800228837 2-7	Autpta escripc Km 14		3170	11/11/2009	Base Poltrona Segovia Triple Cromo	Cromo	1
13149	11/11/2009	Organi. k SA	800228837 2-7	Autpta escripc Km 14		3170	11/11/2009	Base Poltrona Segovia Sandilla Cromo	Cromo	4
13149	11/11/2009	Organi. k SA	800228837 2-7	Autpta escripc Km 14		3170	11/11/2009	Estructura Mesa Segovia Sandilla de 60*60 Cromo	Cromo	2

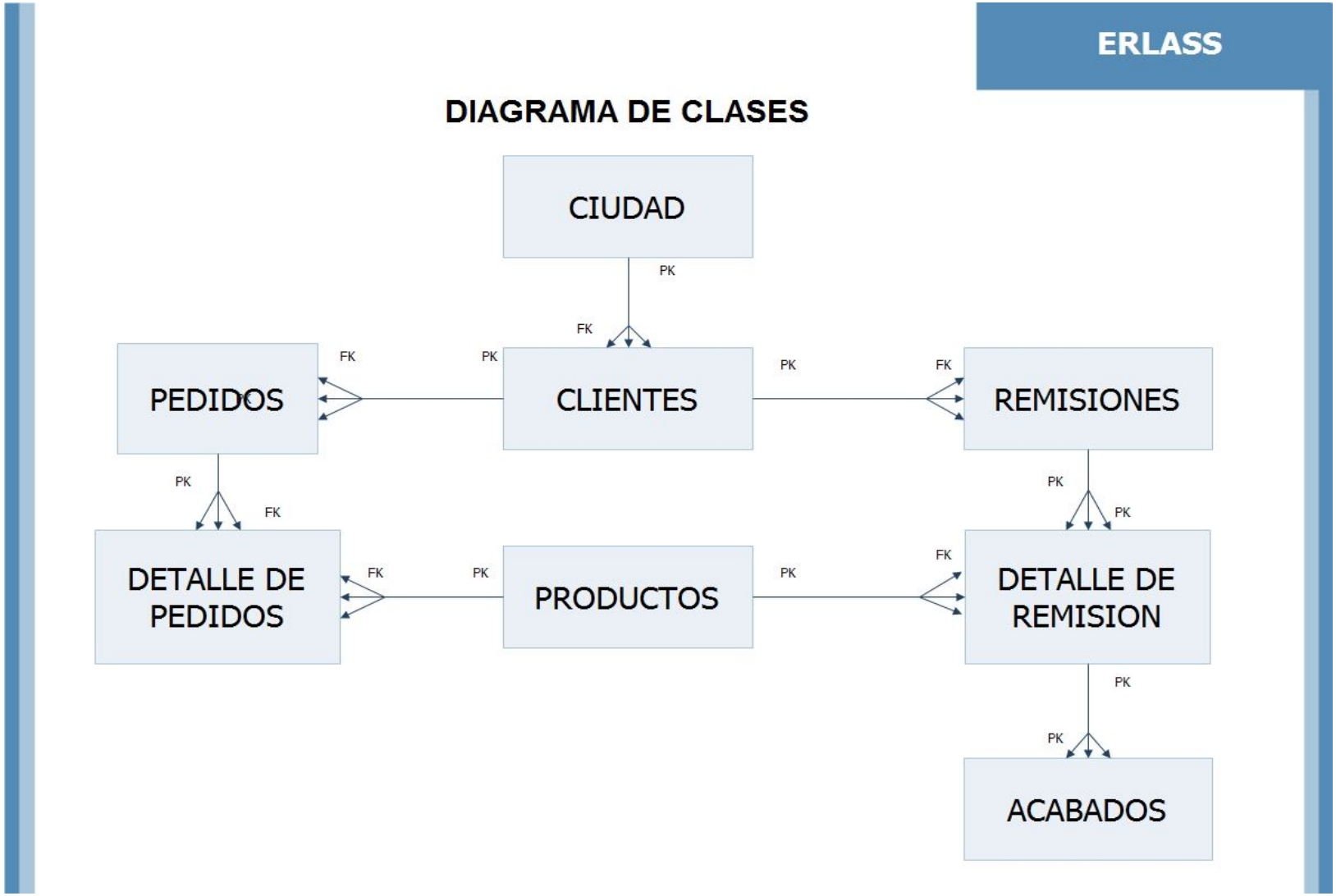
10.2 INTERFAZ GRÁFICA DE USUARIO

El análisis de interfaz gráfica permite ver tres paneles superior, izquierdo y central; en el superior se puede observar los logos de la aplicación y de la empresa, en el panel izquierdo el menú principal y en el panel central el contenido, formularios o submenús dependiendo de la página.





10.3 INTERFAZ DE PROCESAMIENTO DE DATOS



11. DISEÑO DE LA SOLUCION DE SOFTWARE

11.1 ESTRUCTURA DE ALMACENAMIENTO

CLIENTES				
IDCliente	Razón Social	NIT	Dirección	Teléfono
1	Stdinoff Corp SA	800.134.773-2	Carrera 65 Nº 10-68	4463822
2	Organi.k SA	800.228.372-7	Autpta. escripc Km 14	

REMISION			
IDRemision	Nº Remision	Fecha ingreso	cod_cliente
1	13150	16/11/2009	1
2	13151	16/11/2009	1
3	13152	16/11/2009	1
4	13146	05/11/2009	2
5	13147	05/11/2009	2
6	13149	11/11/2009	2

DETALLE DE REMISION						
ID Detalle	N Pedido	Fecha Despacho	Cantidad	cod_remision	cod_producto	cod_acabado
1	31981	16/11/2009	150	1	1	1
2	31986	16/11/2009	12	2	2	2
3	31986	16/11/2009	2	3	3	2
4	31987	16/11/2009	3	3	2	2
5	31987	16/11/2009	3	3	4	2
6	3154	05/11/2009	1	4	5	2
7	3154	05/11/2009	1	5	6	2
8	3154	05/11/2009	1	5	7	3
9	3154	05/11/2009	1	5	8	3
10	3170	11/11/2009	1	6	8	2
11	3170	11/11/2009	4	6	5	2
12	3170	11/11/2009	2	6	9	2

PRODUCTO		
ID Producto	Descripcion	Valor Unitario
1	Tubo Eliptico 25*45*50 Base Albert	$6.000
2	Cromado de Base Albert	$25.680
3	Cromado Base Trazzo*70	$25.680
4	Cromado Tubo Union Superficie 4'	$4200
5	Base Poltrona Segovia Semilla Cromo	$93.882
6	Base Poltrona Segovia Semilla Gris	$53.810
7	Base Poltrona Segovia Doble Gris	$60.680
8	Base Poltrona Segovia Triple Cromo	$143.113
9	Estructura Mesa Segovia Semilla de 60*60 Cromo	$77.853

ACABADO	
IDDetalle	Acabado
1	Cromo
2	Cromo
3	Aluminio

PEDIDO						
ID pedido	N° pedido	Fecha	Subtotal	IVA	Gran Total	cod_cliente
1	31981	06/11/2009	$900.000	144.000	$1.044.000	1
2	31983	06/11/2009	$359.520	$57.523,20	$417.043,20	1
3	31987	06/11/2009	$89.640	$14.342,40	$103.982,40	1
4	3154	30/10/2009	$351.480	$56.237	$407.722	2
5	3170	09/11/2009	$674.347	$107.895,52	$782.242,,52	2

DETALLE DE PEDIDO				
IDDetalle	Cantidad	Valor Total	cod_pedido	cod_producto
1	150	$900.000	1	1
2	12	$308.160	2	2
3	2	$51.360	2	3
4	3	$77.040	3	2
5	3	$12.600	3	4
6	1	$93.882	4	5
7	1	$53.810	4	6
8	1	$60.680	4	7
9	1	$143.113	4	8
10	1	$143.113	5	8
11	4	$375.528	5	5
12	2	$155.708	5	9

DICCIONARIO DE DATOS PEDIDOS Y REMISIONES

Nombre de la tabla: Clientes							
Descripción: En esta tabla se encuentra toda la información de los clientes de las empresas a las que se les realiza las remisiones							
Campo	**Tipo de dato**	**Longitud**	**Clave**	**Unicidad**	**Obligatorio**	**Indexado**	**Descripción**
ID_Cliente	Numérico	11	PK	Si	Si	Si	Campo identificador de cada cliente, este campo es auto incrementable
Razón_Social	Texto	20		No	Si	Si	Nombre de la Empresa
NIT	Texto	10		Si	Si	Si	campo identificador del numero de cada empresa
Dirección	Texto	25		No	Si	No	Dirección de la empresa a la que se le genera la remisión
Teléfono	Texto	10		No	Si	No	Teléfono de la empresa a la que se le genera la remisión

Nombre de la tabla: Remisión							
Descripción: En esta tabla se encuentra la información de las remisiones que se generan a las empresas							
Campo	**Tipo de dato**	**Longitud**	**Clave**	**Unicidad**	**Obligatorio**	**Indexado**	**Descripción**
ID_Remisión	Numérico	11	PK	Si	Si	Si	Campo identificador de cada Remisión, este campo es auto incrementable
Nº Remisión	Texto	30		No	Si	Si	Campo identificador del numero de cada remisión
Fecha ingreso	Texto	10		No	Si	No	Este campo indica la fecha de ingreso de la remisión
cod_cliente	Numérico	11	FK	No	Si	Si	Campo identificador de cada cliente, en esta tabla este dato se puede repetir

Nombre de la tabla: Producto							
Descripción: En esta tabla se encuentran todos los productos que se generan en la empresa							
Campo	**Tipo de dato**	**Longitud**	**Clave**	**Unicidad**	**Obligatorio**	**Indexado**	**Descripción**
ID_Producto	Numerico	11	PK	Si	Si	Si	Campo identificador de cada Producto, este campo es auto incrementable
Descripción	Texto	30		Si	Si	Si	Campo identificador de la descripción de los productos
Valor Unitario	Numerico	10		No	Si	No	Valor Unitario de la remisión que se va a generar

Nombre de la tabla: Acabado

Descripción: En esta tabla se encuentran todos los acabados de los productos que se generan en la empresa

Campo	Tipo de dato	Longitud	Clave	Unicidad	Obligatorio	Indexado	Descripción
ID_Acabado	Numerico	11	PK	Si	Si	Si	Campo identificador de cada Acabado, este campo es autoincrementable
Descripcion	Texto	30		Si	Si	Si	Campo identificador de la descripción de los Acabados

Nombre de la tabla: Detalle de pedido

escripción: En esta tabla se muestra la cantidad de productos que se generaron en el pedido y el subtotal respectivo

Campo	Tipo de dato	Longitud	Clave	Unicidad	Obligatorio	Indexado	Descripción
ID_Detalle	Numerico	11	PK	Si	Si	Si	Campo identificador de cada detalle del pedido, este campo es autoincrementable
Cantidad	Texto	5		No	Si	No	Cantidad de productos generados en el pedido
Subtotal	Numerico	10		No	Si	No	Subtotal del pedido que se va a generar
cod_pedido	Numerico	11	FK	No	Si	Si	Campo identificador de cada pedido, En esta tabla se puede repetir
cod_producto	Numerico	11	FK	No	Si	Si	Campo identificador de cada producto, en esta tabla se puede repetir

Nombre de la tabla: Detalle de remisiones

escripción: En esta tabla se muestra la cantidad de productos que se generaron en la remisión y el subtotal respectivo

Campo	Tipo de dato	Longitud	Clave	Unicidad	Obligatorio	Indexado	Descripción
ID_Detalle	Numérico	11	PK	Si	Si	Si	Campo identificador de cada detalle de la orden de compra, este campo es auto incrementable
Cantidad	Texto	5		No	Si	No	Cantidad de

							productos generados en la orden de compra
cod_remision	Numérico	11	FK	No	Si	Si	Campo identificador de cada orden de compra, En esta tabla se puede repetir
cod_producto	Numérico	11	FK	No	Si	Si	Campo identificador de cada producto, en esta tabla se puede repetir
cod_acabado	Numérico	11	FK	No	Si	Si	Campo identificador de cada producto, en esta tabla se puede repetir

Nombre de la tabla: pedido

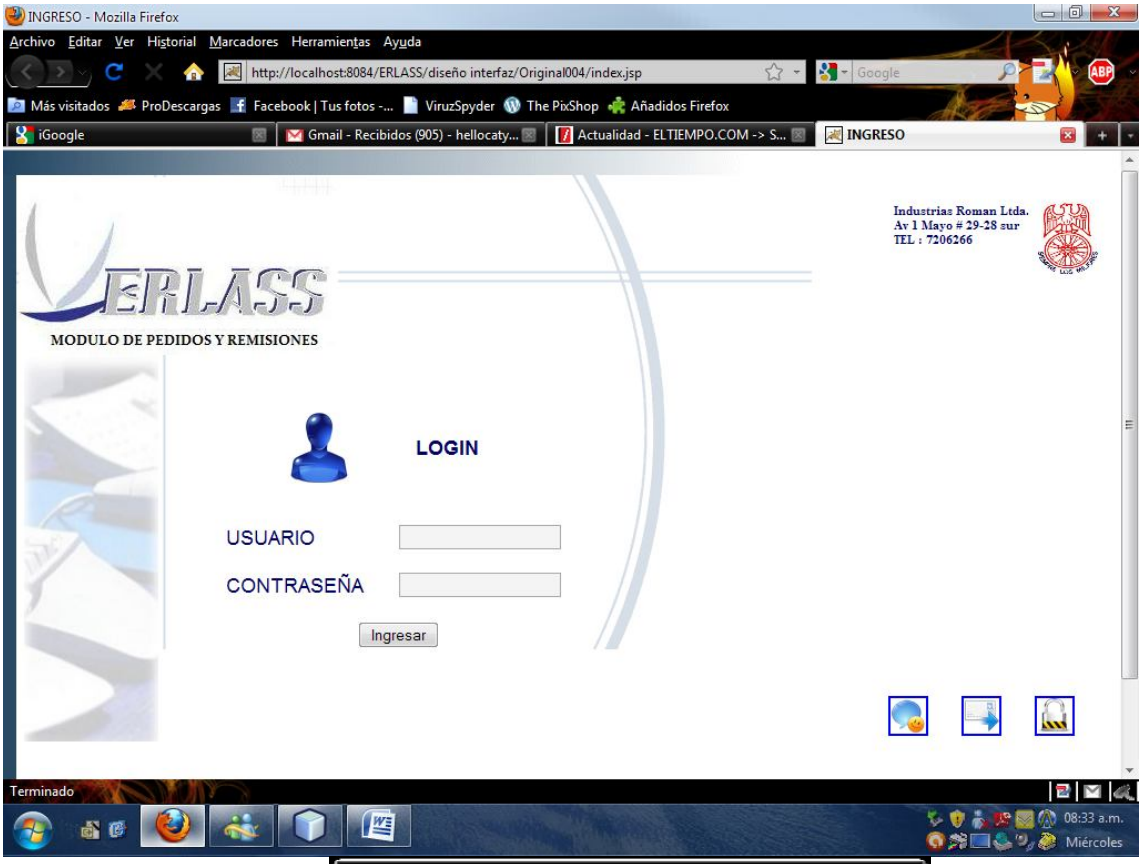
Descripción: En esta tabla se encuentra toda la información acerca del total del pedido generado a las diferentes empresas

Campo	Tipo de dato	Longitud	Clave	Unicidad	Obligatorio	Indexado	Descripción
ID_pedido	Numérico	11	FK	Si	Si	Si	Campo identificador de cada pedido, este campo es autoincrementable
Nº pedido	Texto	8		Si	Si	Si	Campo identificador del numero de cada pedido
Fecha	Texto	8		No	Si	No	Fecha en la que se realiza la pedido
Valor Total	Numérico	10		No	Si	No	Valor total del pedido que se va a generar
IVA	Numérico	10		No	Si	No	Valor del impuesto del valor agregado
Gran Total	Numérico	10		No	Si	No	Valor del pedido completo
cod_cliente	Numérico	11	FK	No	Si	Si	Campo identificador de cada cliente, en esta tabla este dato se puede repetir

11.2 INTERFAZ GRÁFICA DE USUARIO

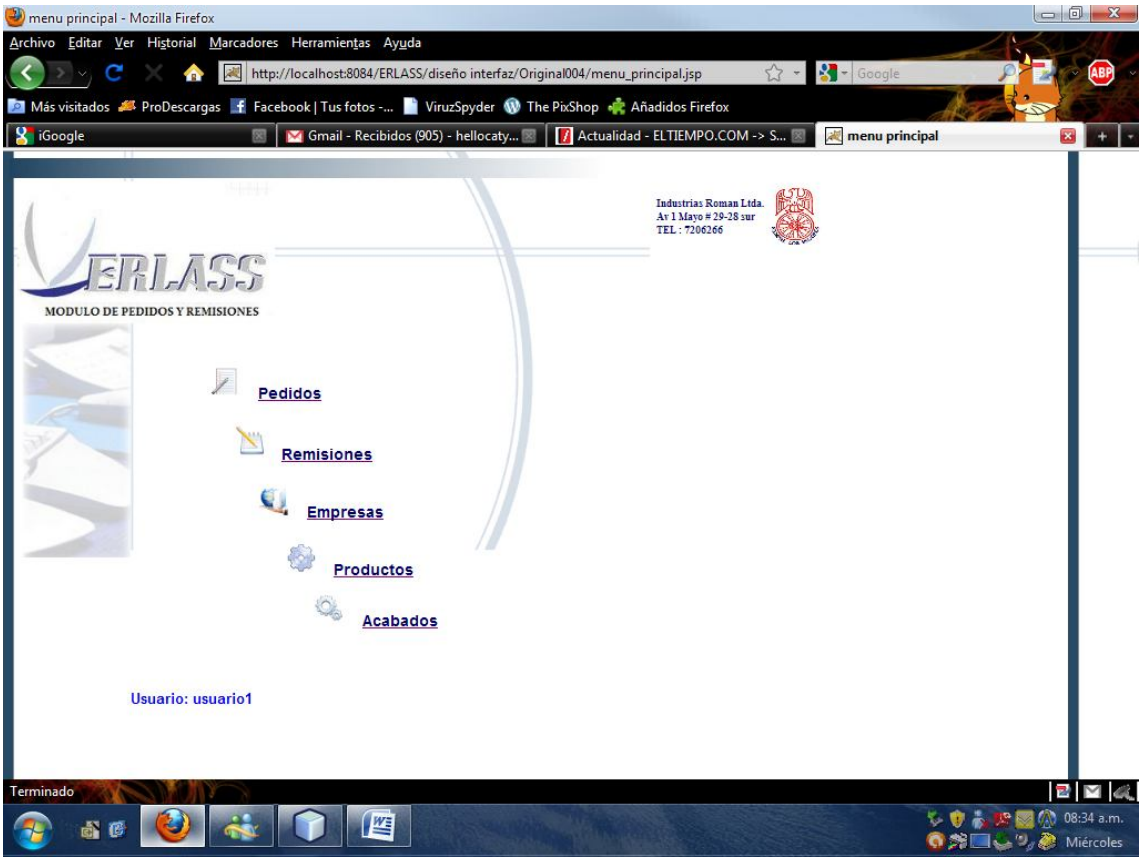
- Ingreso al sistema

La primera parte de ingreso al sistema es el logeo, en esta se debe digitar un usuario y contraseña, generado por el administrador general de la herramienta.



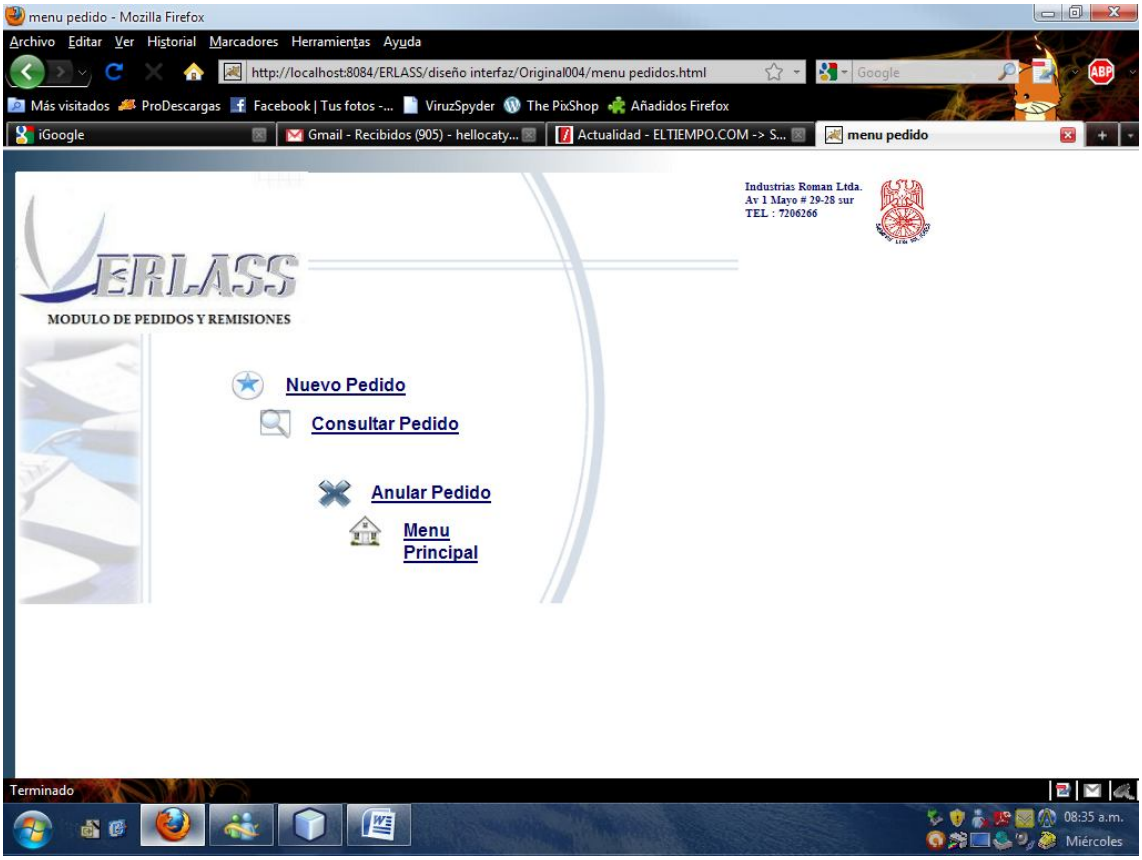
- Menú Principal:

En esta parte del sistema el administrador observa el menú principal del software y puede elegir el sub-módulo al que quiere ingresar.



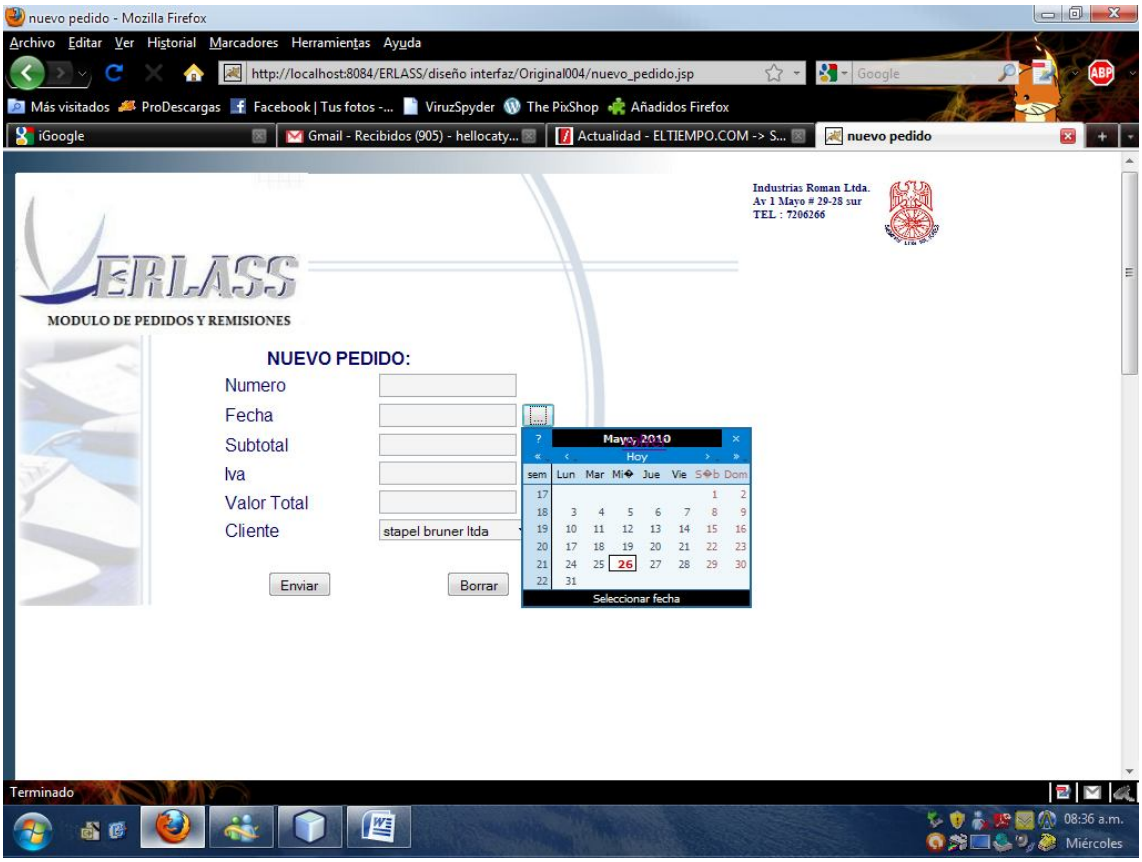
- Menú Pedido:

En esta parte del sistema el administrador puede observar el menú de pedido y puede elegir el sub-módulo al que quiere ingresar.



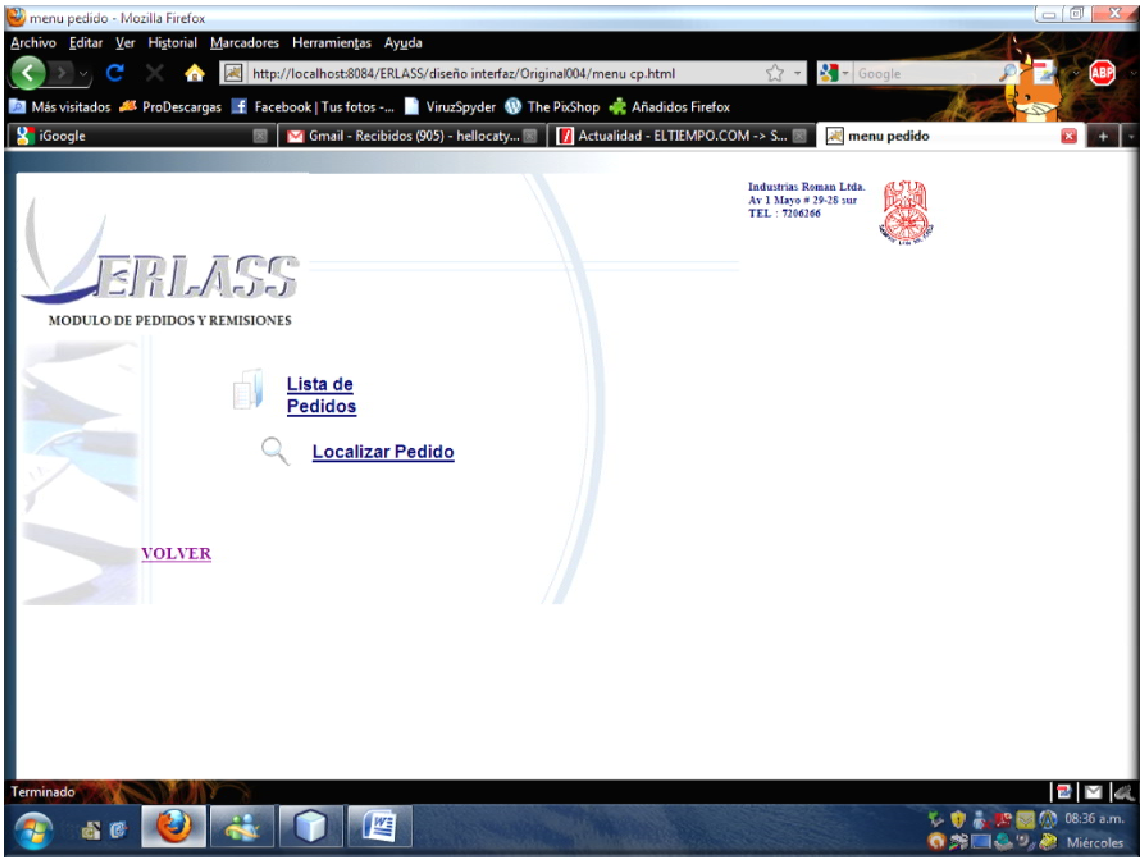
- Nuevo Pedido:

Después de que el administrador seleccione la opción nuevo pedido, aparece una pantalla donde se pude digitar la información, igualmente se genera un calendario donde será mas fácil y cómodo para el usuario seleccionar la fecha que desea.



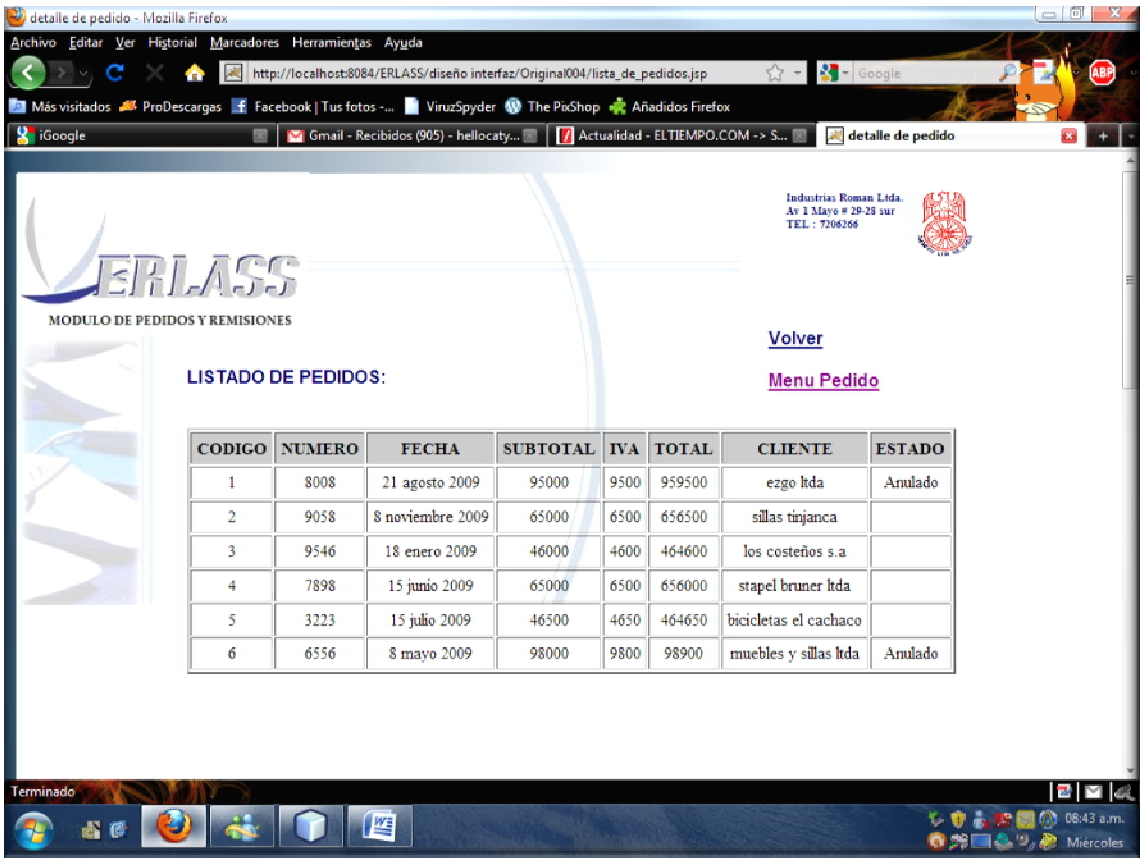
- Consultar Pedido:

Cuando el administrador haya ingresado su nuevo pedido, podrá consultar si realmente fue creado aquí se encuentran dos opciones por medio de las cuales el puede observar la información guardada en la base de datos ya sea por medio de la opción lista de pedido o localizar pedido:



- Consultar Pedido**:**

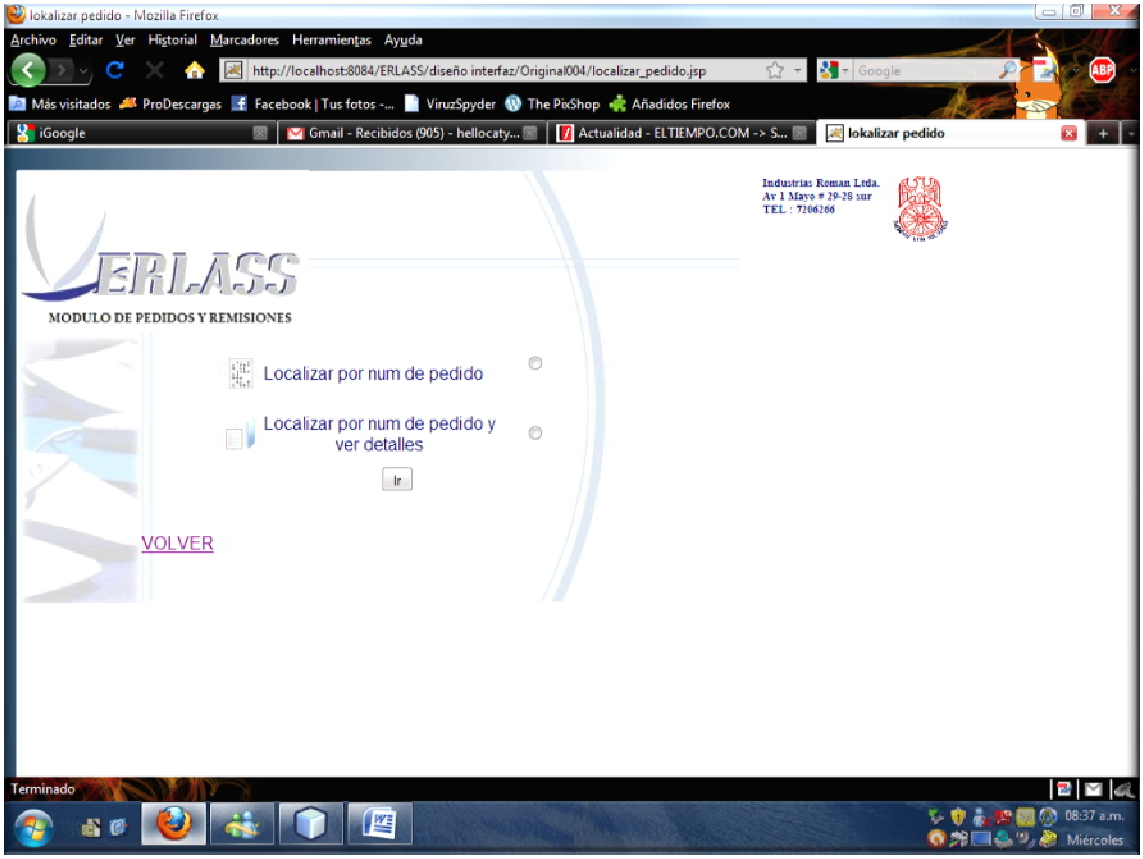
En esta parte del sistema el administrador puede consular los pedidos que se encuentran en la base de datos por medio de un listado.

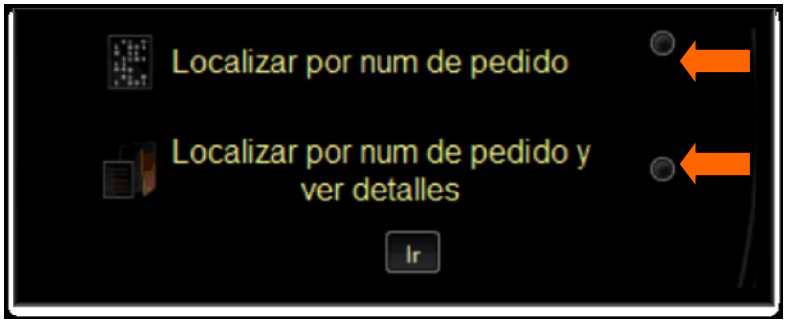


LISTADO DE PEDIDOS:

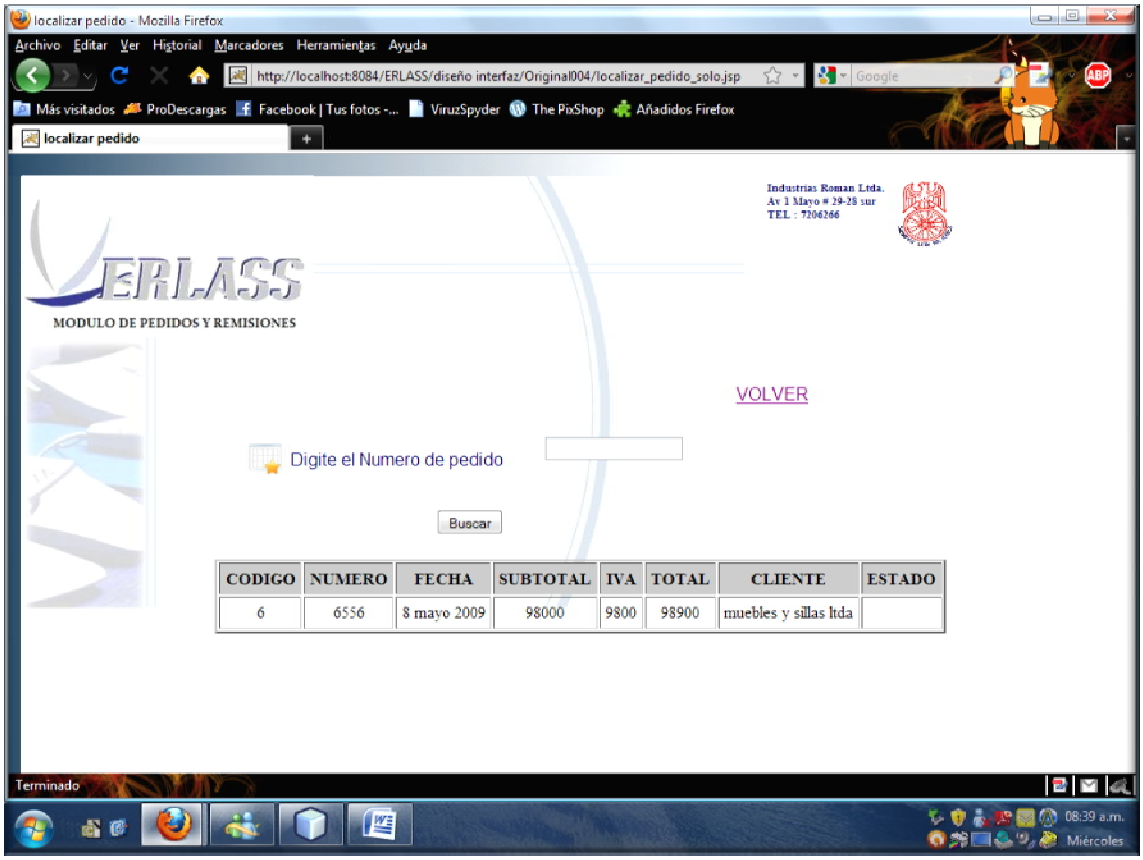
CODIGO	NUMERO	FECHA	SUBTOTAL	IVA	TOTAL	CLIENTE	ESTADO
1	8008	21 agosto 2009	95000	9500	959500	ezgo ltda	Anulado
2	9058	8 noviembre 2009	65000	6500	656500	sillas tinjanca	
3	9546	18 enero 2009	46000	4600	464600	los costeños s.a	
4	7898	15 junio 2009	65000	6500	656000	stapel bruner ltda	
5	3223	15 julio 2009	46500	4650	464650	bicicletas el cachaco	
6	6556	8 mayo 2009	98000	9800	98900	muebles y sillas ltda	Anulado

En este submenú observamos dos opciones las cuales podrá elegir el administrador para realizar su consulta más específicamente, es así como podrá localizar el pedido por medio del número o por el número y el detalle del mismo:



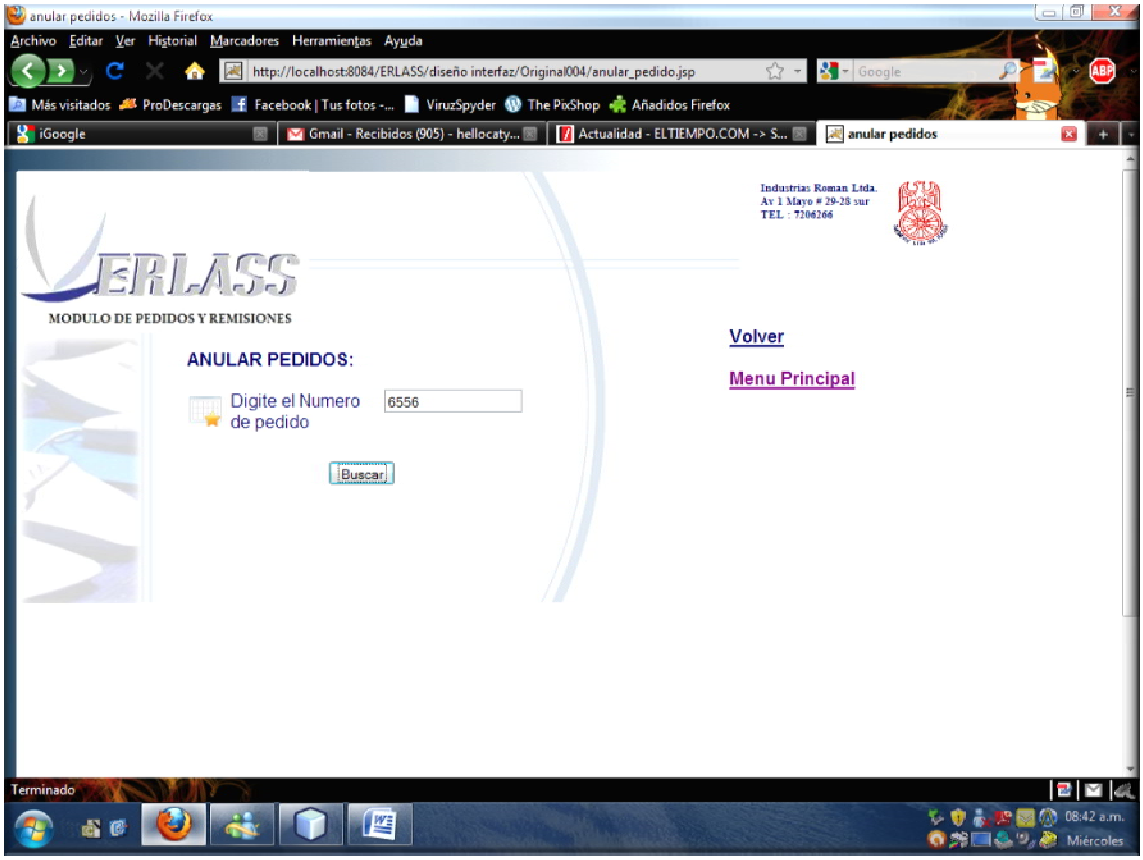


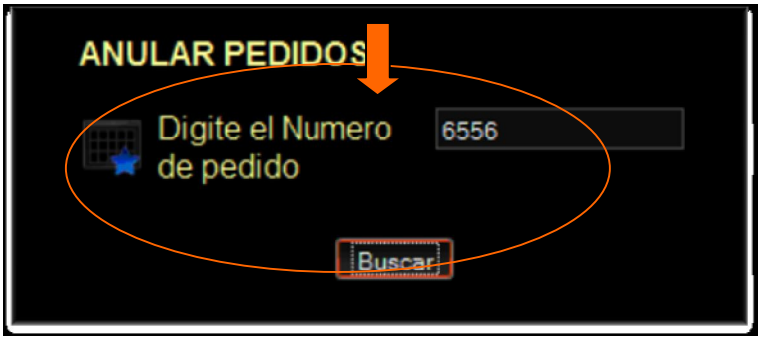
El administrador podrá observar en esta parte del sistema el pedido ingresando el número del mismo y se desplegara una tabla con los detalles del número del pedido ingresado:



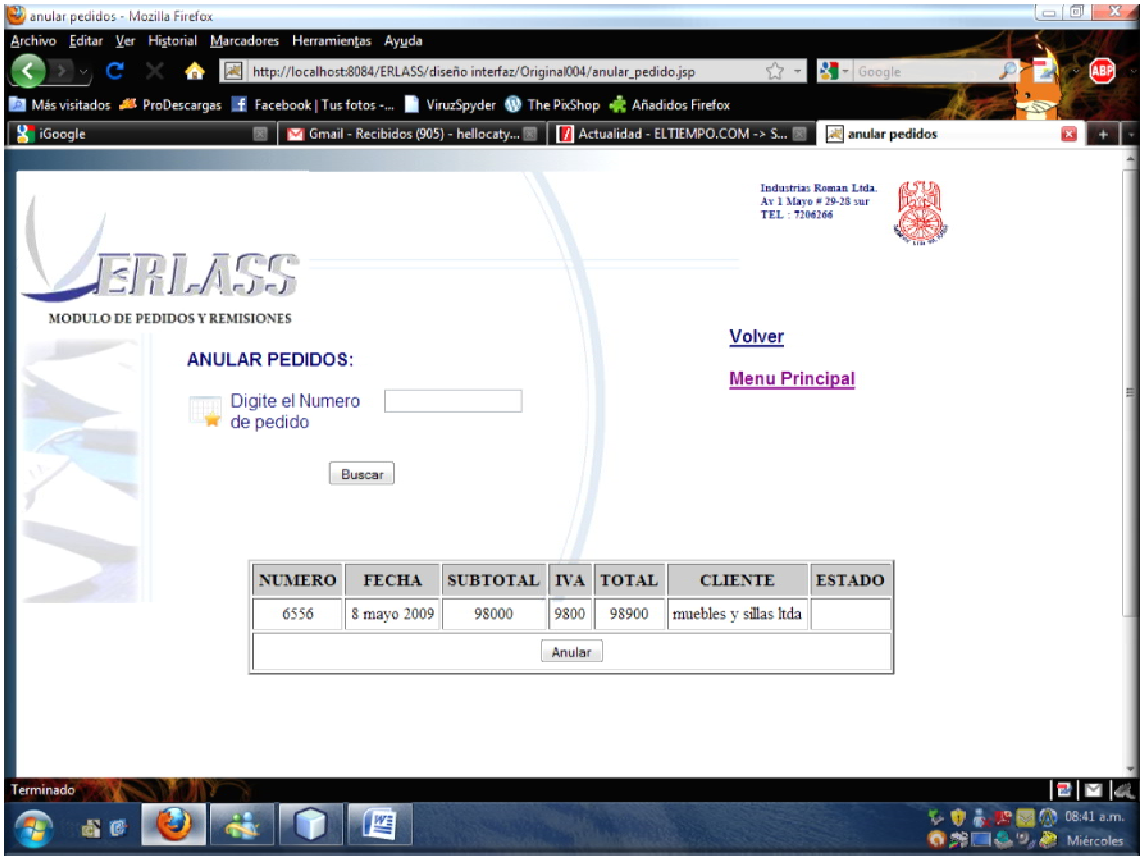
- Anular Pedido**:**

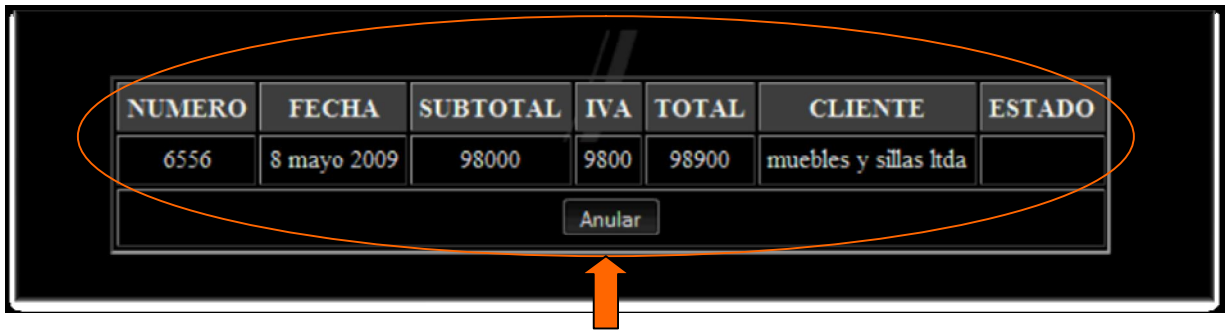
Después de que el administrador haya ingresado su nuevo pedido lo haya consultado tendrá la opción de poder anularlo en caso de que la información haya sido errónea, en este caso el administrador digita el número del pedido que quiere anular y lo busca:



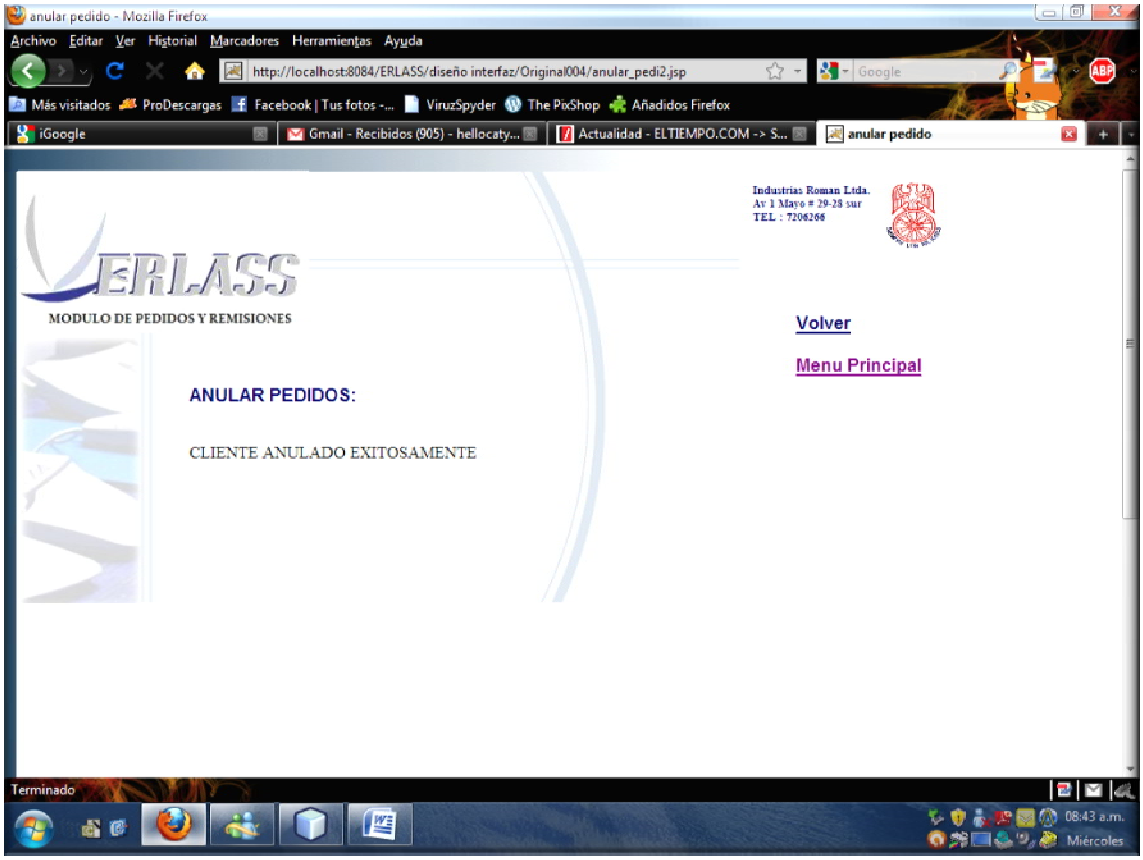


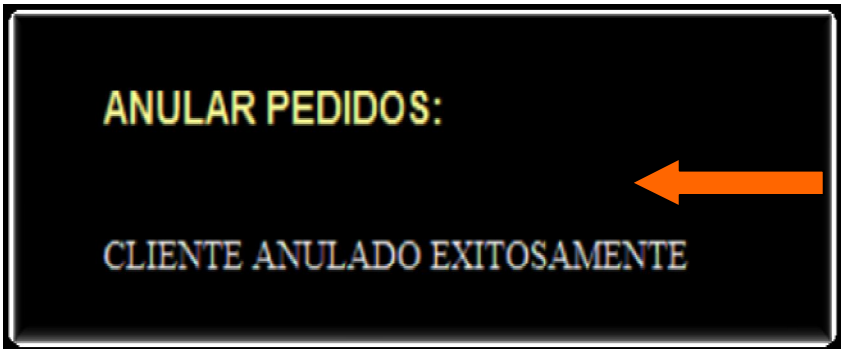
El sistema busca el pedido y se despliega una tabla que muestra los detalles del pedido que se desea anular, allí el administrador observa la información si esta es errónea elige la opción anular:





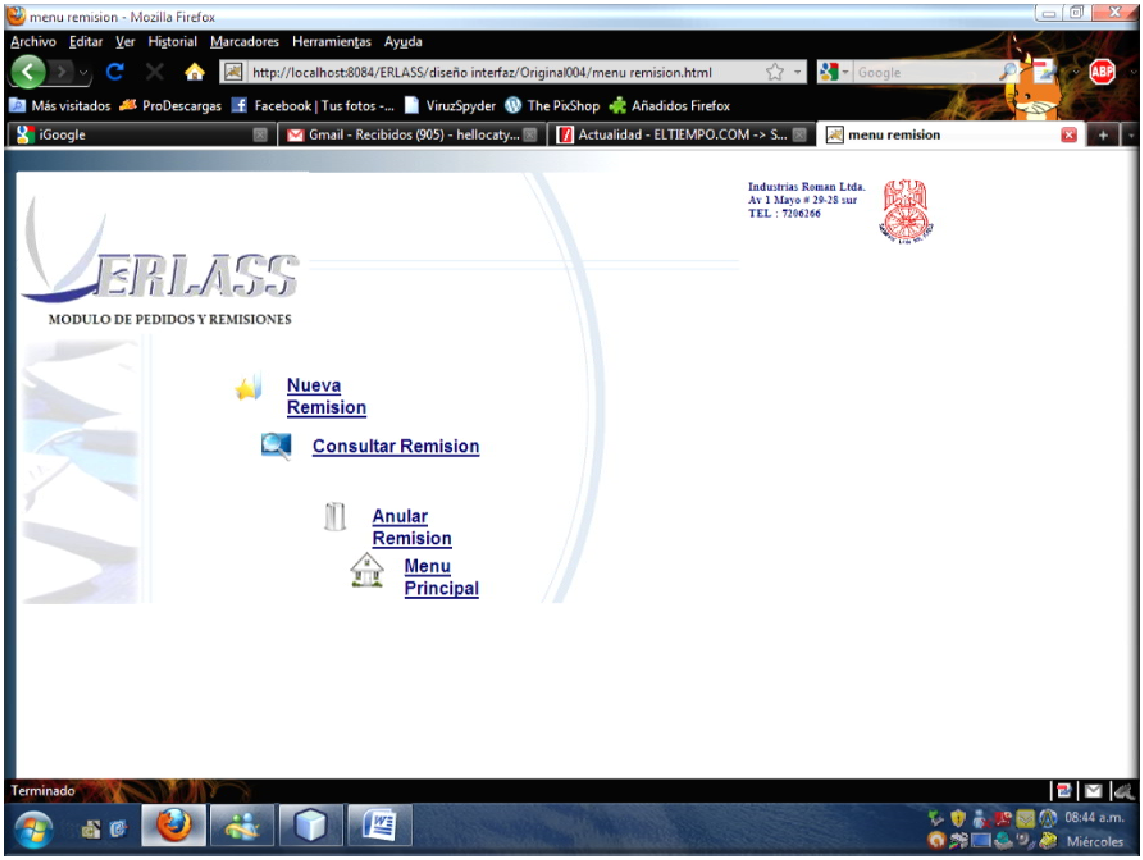
Después el sistema le muestra al administrador que su pedido ha sido anulado satisfactoriamente como se observa:

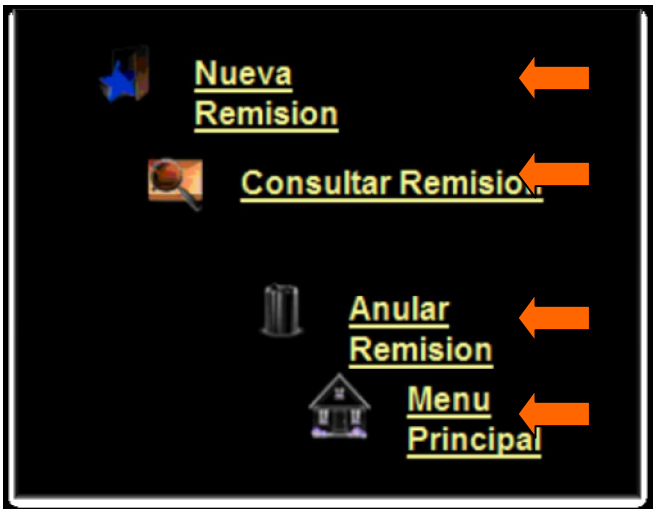




- Menú Remisión:

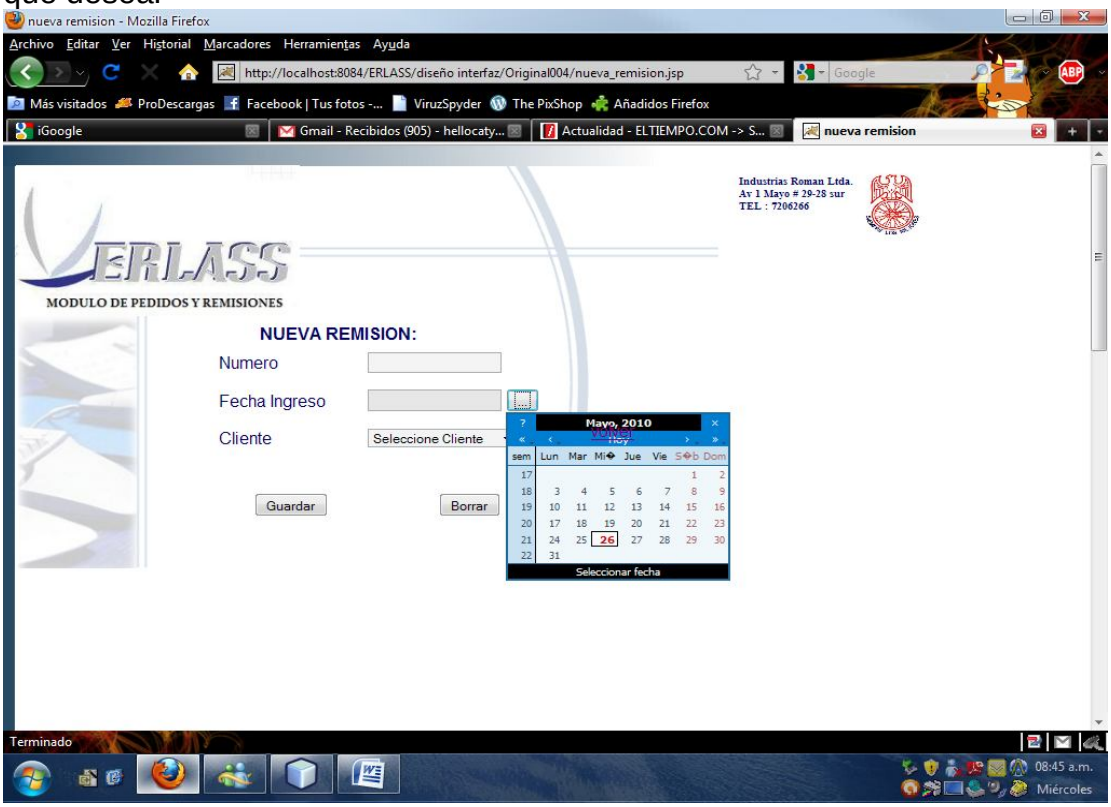
En esta parte del sistema el administrador puede observar el menú de remisión y puede elegir el sub-módulo al que quiere ingresar.





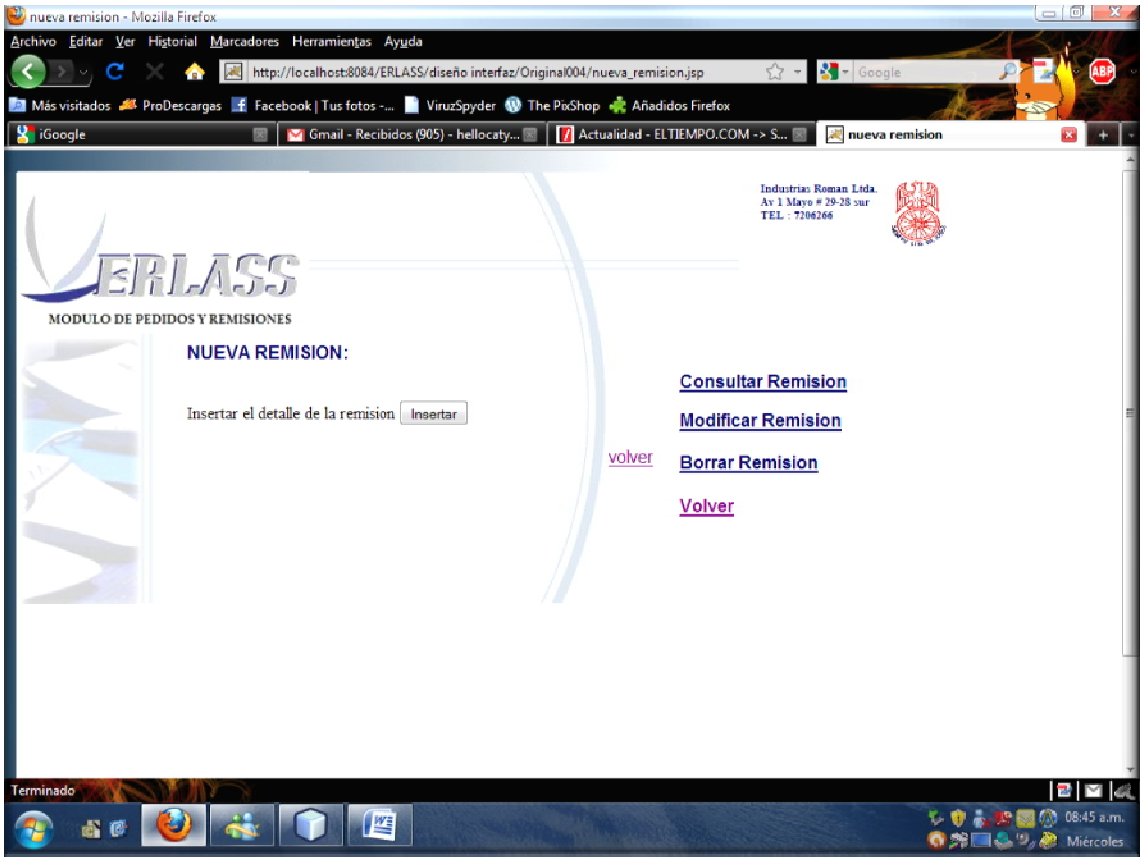
- Nueva Remisión:

Después de que el administrador seleccione la opción nueva remisión, el sistema muestra una pantalla donde se pude digitar la información, igualmente se genera un calendario donde será mas fácil y cómodo para el administrador seleccionar la fecha que desea.



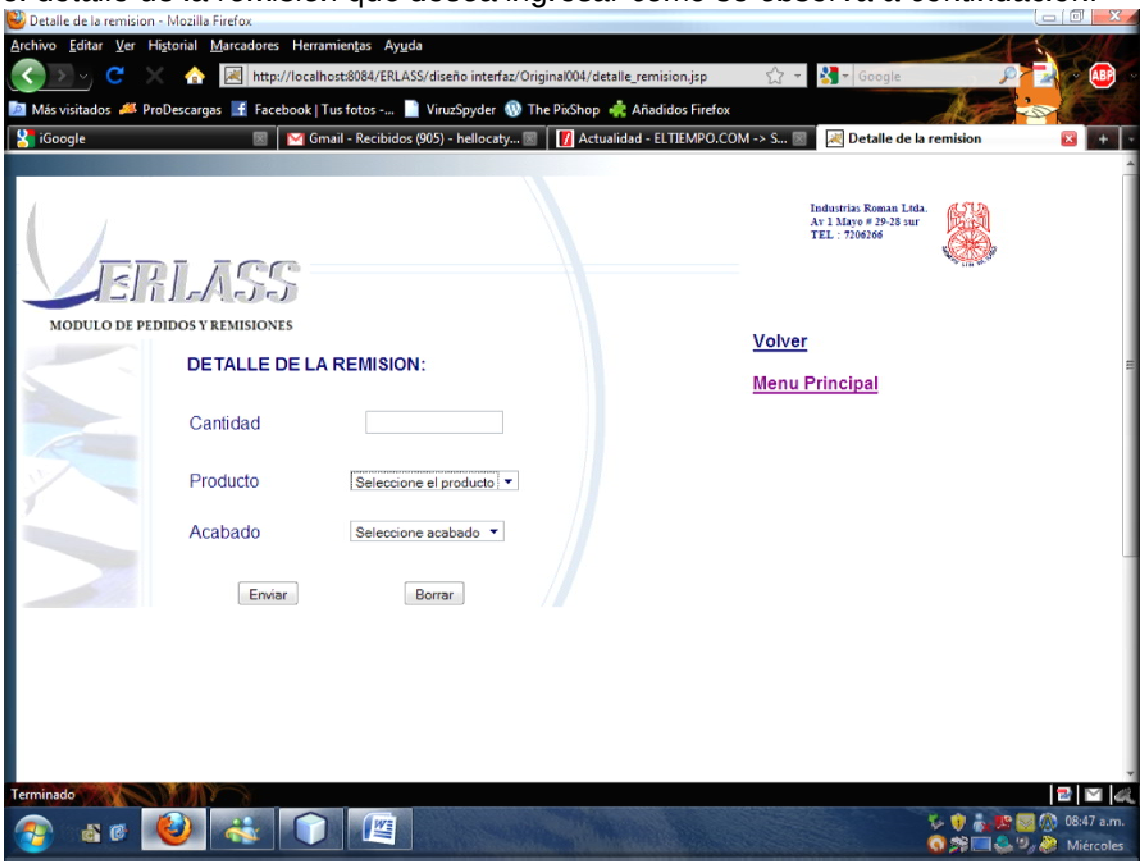
- Nueva Remisión:

En esta parte del sistema el administrador puede ingresar el detalle de la remisión:

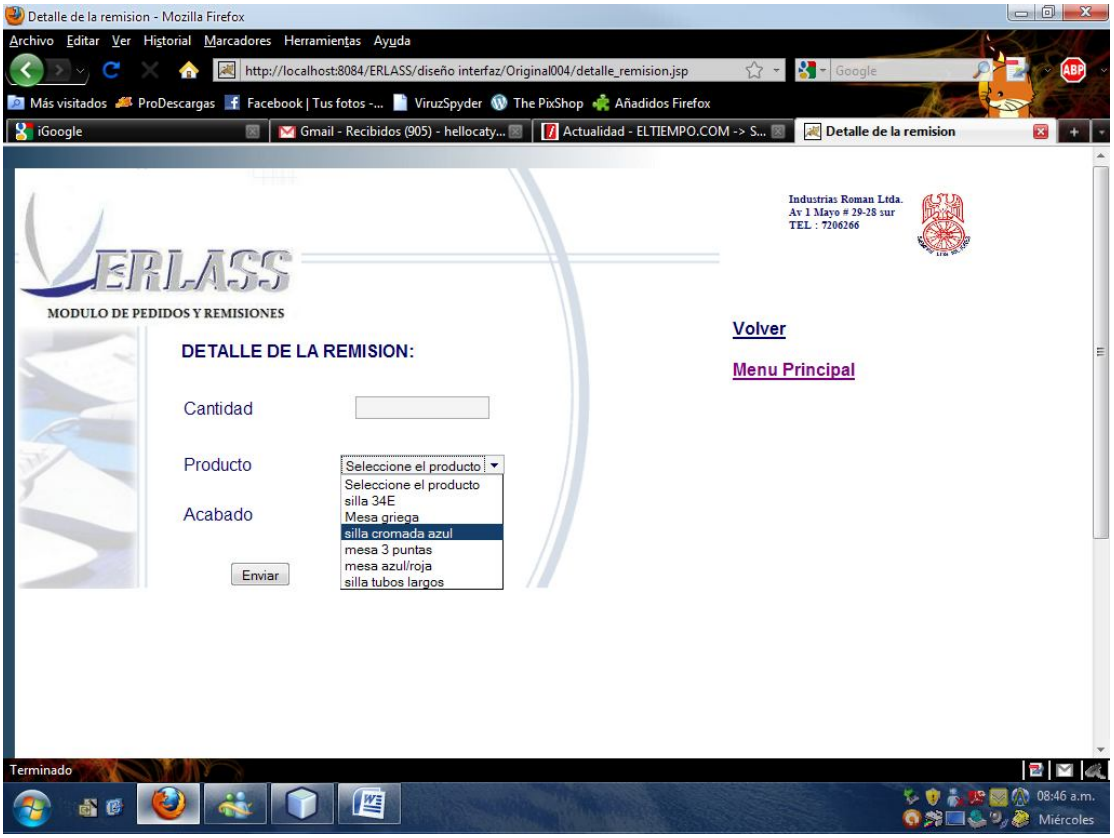


- Detalle Remisión:

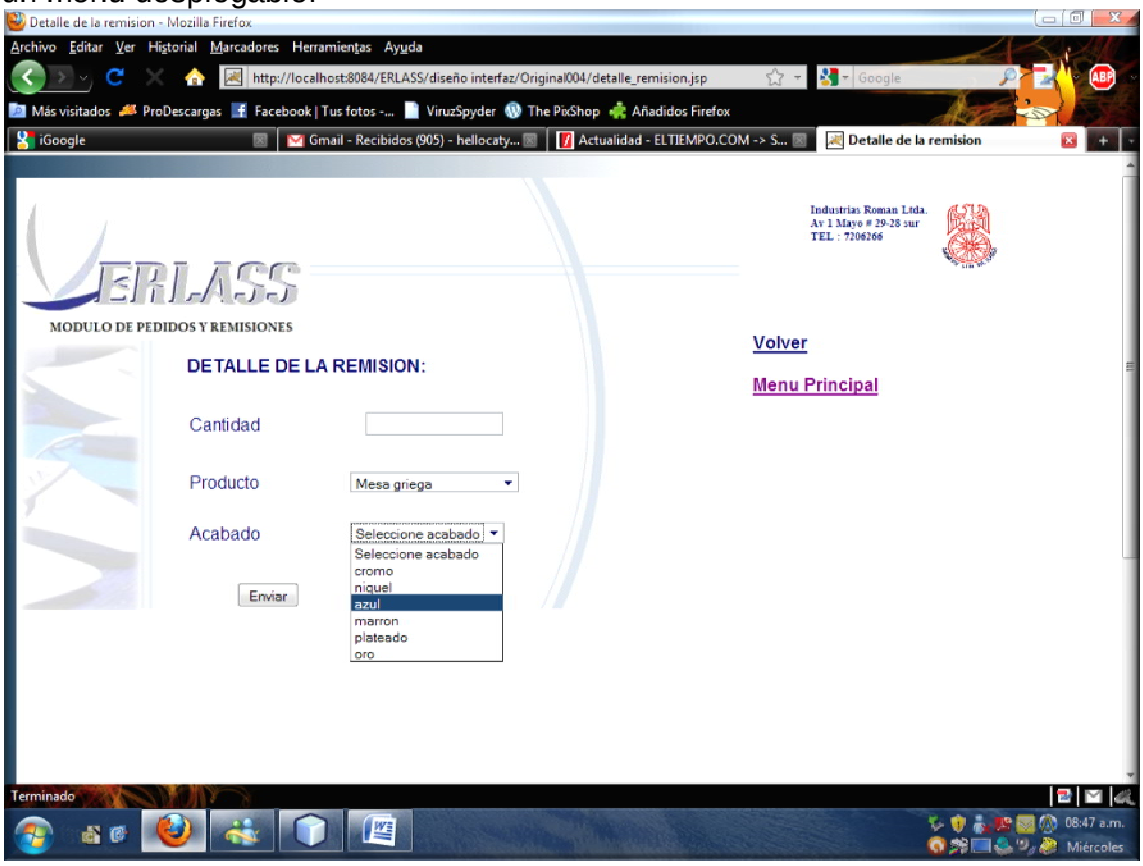
El administrador al elegir la opción insertar podrá generar la información necesaria para el detalle de la remisión que desea ingresar como se observa a continuación:



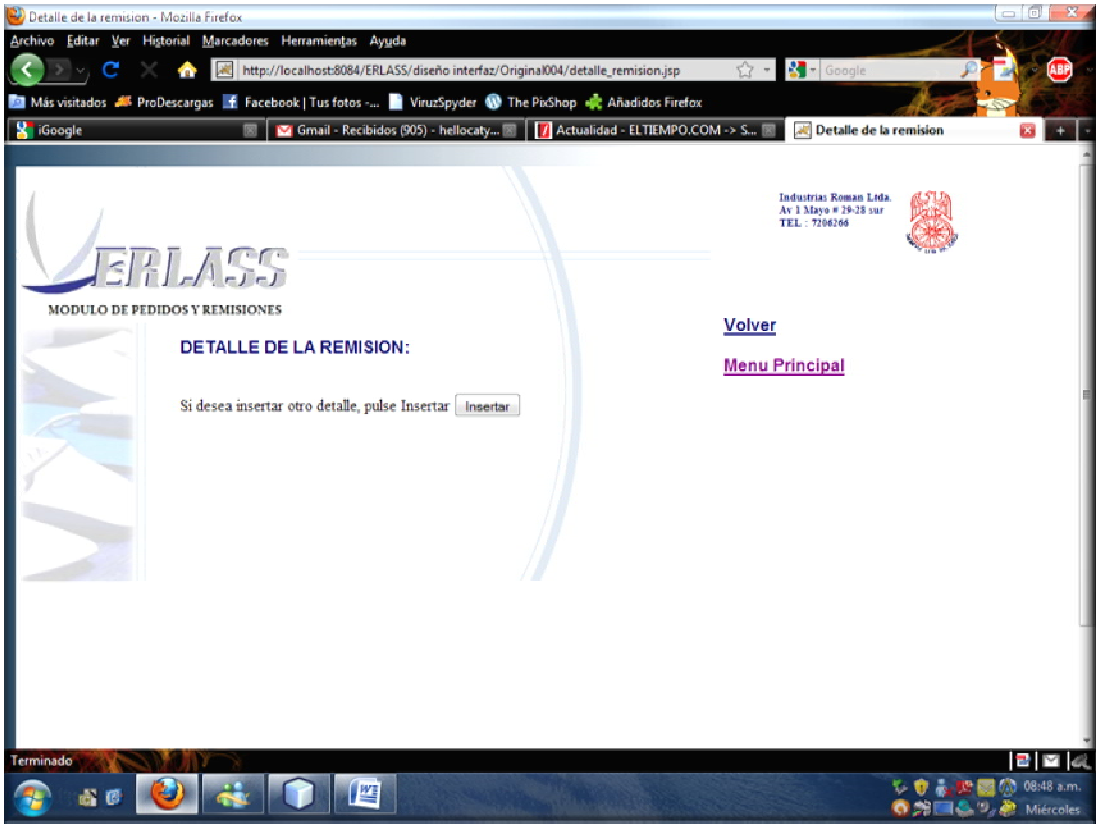
Después de haber ingresado la cantidad del detalle el administrador selecciona el producto que va a utilizar por medio de un menú desplegable:



De igual manera el administrador selecciona el acabado que va a utilizar por medio de un menú desplegable:

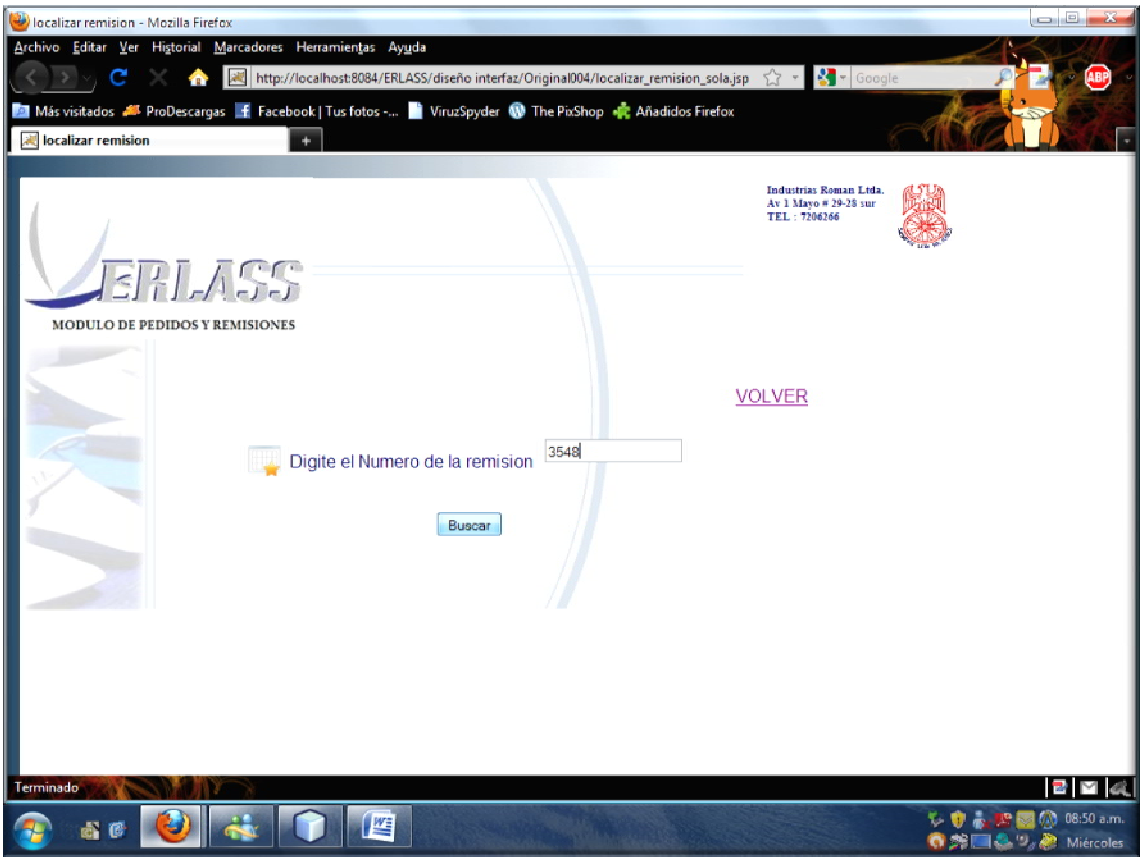


Si el administrador desea ingresar otro detalle de remisión elige la opción insertar nuevamente y realiza el proceso anterior:



- Consultar Remisión:

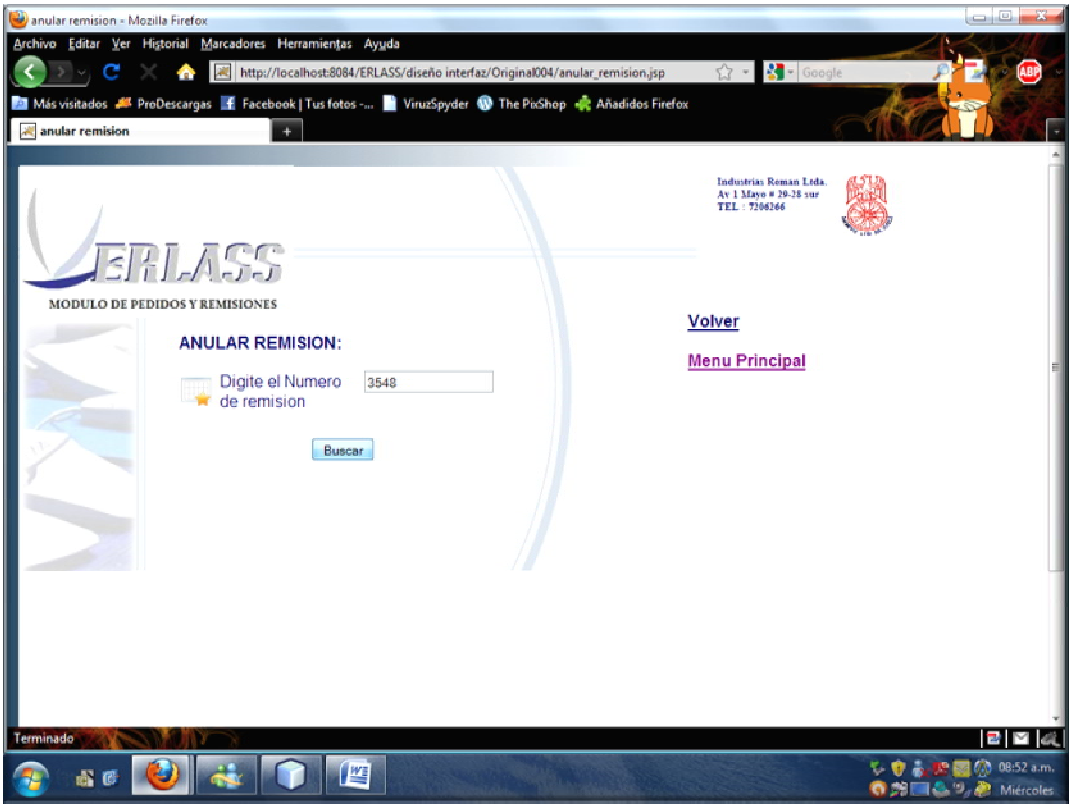
Cuando el administrador haya ingresado su nueva remisión, podrá consultar si realmente fue creada digitando el número de la remisión que desea consultar inmediatamente el sistema la buscara:



Después el sistema le mostrara al administrador un listado con las remisiones existentes en la base de datos y el podrá mirar si la creada se encuentra guardada:

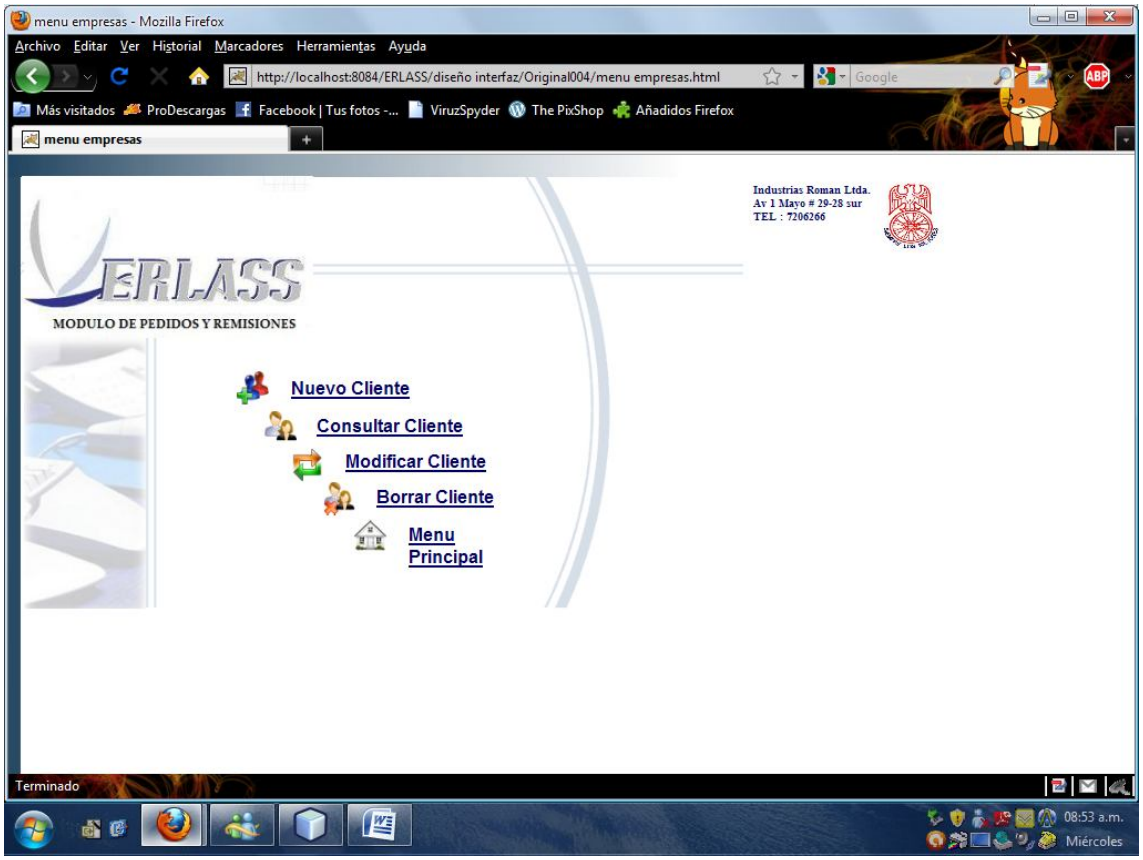
- Anular Remisión:

Después de que el administrador haya ingresado su nueva remisión la haya consultado tendrá la opción de poder anularla en caso de que la información haya sido errónea, en este caso el administrador digita el número del pedido que quiere anular lo busca y así se anulara satisfactoriamente en la base de datos:



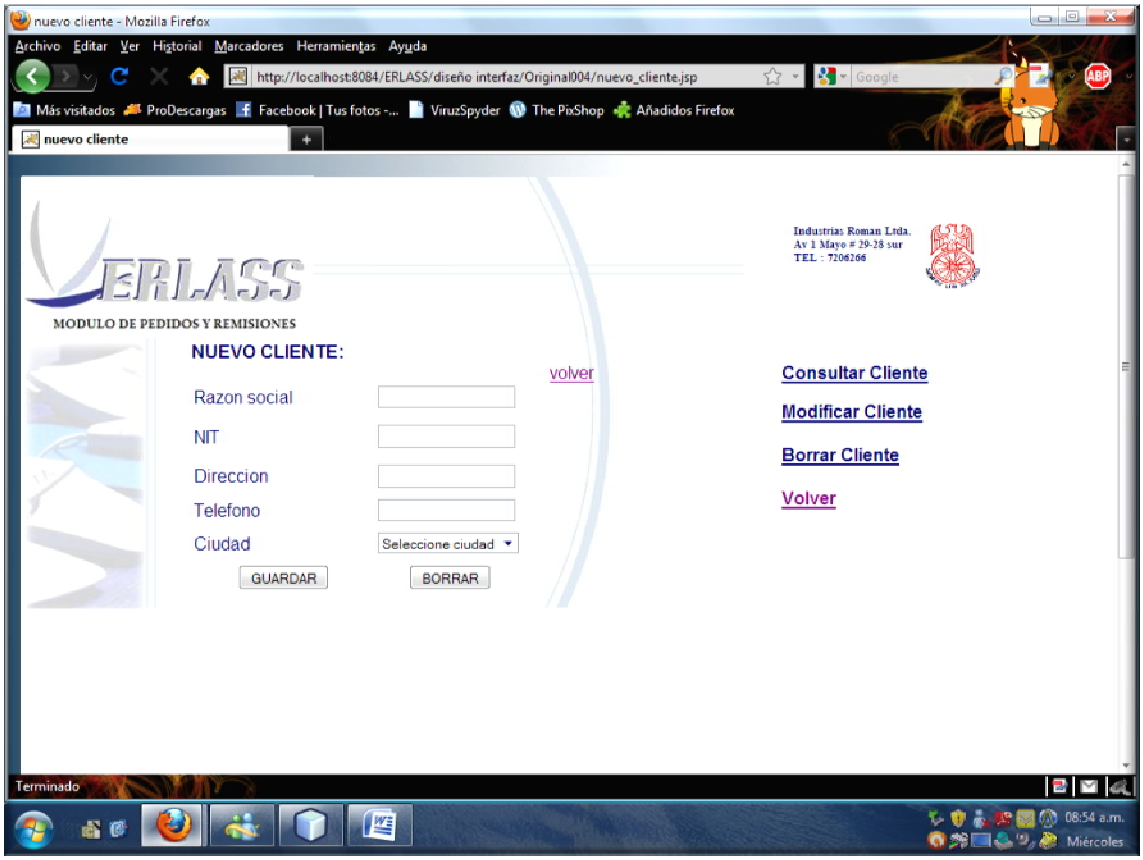
- Menú Clientes:

En esta parte del sistema el administrador puede observar el menú de los clientes y puede elegir el sub-módulo al que quiere ingresar.

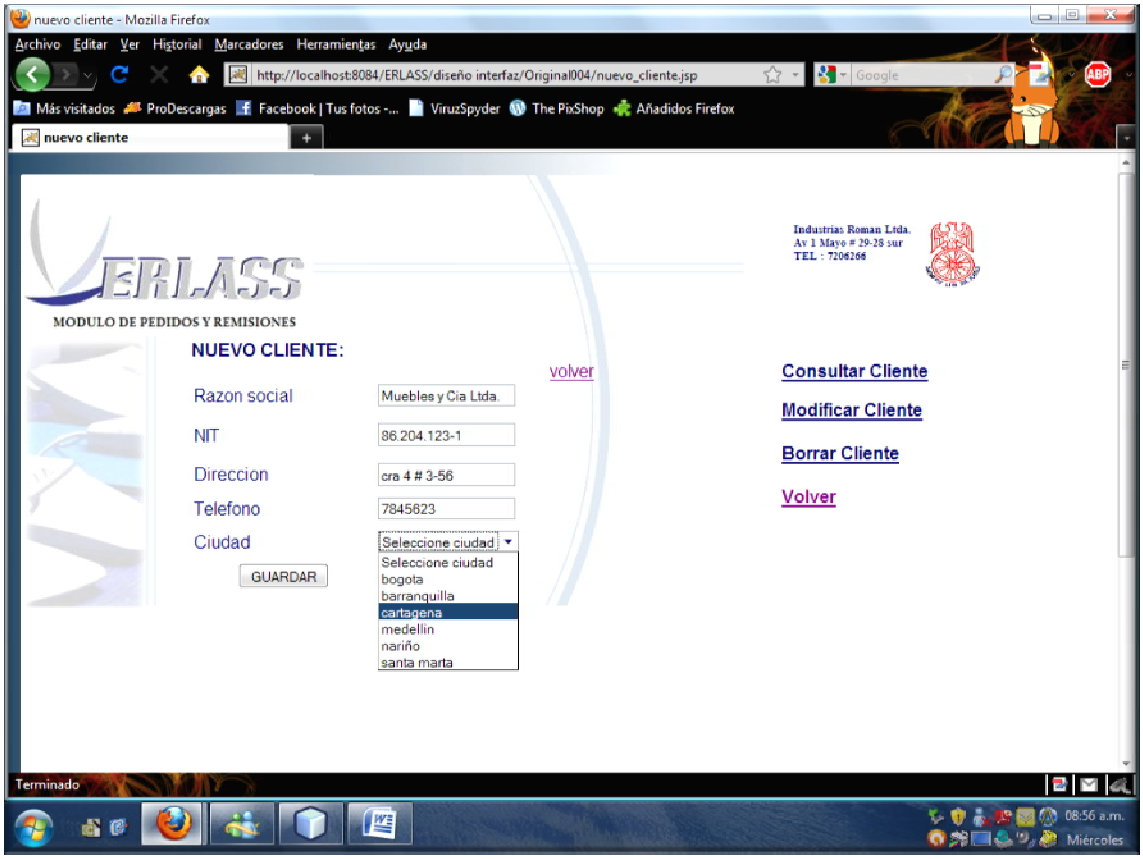


- Nuevo Cliente:

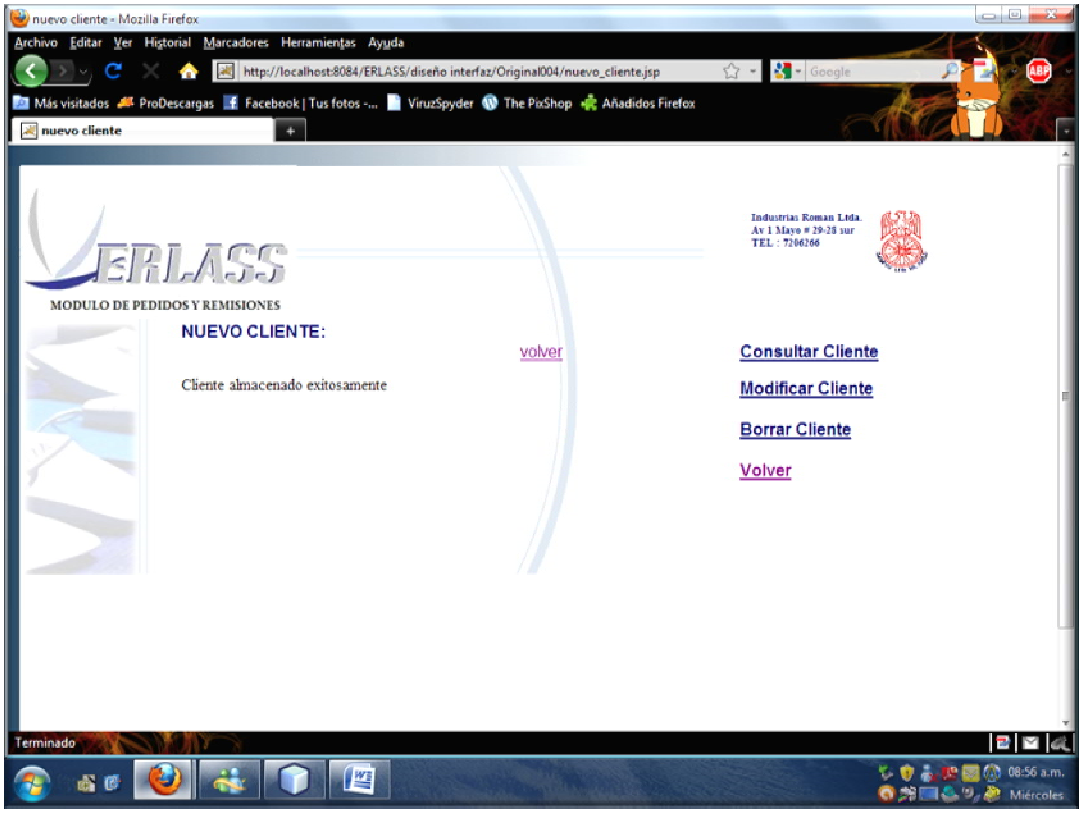
Después de que el administrador seleccione la opción nuevo cliente, el sistema muestra una pantalla donde se pude digitar la información necesaria:

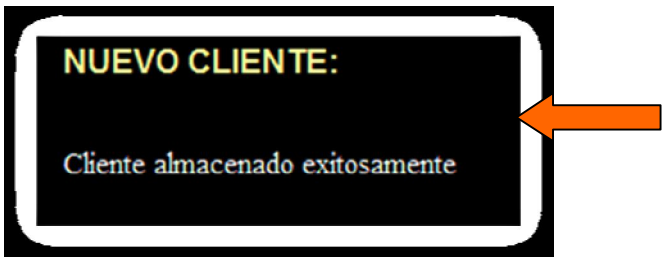


El administrador podrá elegir la ciudad que desea ingresar para el nuevo cliente por medio de un menú desplegable como se muestra a continuación:



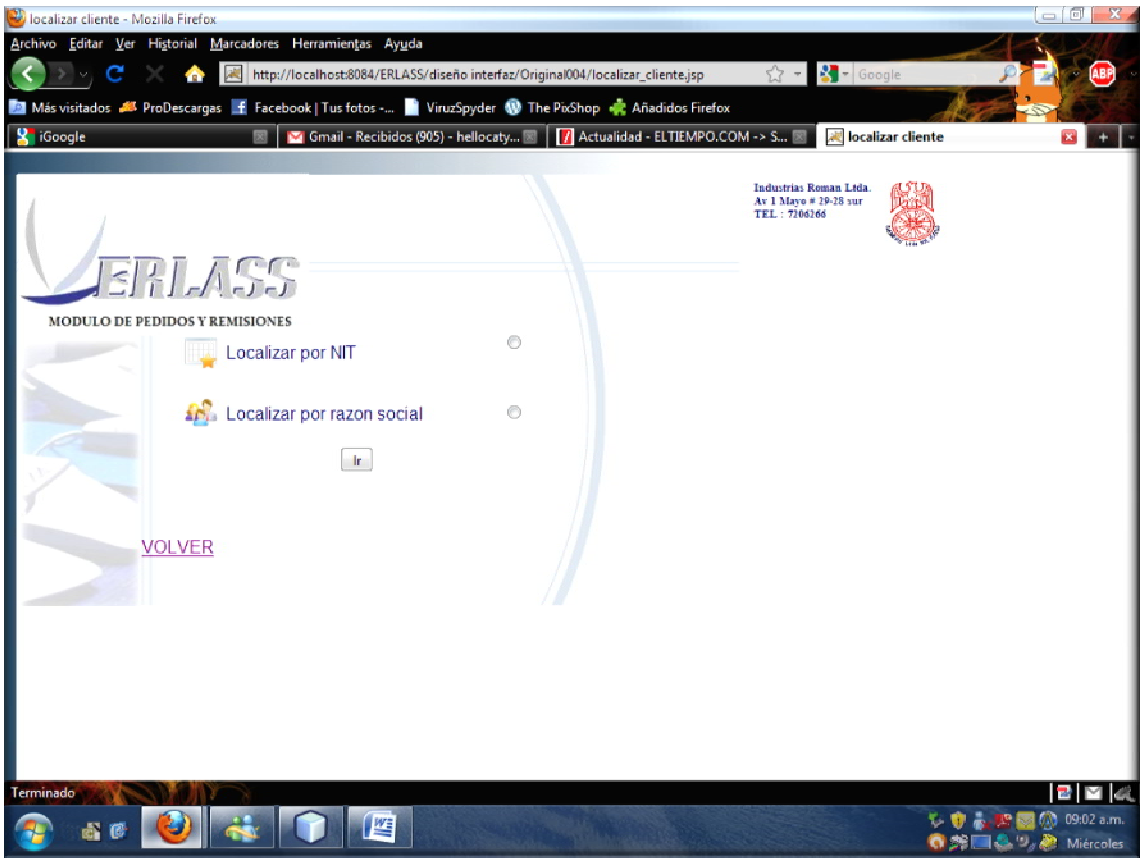
Después de que el administrador inserto su nuevo cliente el sistema le mostrara un mensaje donde le especificara que el cliente fue almacenado exitosamente en la base de datos:



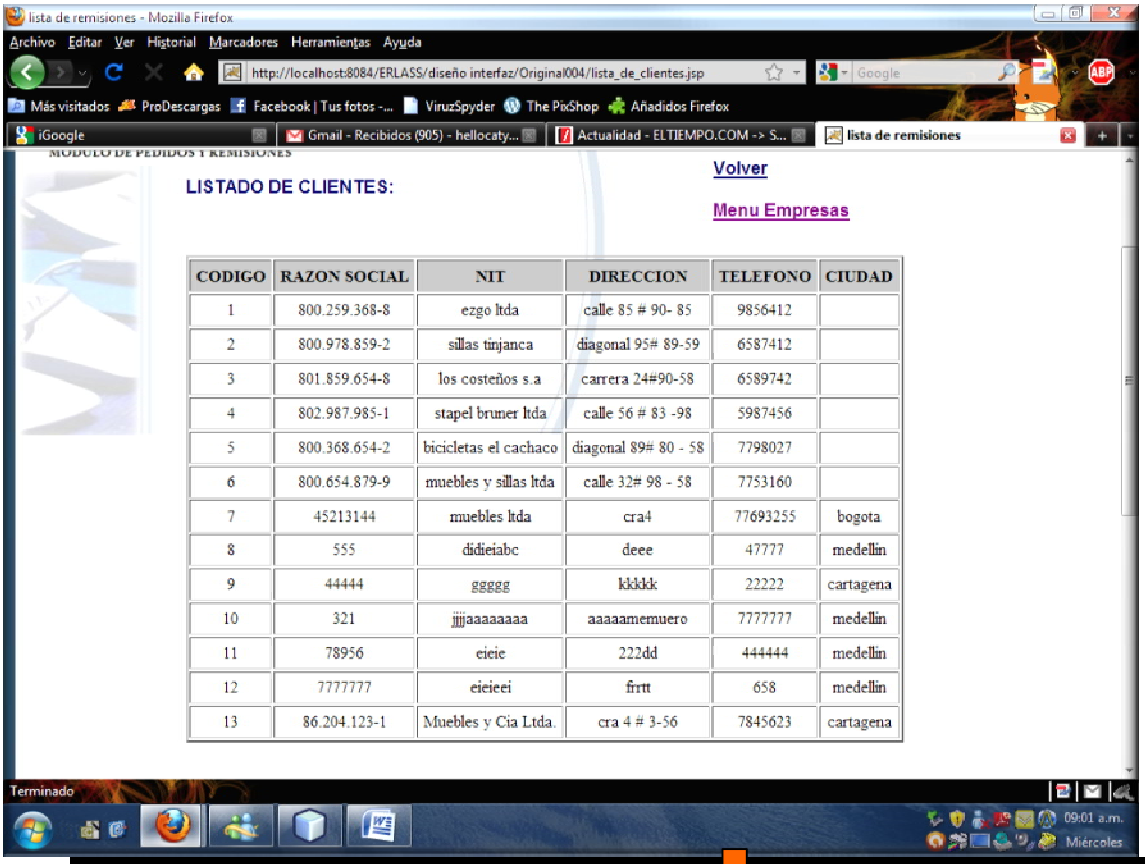


- Consultar Cliente:

El administrador podrá consultar los clientes por medio de dos opciones localizándolos por NIT o por la razón social:



El sistema le mostrara al administrador un listado con los clientes existentes en la base de datos y el podrá mirar si el que creo se encuentra guardado:



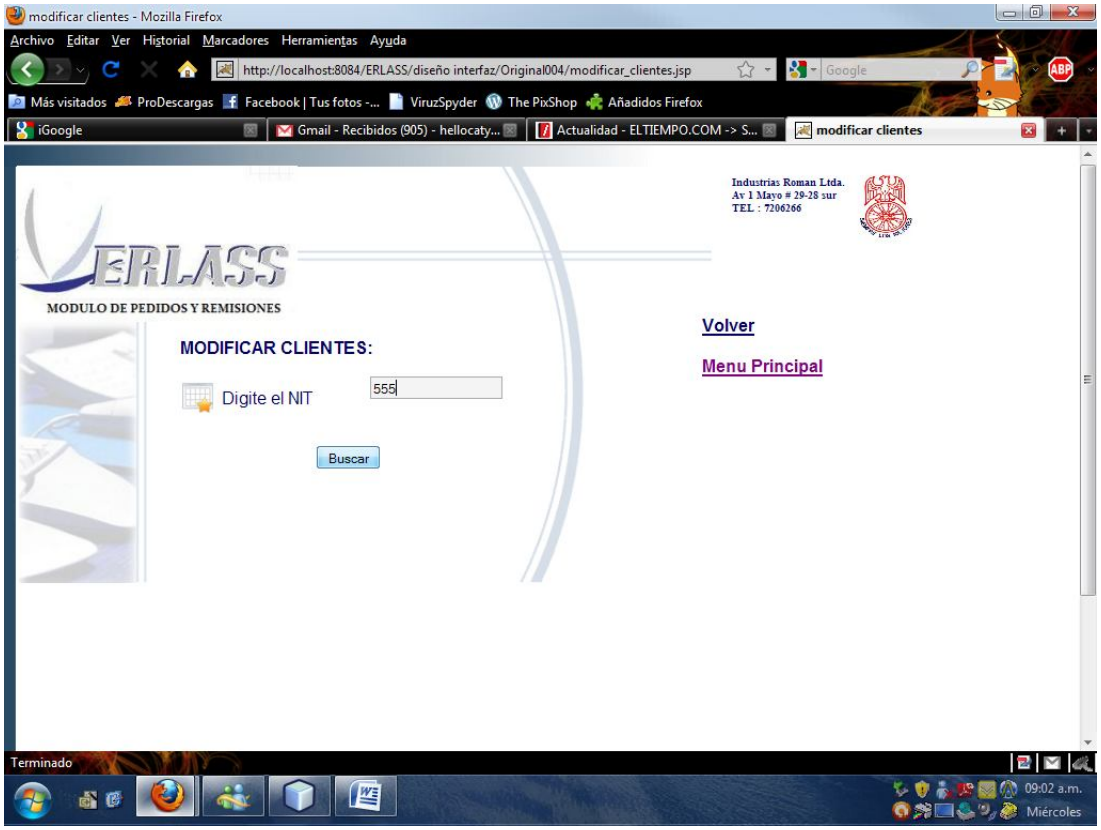
CODIGO	RAZON SOCIAL	NIT	DIRECCION	TELEFONO	CIUDAD
1	800.259.368-8	ezgo ltda	calle 85 # 90- 85	9856412	
2	800.978.859-2	sillas tinjanca	diagonal 95# 89-59	6587412	
3	801.859.654-8	los costeños s.a	carrera 24#90-58	6589742	
4	802.987.985-1	stapel bruner ltda	calle 56 # 83 -98	5987456	
5	800.368.654-2	bicicletas el cachaco	diagonal 89# 80 - 58	7798027	
6	800.654.879-9	muebles y sillas ltda	calle 32# 98 - 58	7753160	
7	45213144	muebles ltda	cra4	77693255	bogota
8	555	didieiabc	deee	47777	medellin
9	44444	ggggg	kkkkk	22222	cartagena
10	321	jjjaaaaaaaa	aaaaamemuero	7777777	medellin
11	78956	eieie	222dd	444444	medellin
12	7777777	eieieei	frrtt	658	medellin
13	86.204.123-1	Muebles y Cia Ltda.	cra 4 # 3-56	7845623	cartagena

LISTADO DE CLIENTES:

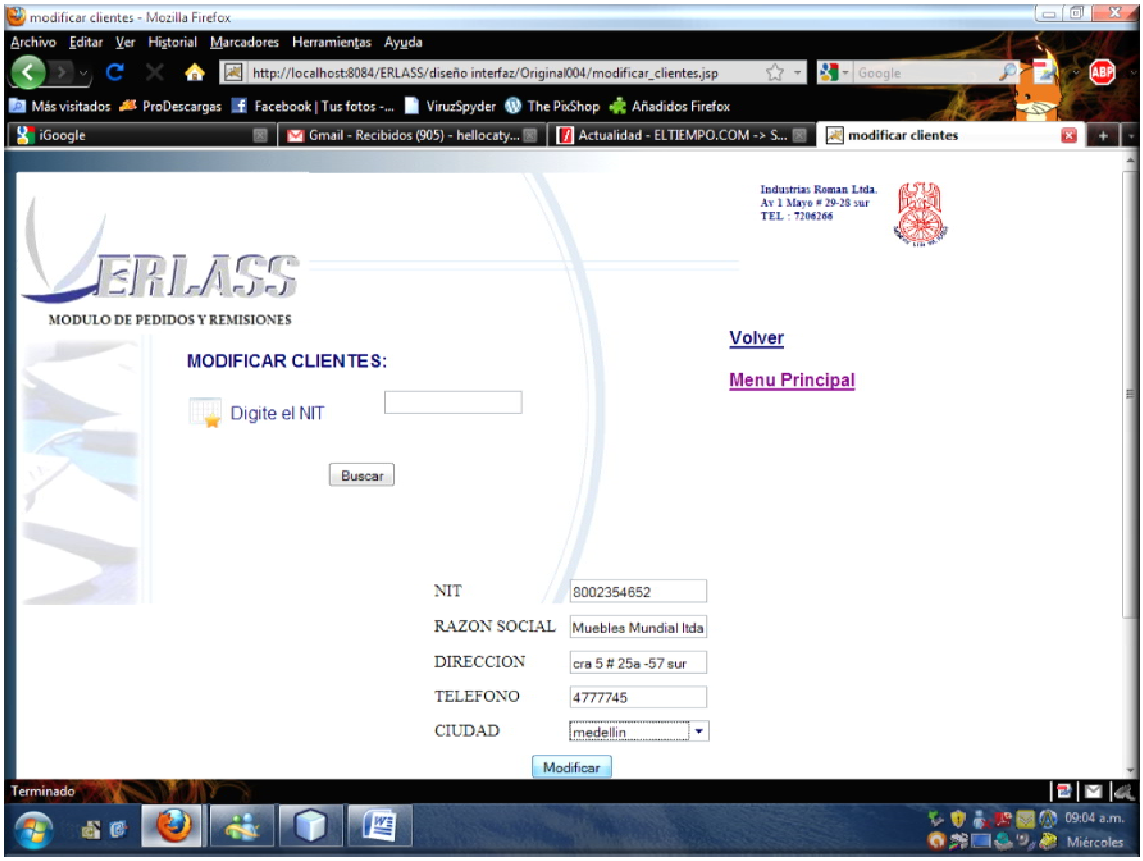
CODIGO	RAZON SOCIAL	NIT	DIRECCION	TELEFONO	CIUDAD
1	800.259.368-8	ezgo ltda	calle 85 # 90- 85	9856412	
2	800.978.859-2	sillas tinjanca	diagonal 95# 89-59	6587412	
3	801.859.654-8	los costeños s.a	carrera 24#90-58	6589742	
4	802.987.985-1	stapel bruner ltda	calle 56 # 83 -98	5987456	
5	800.368.654-2	bicicletas el cachaco	diagonal 89# 80 - 58	7798027	
6	800.654.879-9	muebles y sillas ltda	calle 32# 98 - 58	7753160	
7	45213144	muebles ltda	cra4	77693255	bogota
8	555	didieiabc	deee	47777	medellin
9	44444	ggggg	kkkkk	22222	cartagena

- Modificar Cliente:

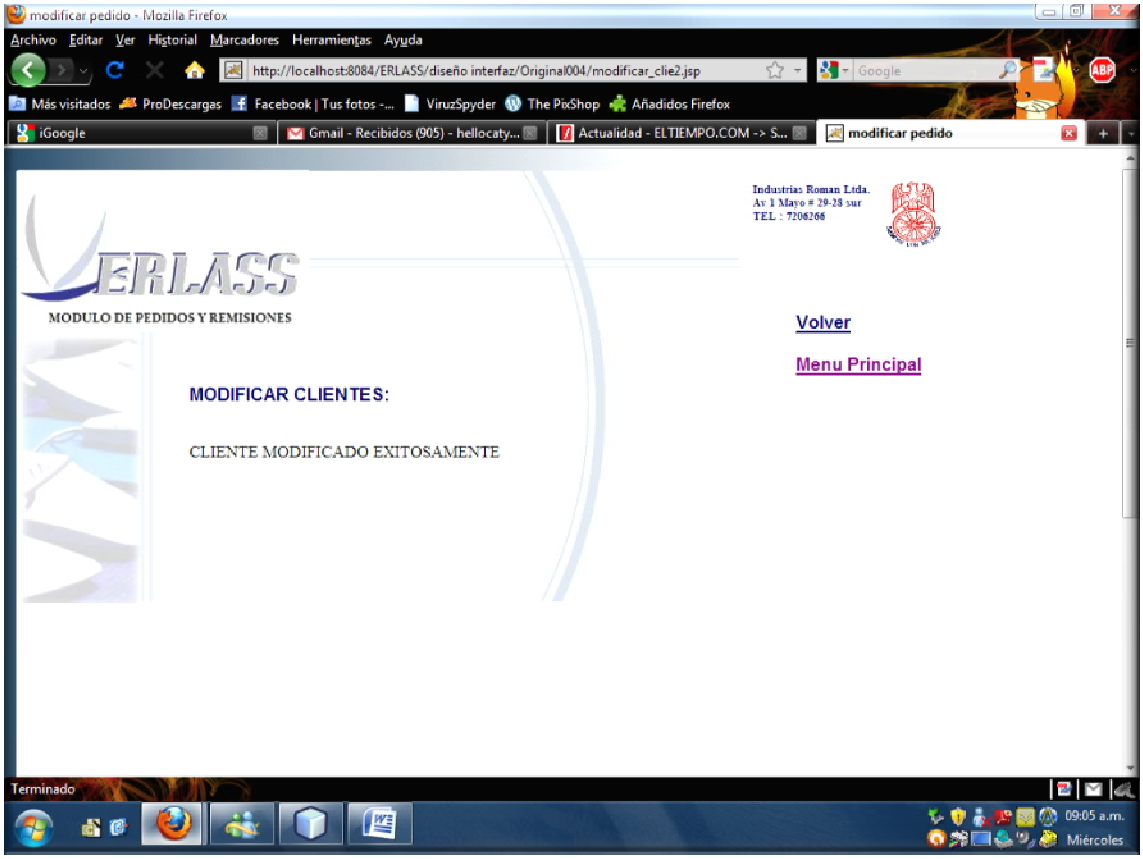
Después de que el administrador haya creado y consultado el cliente podrá modificar la información digitando el NIT:



Al digitar el NIT el sistema le mostrara al administrador la información que desea modificar para que el realice los cambios necesarios del cliente y luego le da la opción modificar:

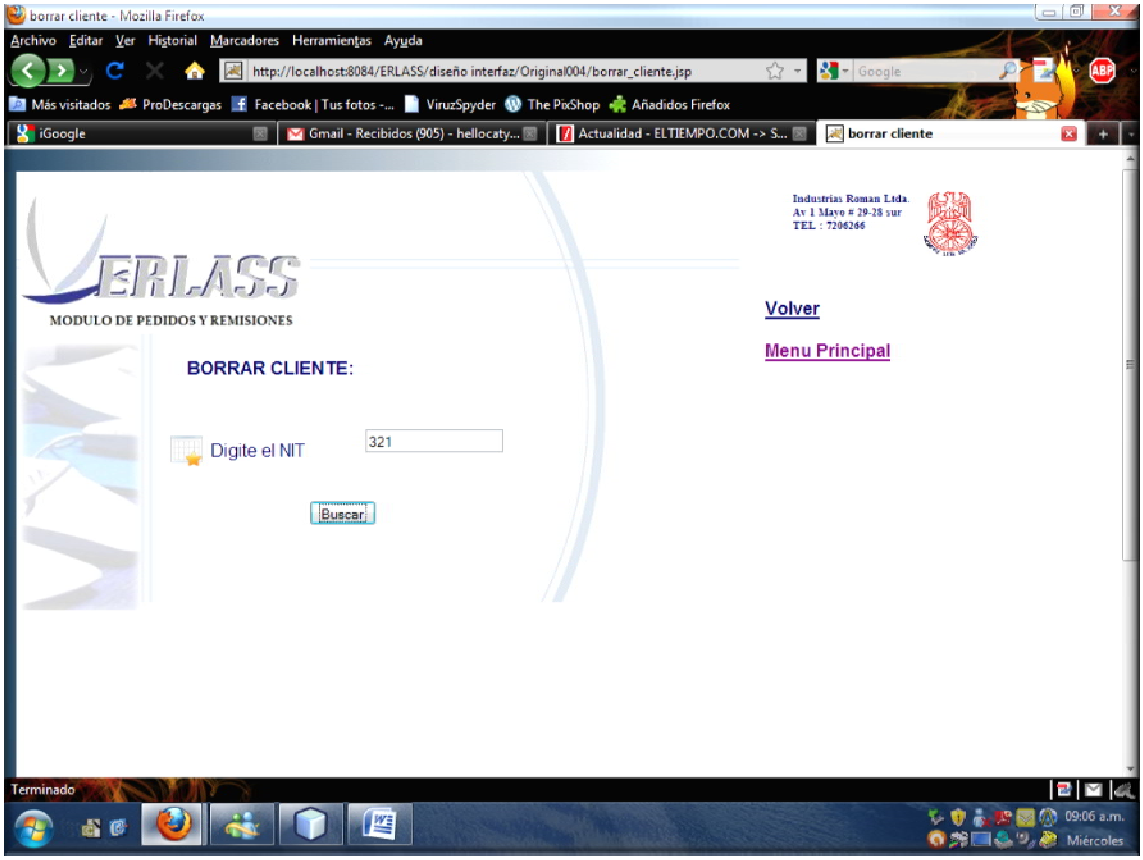


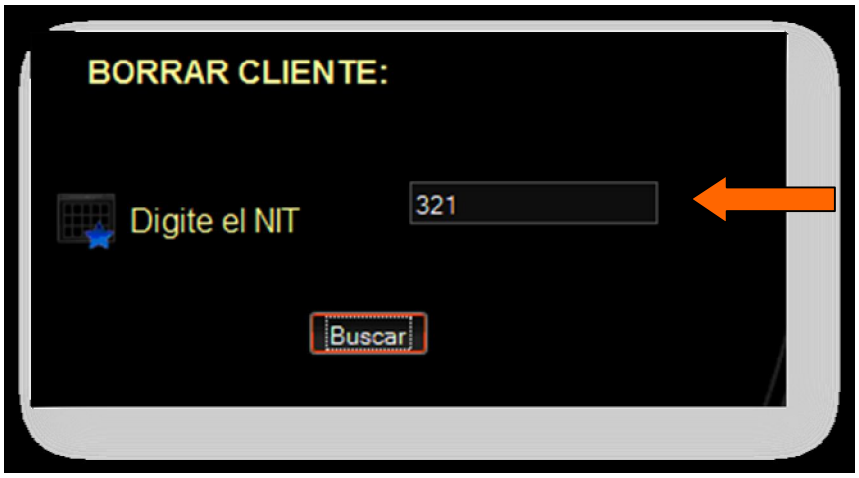
Después de que el administrador modifico su nuevo cliente el sistema le mostrara un mensaje donde le especificara que el cliente fue modificado exitosamente en la base de datos:



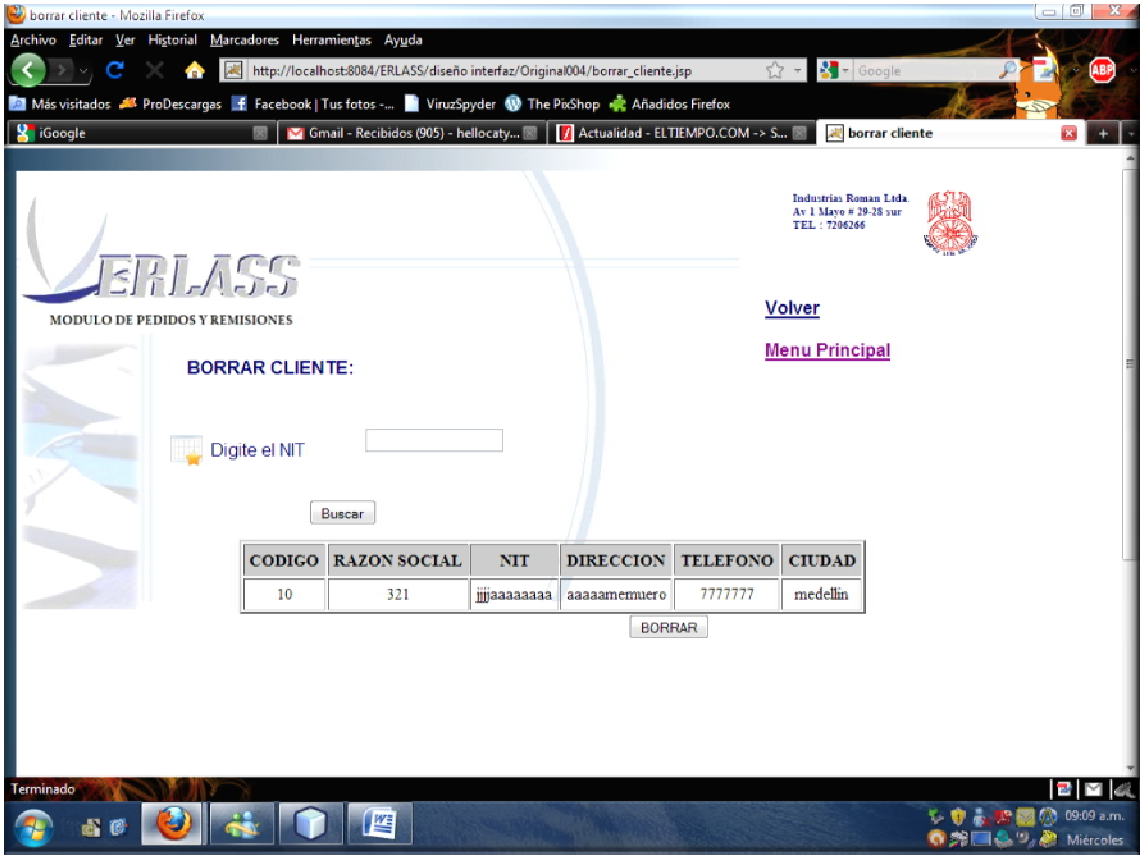
- **Borrar Cliente:**

El administrador ingresa el NIT del cliente que desea borrar del sistema este inmediatamente lo buscara:

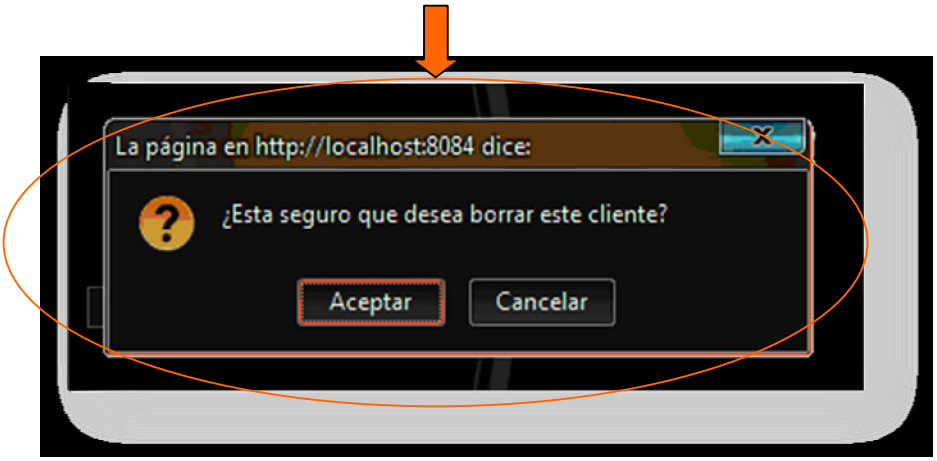




El sistema le mostrara al administrador la información del cliente que desea borra por medio de una tabla:

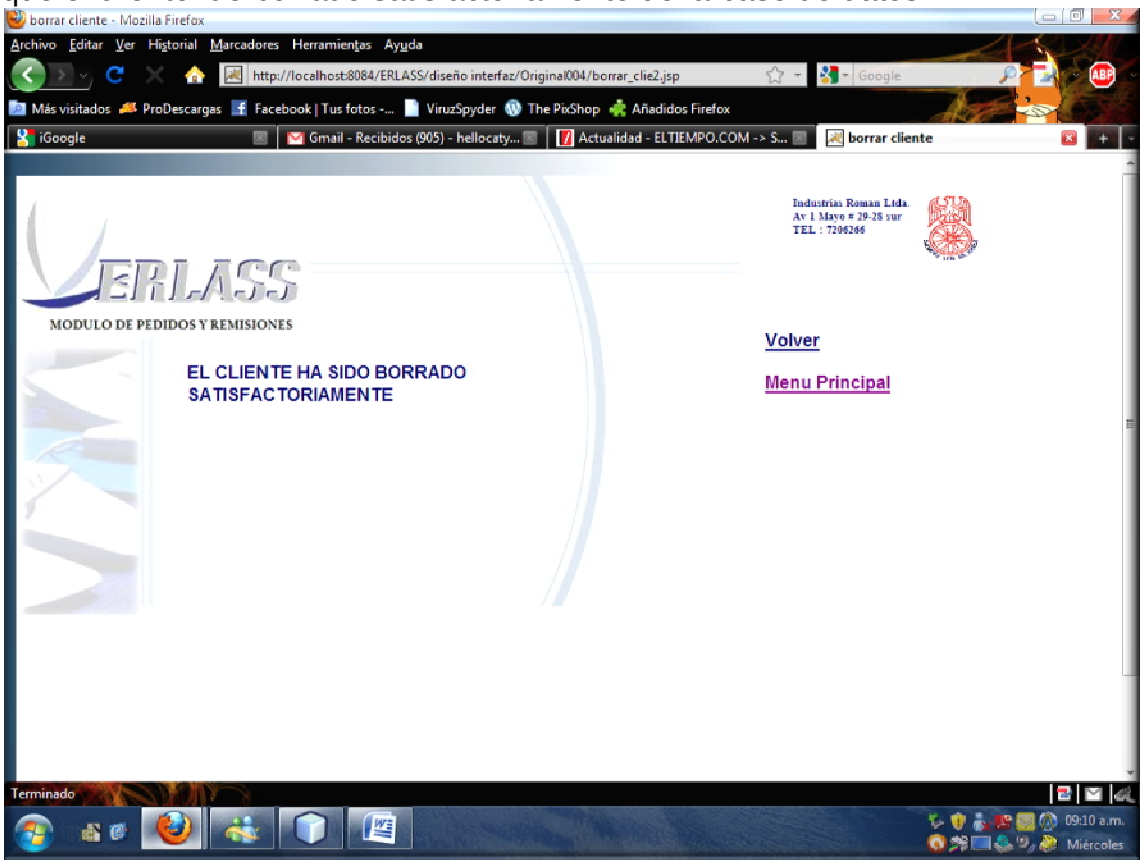


Después de que el administrador haya elegido la opción borrar el sistema le preguntara si realmente esta seguro de borrar el cliente como observamos a continuación:



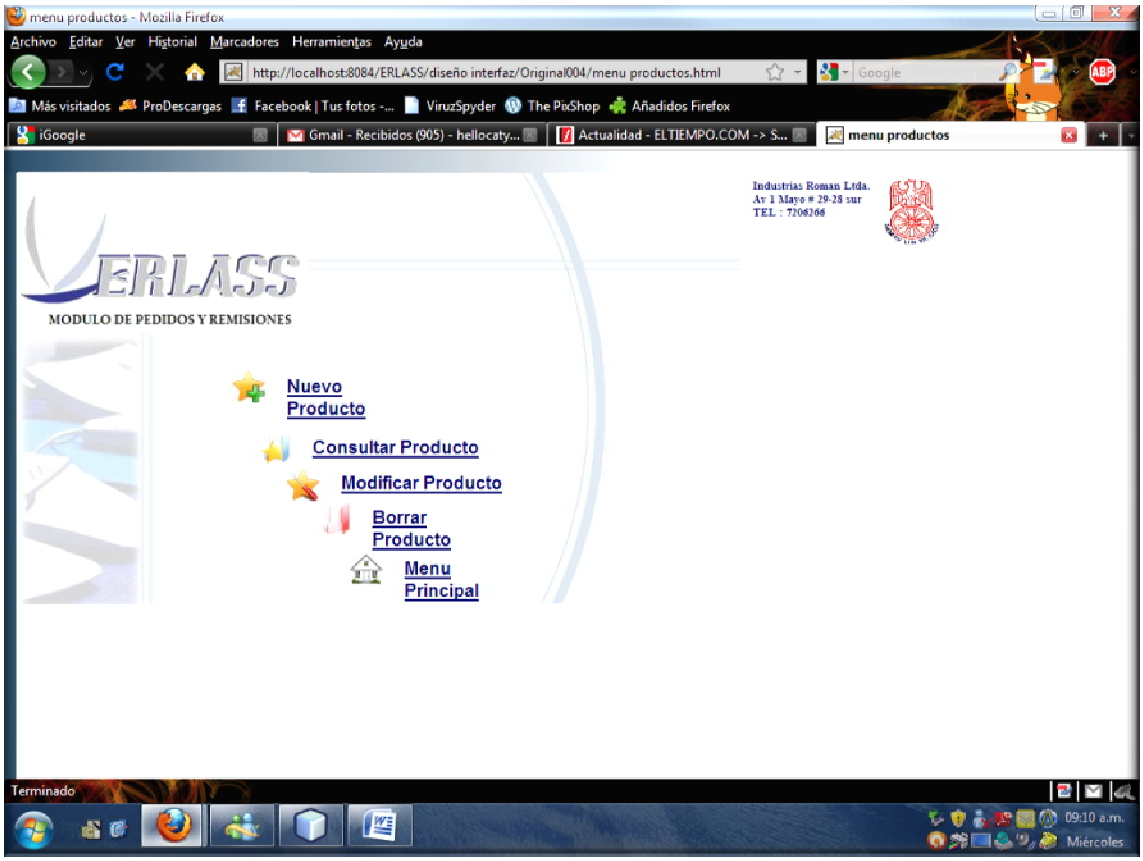
Borrar Cliente:

Si la opción escogida por el administrador el sistema le mostrara un mensaje que le dirá que el cliente fue borrado satisfactoriamente de la base de datos:



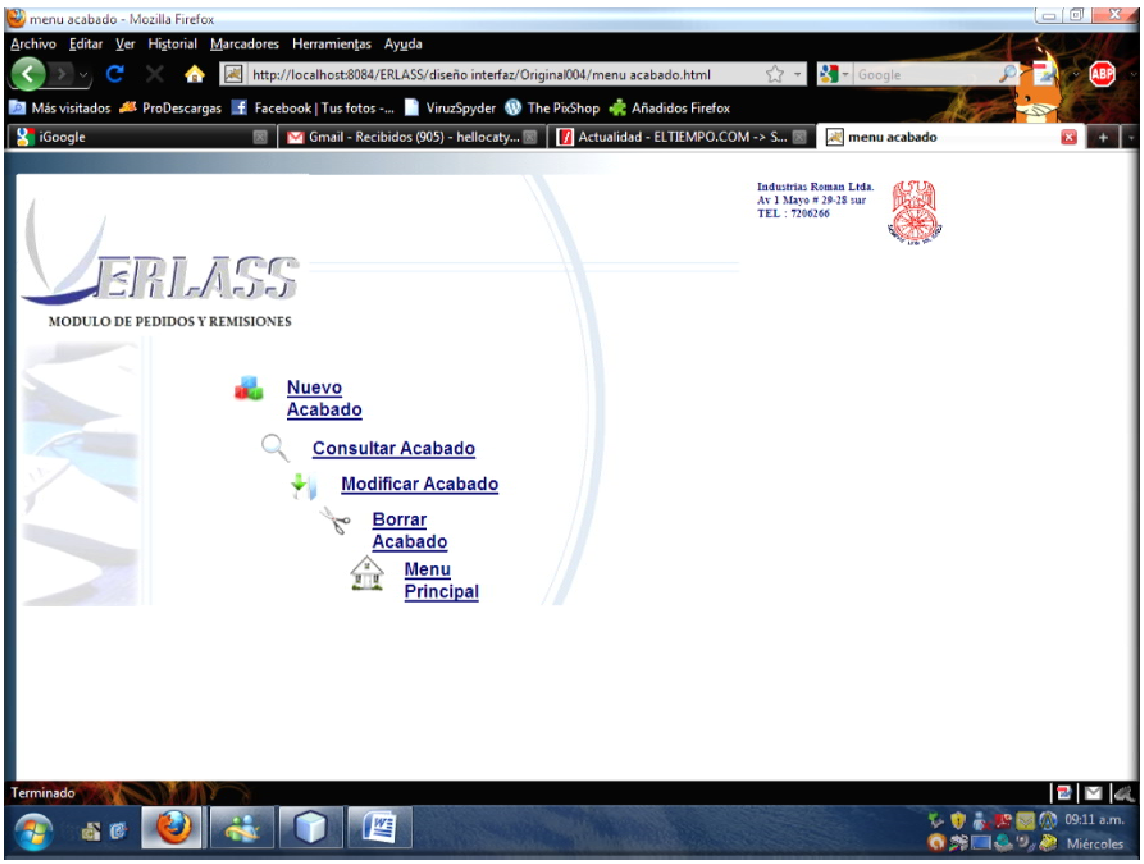
- Menú Productos:

En esta parte del sistema el administrador puede observar el menú de los productos y puede elegir el sub-módulo al que quiere ingresar.



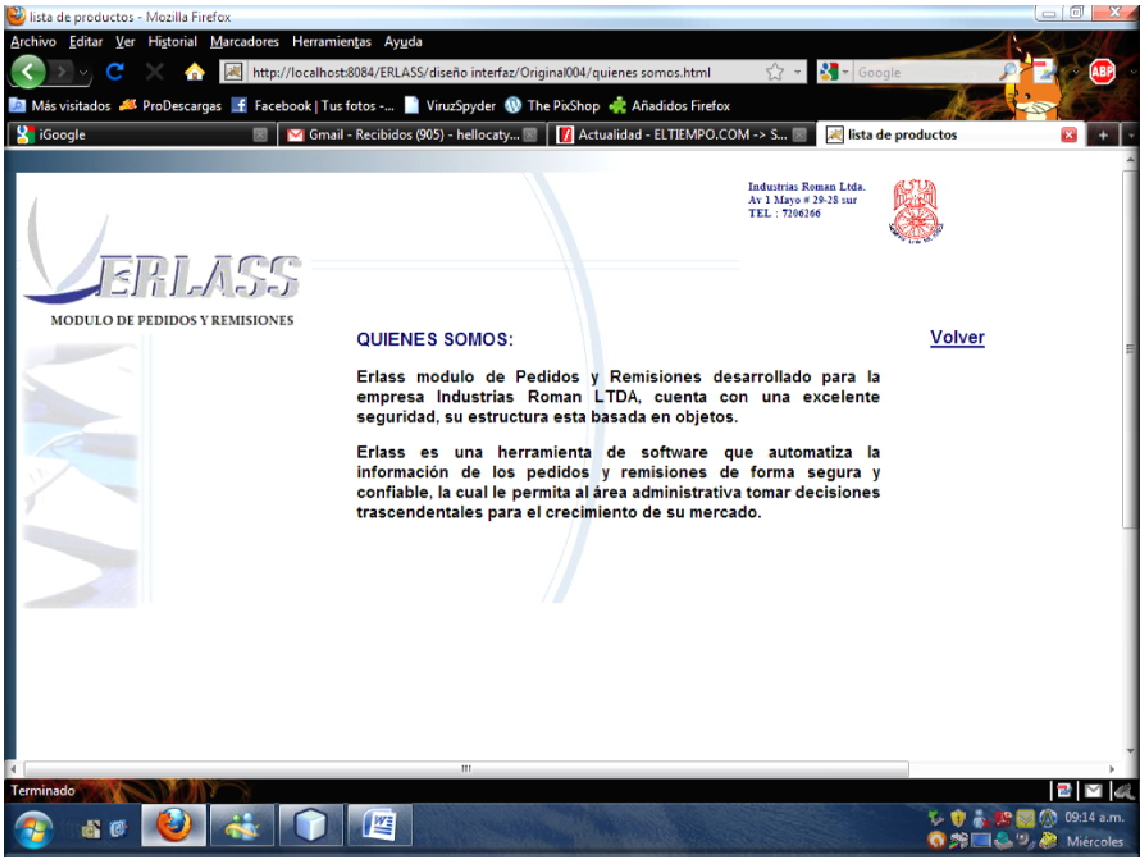
- Menú Acabados:

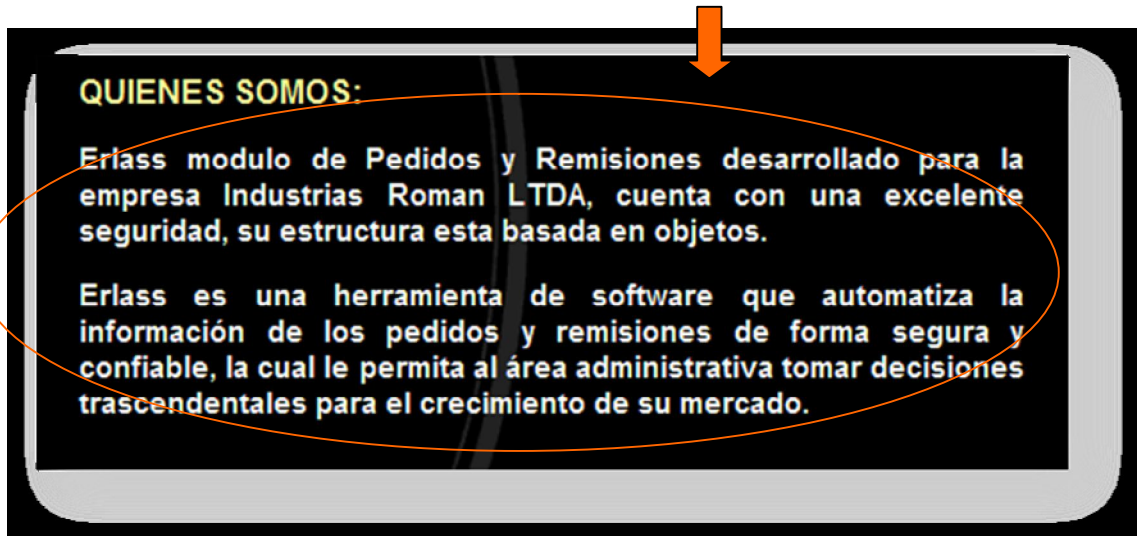
En esta parte del sistema el administrador puede observar el menú de los acabados y puede elegir el sub-módulo al que quiere ingresar



- Quienes Somos:

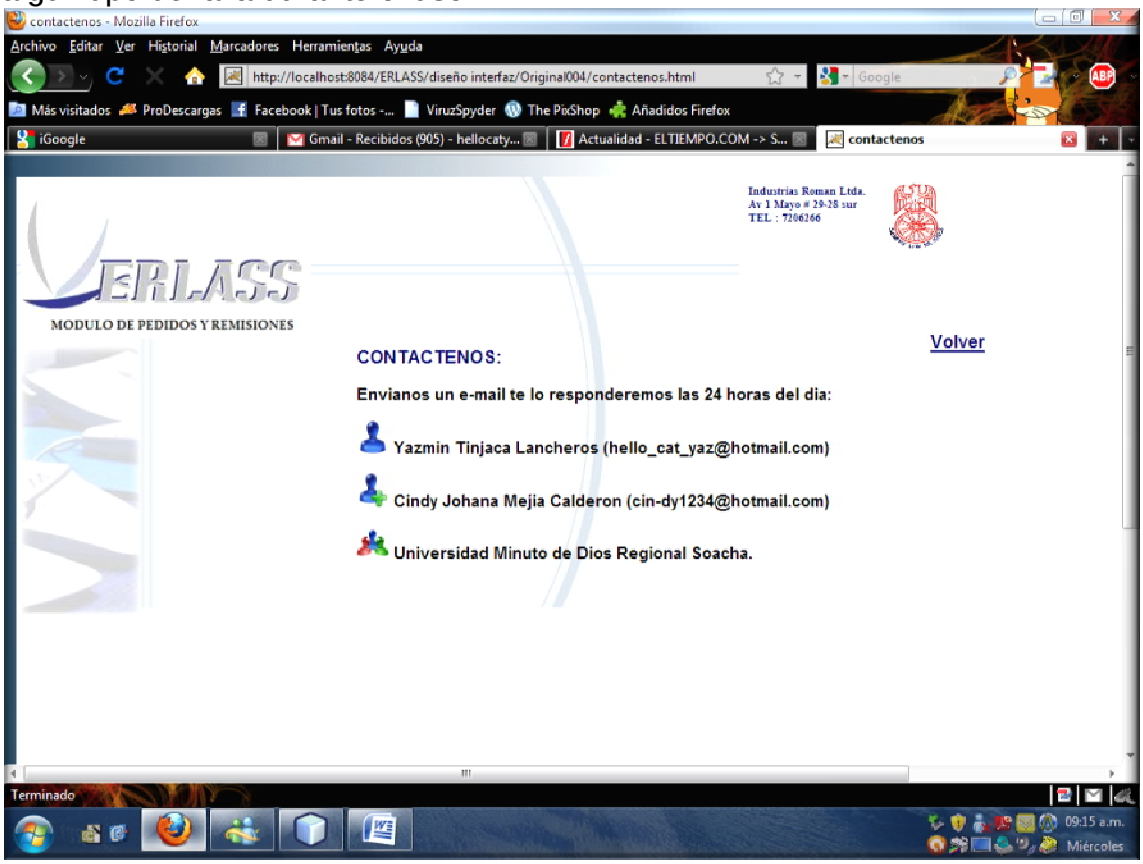
El administrador tendrá la opción de saber acerca de ERLASS para que sirve y el fin con el cual se creo como se muestra a continuación:

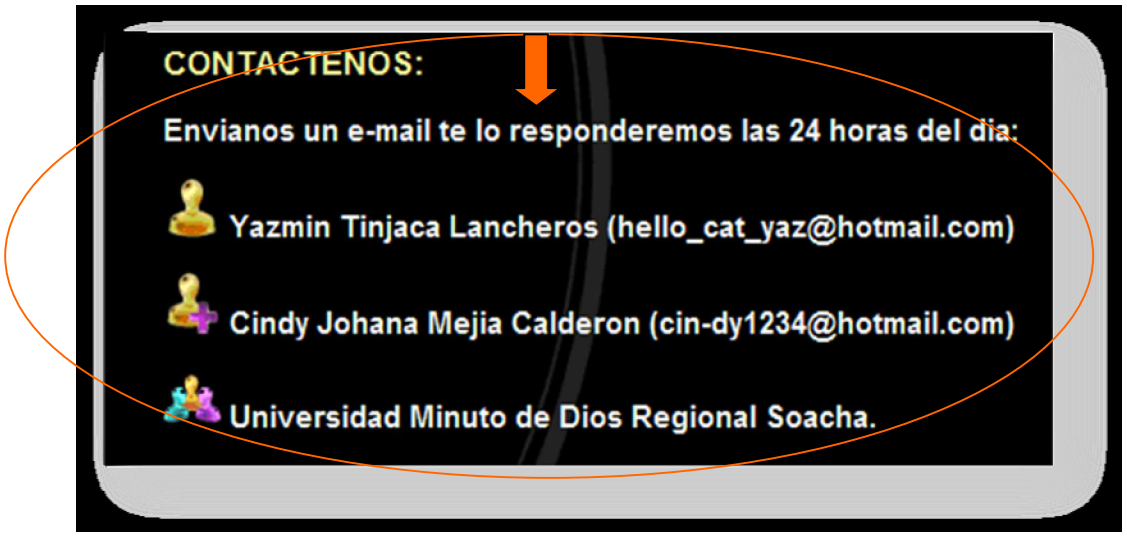




Contáctenos:

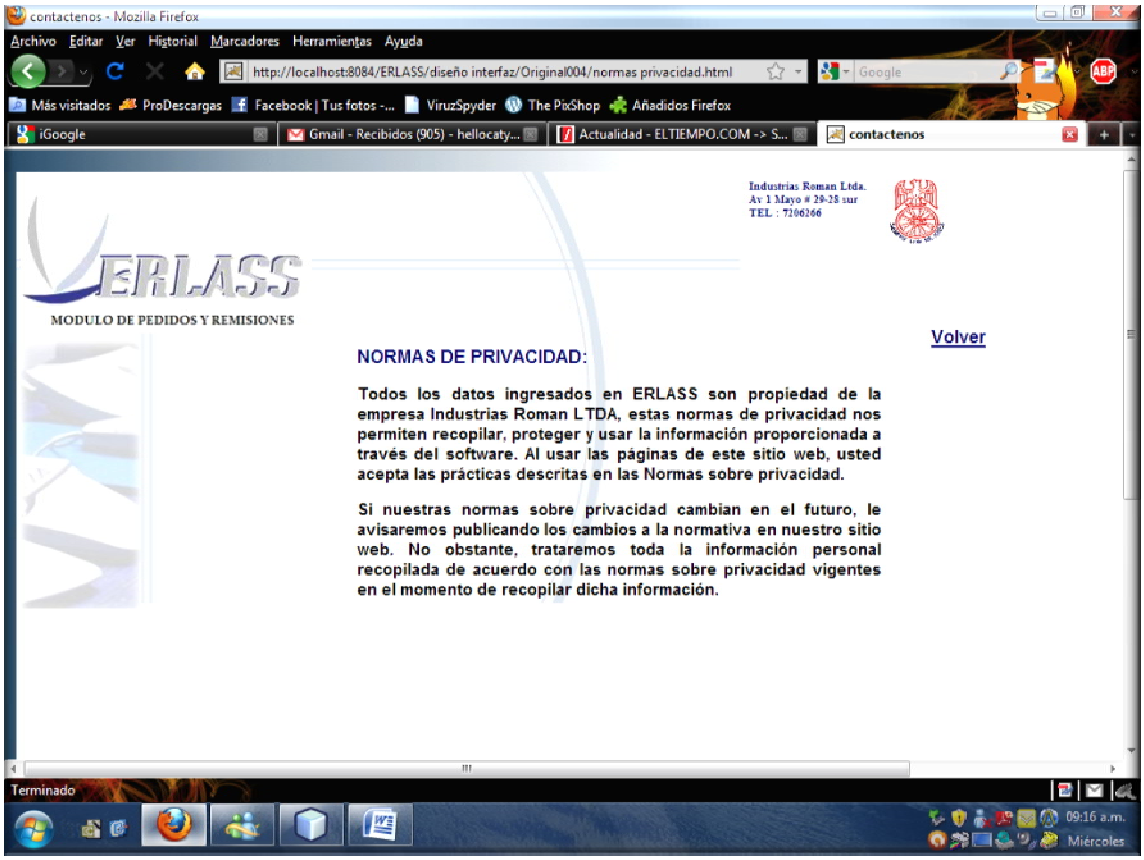
De igual forma el administrador podrá contactarnos en caso de que el software presente algún tipo de falla durante el uso:

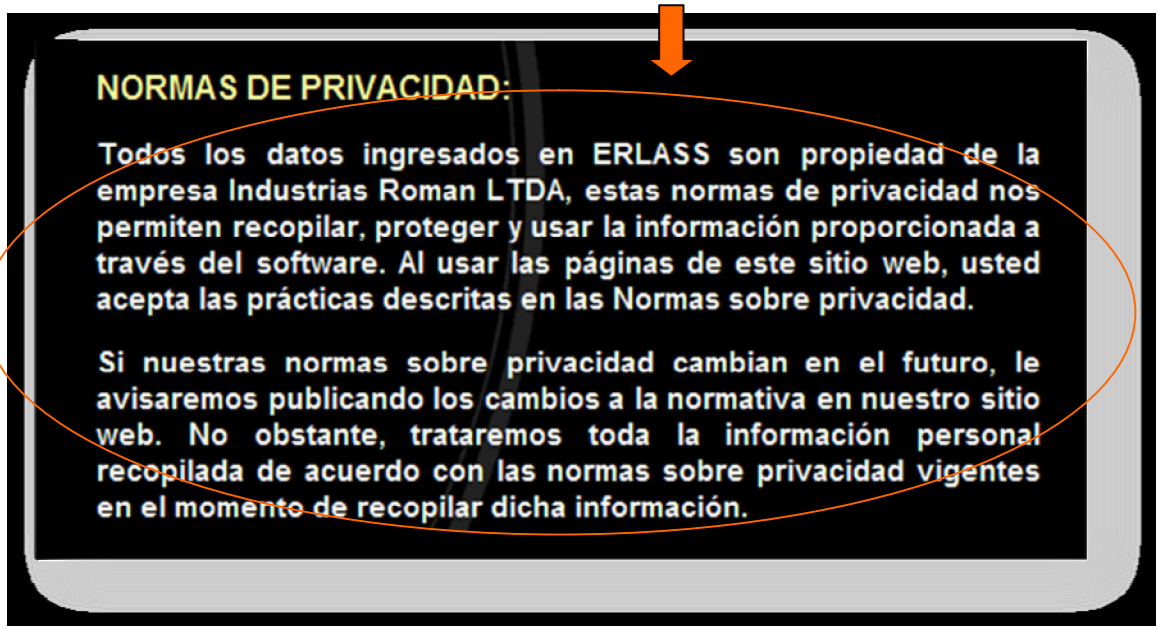




Normas de Privacidad:

El administrador podrá darse cuenta que ERLASS es seguro confiable y que los datos ingresados se encuentran en privacidad porque cuenta con un sistema de seguridad como se muestra a continuación:





11.3 INTERFAZ DE PROCESAMIENTO DE DATOS

PAQUETE CONECTIVIDAD

- **Clase Conexión:**

```
package Conectividad;

import java.sql.*;

//Declaracion de la clase conexion
public class Conexion {
    protected Connection canal; //Instanciar un objeto como canal de conexión
    protected Statement instruccion; //Instanciar un objeto como flujo de conexion

    protected String strcon; //Variable que almacena el logueo al web server

    public Conexion()
    {
        canal = null; //Instanciar un objeto como canal de conexión
        instruccion = null; //Instanciar un objeto como flujo de conexion

        strcon = "";

    }//Cierre metodo constructor

    public void Conectar()
    {
        strcon = "jdbc:mysql://localhost/pedidos_remisiones?user=root&password=root";
```

```
try
{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
   System.out.println("\nDriver de conexion instanciado...\n");

        canal = DriverManager.getConnection(strcon);
   System.out.println("\nCanal de conexión establecido...\n");

        instruccion =
canal.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UP
DATABLE);
   System.out.println("\nFlujo de conexión disponible...\n");

}
  catch(java.lang.ClassNotFoundException e)
{
   System.out.println("\nNo se encontro el driver de conexion.\n");
}
  catch(SQLException e)
{
   System.out.println("\nNo fue posible establecer la conexion.\n");
}
catch(java.lang.InstantiationException e)
{
   System.out.println("\nNo fue posible instanciar la clase de conexión.\n");
}
catch(java.lang.IllegalAccessException e)
{
   System.out.println("\nSe produjo un acceso ilegal a la clase.\n");
}
```

```
    }//Cierre método Conectar

    public Statement getInstruccion()
    {
        return instruccion;

    }//Cierre método getInstruccion

    public Connection getCanal()
    {
        return canal;

    }//Cierre método getCanal

    public void liberarRecursos()
    {
        try
        {
            instruccion.close();
            canal.close();
            System.out.println("\nSe liberaron los recursos utilizados....(Canal y Flujo de
conexión).\n");
        }//Cierre del try
        catch(SQLException e)
        {
            System.out.println("\nNo fue posible cerrar el canal y el flujo de conexion...\n");
        }//Cierre del catch

    }//Cierre método liberarCursor
```

```
}//Cierre de la clase conexión
```

- **Clase Cursor**

```
package Conectividad;

import java.sql.*;

//Declaración de la clase cursor
public class Cursor {
   protected ResultSet tabla; //Instanciar un objeto como cursor

   public Cursor()
   {
      tabla = null; //Instanciar un objeto como cursor

   }//Cierre método constructor

   public void Ejecutarconsulta(Statement instruccion1, String Consulta1)
   {
      try
      {
         tabla = instruccion1.executeQuery(Consulta1);
         System.out.println("\nSe ejecutó con éxito la sentencia SQL....\n");

      }//Cierre del try

      //Si la sentencia NO se ejecutó con éxito, capturar la excepción...
      catch(SQLException e)
      {
```

```
            System.out.println("\nNo se ejecutó la sentencia SQL....\n");

        }//Cierre del catch

    }//Cierre método Ejecutarconsulta

    public ResultSet getCursor()
    {
        return tabla;

    }//Cierre método getCursor

    public void liberarCursor()
    {
        try
        {
            tabla.close();
            System.out.println("\nSe liberaron los recursos utilizados....(Cursor).\n");
        }//Cierre del try
        catch(SQLException e)
        {
            System.out.println("\nNo fue posible cerrar el cursor...\n");
        }//Cierre del catch

    }//Cierre método liberarCursor

}//Cierre clase Cursor
```

PAQUETE PERSISTENCIA

- **Clase Acabado**

```
package Persistencia;

import java.util.Vector;

//Declaración de la clase Acabado
public class Acabado {
    protected int id_acabado;
    protected String des_acabado;
    protected int cod_tipo_acabado;

    protected Vector arrayAcabado;

    public Acabado()
    {
        id_acabado = 0;
        des_acabado = "";
        cod_tipo_acabado = 0;

        arrayAcabado = new Vector();
        asignarTamanioVector();

    }//Cierre método constructor

    protected void asignarTamanioVector()
    {
        arrayAcabado.setSize(3);

    }//Cierre método asignarTamanioVector
```

```
    public void setidAcabado(int id_ac)
    {
        id_acabado = id_ac;
        arrayAcabado.setElementAt(id_acabado, 0);

    }//Cierre método setidAcabado

    public int getidAcabado()
    {
        return id_acabado;

    }//Cierre método getidAcabado

    public void setDescripcion(String des_ac)
    {
        des_acabado = des_ac;
        arrayAcabado.setElementAt(des_acabado, 1);

    }//Cierre método setDescripcion

    public String getDescripcion()
    {
        return des_acabado;

    }//Cierre método getDescripcion


public void setCod_tipoacabado(int cod_tipaca)
    {
        cod_tipo_acabado = cod_tipaca;
        arrayAcabado.setElementAt(cod_tipo_acabado, 2);
```

```
    }//Cierre método setCod_tipoacabado

    public int getCod_tipoacabado()
    {
        return cod_tipo_acabado;

    }//Cierre método getCod_tipoacabado


    public Vector getAcabado()
    {
        return arrayAcabado;

    }//Cierre método getAcabado

}//Cierre clase Acabado
```

- **Clase Ciudad**

```
package Persistencia;

import java.util.Vector;

//Declaración de la clase Ciudad
public class Ciudad {
    protected int id_ciudad;
    protected String nom_ciudad;


    protected Vector arrayCiudad;
```

```
public Ciudad()
{
    id_ciudad = 0;
    nom_ciudad = "";


    arrayCiudad = new Vector();
    asignarTamanioVector();

}//Cierre método constructor

protected void asignarTamanioVector()
{
    arrayCiudad.setSize(2);

}//Cierre método asignarTamanioVector

public void setidCiudad(int id_ciu)
{
    id_ciudad = id_ciu;
    arrayCiudad.setElementAt(id_ciudad, 0);

}//Cierre método setidCiudad

public int getidCiudad()
{
    return id_ciudad;

}//Cierre método getCiudad
```

```
    public void setNombre(String nombre_ciu)
    {
        nom_ciudad = nombre_ciu;
        arrayCiudad.setElementAt(nom_ciudad, 1);

    }//Cierre método setNombre

    public String getNombre()
    {
        return nom_ciudad;

    }//Cierre método getNombre




    public Vector getCiudad()
    {
        return arrayCiudad;

    }//Cierre método getCiudad

}//Cierre clase Ciudad
```

- **Clase Cliente**

```
package Persistencia;
```

```
import java.util.Vector;

//Declaración de la clase Cliente
public class Cliente {
    protected int id_cliente;
    protected String razon_social_cliente;
    protected String nit_cliente;
    protected String dir_cliente;
    protected String tel_cliente;
    protected int cod_ciudad;

    protected Vector arrayCliente;

    public Cliente()
    {
        id_cliente = 0;
        razon_social_cliente = "";
        nit_cliente = "";
        dir_cliente = "";
        tel_cliente = "";
        cod_ciudad = 0;

        arrayCliente = new Vector();
        asignarTamanioVector();

    }//Cierre método constructor

    protected void asignarTamanioVector()
    {
        arrayCliente.setSize(6);
```

```
}//Cierre método asignarTamanioVector


public void setidCliente(int id_cl)
{
    id_cliente = id_cl;
    arrayCliente.setElementAt(id_cliente, 0);

}//Cierre método setidCliente

public int getidCliente()
{
    return id_cliente;

}//Cierre método getCliente

public void setNombre(String razon_cl)
{
    razon_social_cliente = razon_cl;
    arrayCliente.setElementAt(razon_social_cliente, 1);

}//Cierre método setNombre

public String getNombre()
{
    return razon_social_cliente;

}//Cierre método getNombre


public void setNit(String nit_cl)
```

```
{
    nit_cliente = nit_cl;
    arrayCliente.setElementAt(nit_cliente, 2);

}//Cierre método setNit

public String getNit()
{
    return nit_cliente;

}//Cierre método getNit

public void setDireccion(String dir_cl)
{
    dir_cliente = dir_cl;
    arrayCliente.setElementAt(dir_cliente, 3);

}//Cierre método setDireccion

public String getDireccion()
{
    return dir_cliente;

}//Cierre método getDireccion

public void setTelefono(String tel_cl)
{
    tel_cliente = tel_cl;
    arrayCliente.setElementAt(tel_cliente, 4);

}//Cierre método setTelefono
```

```
    public String getTelefono()
    {
        return tel_cliente;

    }//Cierre método getTelefono

    public void setcodCiudad(int codciudad_cl)
    {
        cod_ciudad = codciudad_cl;
        arrayCliente.setElementAt(cod_ciudad, 5);

    }//Cierre método setidCliente

    public int getcodCiudad()
    {
        return cod_ciudad;

    }//Cierre método getCliente



    public Vector getCliente()
    {
        return arrayCliente;

    }//Cierre método getCliente

}//Cierre clase Cliente
```

- **Clase Detalle_Pedido**

```
package Persistencia;

import java.util.Vector;

//Declaración de la clase DetallePedido

public class DetallePedido {
    protected int id_detalle_pedido;
    protected int cant_detalle;
    protected int vtotal_detalle ;
    protected int cod_pedido ;
    protected int cod_producto;


    protected Vector arrayDetallePedido;

    public DetallePedido()
    {
    id_detalle_pedido= 0;
    cant_detalle= 0;
    vtotal_detalle= 0 ;
    cod_pedido= 0;
    cod_producto= 0;

        arrayDetallePedido = new Vector();
        asignarTamanioVector();

    }//Cierre método constructor

    protected void asignarTamanioVector()
```

```
{
    arrayDetallePedido.setSize(5);

}//Cierre método asignarTamanioVector


public void setidDetallePedido(int id_dpd)
{
    id_detalle_pedido = id_dpd;
    arrayDetallePedido.setElementAt(id_detalle_pedido, 0);

}//Cierre método setidDetallePedido

public int getidDetallePedido()
{
    return id_detalle_pedido;

}//Cierre método getDetallePedido

public void setCantidad(int cant_dpd)
{
    cant_detalle = cant_dpd;
    arrayDetallePedido.setElementAt(cant_detalle, 1);

}//Cierre método setCantidad

public int getCantidad()
{
    return cant_detalle;

}//Cierre método getCantidad
```

```
    public void setVtotal(int vtotal_dpd)
    {
        vtotal_detalle = vtotal_dpd;
        arrayDetallePedido.setElementAt(vtotal_detalle, 2);

    }//Cierre método setVtotal

    public int getVtotal()
    {
        return vtotal_detalle;

    }//Cierre método getVtotal


public void setCodPedido(int cod_pd)
    {
        cod_pedido = cod_pd;
        arrayDetallePedido.setElementAt(cod_pedido, 3);

    }//Cierre método setCodPedido

    public int getCodPedido()
    {
        return cod_pedido;

    }//Cierre método getCodPedido

    public void setCodProducto(int cod_prod)
    {
```

```
        cod_producto = cod_prod;
        arrayDetallePedido.setElementAt(cod_producto, 4);

    }//Cierre método setCodProducto

    public int getCodProducto()
    {
        return cod_producto;

    }//Cierre método getCodProducto

    public Vector getDetallePedido()
    {
        return arrayDetallePedido;

    }//Cierre método getDetallePedido

}//Cierre clase DetallePedido
```

- **Clase DetalleRemision**

```
package Persistencia;

import java.util.Vector;

//Declaración de la clase DetalleRemision

public class DetalleRemision {
    protected int id_detalle_remision;
```

```
 protected int cant_remision;
 protected int subtotal_remision ;
 protected int cod_remision ;
 protected int cod_producto;
protected int cod_acabado;

 protected Vector arrayDetalleRemision;

 public DetalleRemision()
 {
  id_detalle_remision=0;
  cant_remision=0;
  subtotal_remision=0;
  cod_remision=0;
  cod_producto=0;
  cod_acabado=0;

     arrayDetalleRemision = new Vector();
     asignarTamanioVector();

 }//Cierre método constructor

 protected void asignarTamanioVector()
 {
     arrayDetalleRemision.setSize(6);

 }//Cierre método asignarTamanioVector


 public void setidDetalleRemision (int id_drem)
 {
```

```
        id_detalle_remision = id_drem;
        arrayDetalleRemision.setElementAt(id_detalle_remision, 0);

    }//Cierre método setidDetalleRemision

    public int getidDetalleRemision()
    {
        return id_detalle_remision;

    }//Cierre método getDetalleRemision

    public void setCantidad(int cant_drem)
    {
        cant_remision = cant_drem;
        arrayDetalleRemision.setElementAt(cant_remision, 1);

    }//Cierre método setCantidad

    public int getCantidad()
    {
        return cant_remision;

    }//Cierre método getCantidad




public void setCodRemision(int cod_drem)
    {
        cod_remision = cod_drem;
```

```
    arrayDetalleRemision.setElementAt(cod_remision, 3);

}//Cierre método setCodRemision

public int getCodRemision()
{
    return cod_remision;

}//Cierre método getCodRemision

public void setCodProducto(int cod_prod)
{
    cod_producto = cod_prod;
    arrayDetalleRemision.setElementAt(cod_producto, 4);

}//Cierre método setCodProducto

public int getCodProducto()
{
    return cod_producto;

}//Cierre método getCodProducto

 public void setCodAcabado(int cod_ac)
{
    cod_acabado = cod_ac;
    arrayDetalleRemision.setElementAt(cod_acabado, 5);

}//Cierre método setCodAcabado

public int getCodAcabado()
```

```
    {
        return cod_acabado;

    }//Cierre método getCodAcabado

    public Vector getDetalleRemision()
    {
        return arrayDetalleRemision;

    }//Cierre método getDetalleRemision

}//Cierre clase DetalleRemision
```

- **Clase Pedido**

```
package Persistencia;

import java.util.Vector;

//Declaración de la clase Pedido
public class Pedido {
    protected int id_pedido;
    protected String num_pedido;
    protected String fecha_pedido;
    protected int subtotal_pedido ;
    protected int iva_pedido;
    protected int total_pedido;
    protected int cod_cliente;
     protected String estado_pedido;
```

```
protected Vector arrayPedido;

public Pedido()
{
 id_pedido= 0;
 num_pedido= "";
 fecha_pedido= "";
 subtotal_pedido= 0;
 iva_pedido= 0;
 total_pedido= 0;
 cod_cliente= 0;
 estado_pedido= "";
    arrayPedido = new Vector();
    asignarTamanioVector();

}//Cierre método constructor

protected void asignarTamanioVector()
{
    arrayPedido.setSize(8);

}//Cierre método asignarTamanioVector


public void setidPedido(int id_pd)
{
    id_pedido = id_pd;
    arrayPedido.setElementAt(id_pedido, 0);

}//Cierre método setidPedido
```

```
public int getidPedido()
{
    return id_pedido;

}//Cierre método getPedido

public void setNumero(String num_pd)
{
    num_pedido = num_pd;
    arrayPedido.setElementAt(num_pedido, 1);

}//Cierre método setNumero

public String getNumero()
{
    return num_pedido;

}//Cierre método getNumero


public void setFecha(String fecha_pd)
{
    fecha_pedido = fecha_pd;
    arrayPedido.setElementAt(fecha_pedido, 2);

}//Cierre método setFecha

public String getFecha()
{
    return fecha_pedido;
```

```
}//Cierre método getFecha

public void setSubtotal(int subt_pd)
{
    subtotal_pedido = subt_pd;
    arrayPedido.setElementAt(subtotal_pedido, 3);

}//Cierre método setSubtotal

public int getSubtotal()
{
    return subtotal_pedido;

}//Cierre método getSubtotal

public void setIva(int iva_pd)
{
    iva_pedido = iva_pd;
    arrayPedido.setElementAt(iva_pedido, 4);

}//Cierre método setIva

public int getIva()
{
    return iva_pedido;

}//Cierre método getIva

public void setTotal(int total_pd)
{
    total_pedido = total_pd;
```

```
        arrayPedido.setElementAt(total_pedido, 5);

    }//Cierre método setTotal

    public int getTotal()
    {
        return total_pedido;

    }//Cierre método getTotal

public void setCodCliente(int cod_cl)
    {
        cod_cliente = cod_cl;
        arrayPedido.setElementAt(cod_cliente, 6);

    }//Cierre método setCodCliente

    public int getCodCliente()
    {
        return cod_cliente;

    }//Cierre método getCodCliente

    public void setEstado(String estado_pd)
    {
        estado_pedido = estado_pd;
        arrayPedido.setElementAt(estado_pedido, 7);

    }//Cierre método setFecha

    public String getEstado()
```

```
    {
        return estado_pedido;

    }//Cierre método getFecha
    public Vector getPedido()
    {
        return arrayPedido;

    }//Cierre método getPedido

}//Cierre clase Pedido
```

- **Clase Producto**

```
package Persistencia;

import java.util.Vector;

//Declaración de la clase Producto
public class Producto {
    protected int id_producto;
    protected String des_producto;
    protected int valor_uni_producto;
    protected int cod_tipo_producto;

    protected Vector arrayProducto;

    public Producto()
    {
```

```
    id_producto = 0;
    des_producto = "";
    valor_uni_producto = 0;
    cod_tipo_producto = 0;

    arrayProducto = new Vector();
    asignarTamanioVector();

}//Cierre método constructor

protected void asignarTamanioVector()
{
    arrayProducto.setSize(4);

}//Cierre método asignarTamanioVector

public void setidProducto(int id_pro)
{
    id_producto = id_pro;
    arrayProducto.setElementAt(id_producto, 0);

}//Cierre método setidProducto

public int getidProducto()
{
    return id_producto;

}//Cierre método getProducto

public void setDescripcion(String des_pro)
{
```

```
        des_producto = des_pro;
        arrayProducto.setElementAt(des_producto, 1);

    }//Cierre método setDescripcion

    public String getDescripcion()
    {
        return des_producto;

    }//Cierre método getDescripcion


public void setVunitario(int vunit_pro)
    {
        valor_uni_producto = vunit_pro;
        arrayProducto.setElementAt(valor_uni_producto, 2);

    }//Cierre método setVunitario

    public int getVunitario()
    {
        return valor_uni_producto;

    }//Cierre método getVunitario

    public void setCod_Tipoproducto(int cod_tipro)
    {
        cod_tipo_producto = cod_tipro;
        arrayProducto.setElementAt(cod_tipo_producto, 3);

    }//Cierre método setCod_Tipoproducto
```

```
    public int getCod_Tipoproducto()
    {
        return cod_tipo_producto;

    }//Cierre método getCod_Tipoproducto


    public Vector getProducto()
    {
        return arrayProducto;

    }//Cierre método getProducto

}//Cierre clase Producto
```

- **Clase Remision**

```
package Persistencia;

import java.util.Vector;

//Declaración de la clase Remision
public class Remision {
    protected int id_remision;
    protected String num_remision;
    protected String fecha_remision;
    protected int cod_cliente;
    protected String estado_remision;
```

```
protected Vector arrayRemision;

public Remision()
{
 id_remision= 0;
 num_remision= "";
 fecha_remision= "";
 cod_cliente= 0;
 estado_remision = "";

    arrayRemision = new Vector();
    asignarTamanioVector();

}//Cierre método constructor

protected void asignarTamanioVector()
{
    arrayRemision.setSize(5);

}//Cierre método asignarTamanioVector



public void setidRemision(int id_rem)
{
    id_remision = id_rem;
    arrayRemision.setElementAt(id_remision, 0);

}//Cierre método setidRemision

public int getidRemision()
{
```

```
    return id_remision;

}//Cierre método getRemision

public void setNumero(String num_rem)
{
    num_remision = num_rem;
    arrayRemision.setElementAt(num_remision, 1);

}//Cierre método setNumero

public String getNumero()
{
    return num_remision;

}//Cierre método getNumero


public void setFecha(String fecha_rem)
{
    fecha_remision = fecha_rem;
    arrayRemision.setElementAt(fecha_remision, 2);

}//Cierre método setFecha

public String getFecha()
{
    return fecha_remision;

}//Cierre método getFecha
```

```
public void setCodCliente(int cod_cl)
    {
        cod_cliente = cod_cl;
        arrayRemision.setElementAt(cod_cliente, 3);

    }//Cierre método setCodCliente

    public int getCodCliente()
    {
        return cod_cliente;

    }//Cierre método getCodCliente

     public void setEstado(String estado_rem)
    {
        estado_remision = estado_rem;
        arrayRemision.setElementAt(estado_remision, 4);

    }//Cierre método setEstado

    public String getEstado()
    {
        return estado_remision;

    }//Cierre método getEstado

    public Vector getRemision()
    {
        return arrayRemision;
```

```
    }//Cierre método getRemision

}//Cierre clase Remision
```

12. DESARROLLO Y PRUEBAS

El desarrollo se llebo a cabo en JSP (HTML+ JAVA), la interfaz gráfica del proyecto se trabajó en HTML Y CSS y se trabajo MySql como servidor de base de datos.

13. CONCLUSIONES

Erlass se programó a partir de un análisis, diseño y un desarrollo orientados a objetos permitiendo mejorar las condiciones de seguridad de los datos, ya que con esto la información queda oculta y menos susceptible a ser robada o modificada por terceros, también admite una reutilización de código gracias a las clases implementadas en la capa de negocio, con ello la aplicación puede evolucionar o añadirse a otros sistemas de información orientados a la web que posea la empresa.

La capa de datos Normalizada en quinta forma permite optimizar el espacio disminuyendo la información redundante permitiendo así un fácil mantenimiento de las base de datos.

El diseño de la interfaz gráfica con listas desplegables permite una captura más óptima reduciendo el margen de errores que se pueden presentar en la digitación de la información, los colores utilizados, la fuente y la ubicación del texto facilita la lectura y permite al usuario un mejor ambiente de trabajo.

14. RECOMENDACIONES

Para el desarrollo de aplicaciones web se recomienda Manejar Windows XP o equivalente Linux Ubuntu, ya que estas plataformas son las más compatibles con los programas de diseño y desarrollo de soluciones.

REFERENCIAS

CAMPDERRICH, Benet. Ingeniería del Software. Barcelona 2003. Catalunya. Universitat Oberta de Catalunya, 320 p

DEITEL, Paul y DEITEL Harvey. Como programar en java. 2004 quinta edición. México. Pearson educacion. 1268 p

PRESSMAN, Roger. Ingeniería del software un enfoque practico. 2002. Quinta edición. Madrid .Mc-Graw-Hill. 601 p

ROMAN, Mauricio. Plan de negocios 2009. Industrias Román Ltda. 56 p

SUNMICROSYSTEM. Conozca más sobre la tecnología Java. Extraído el 14 de noviembre de 2009, de http://www.java.com/es/about/

TRIGOS García, Esteban Guía practica para usuarios JSP. 2001. Anaya Multimedia. 303 p.

WINBLAD, Ann y Otros. Software Orientado a Objetos. 1993. Madrid. Primera edición. DIAZ DE SANTOS, S.A. EDICIONES. 314 p.