

Universidad de las Ciencias Informáticas

Facultad 1



Título: Implementación de un Sistema de Control de Pago para el Departamento de Transportaciones Nacionales.

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

Autores: Mauris Yadira Cruz Guerrero

Yudisel Santana Pacheco

Tutor: Ing. Radel Calzada Pando

Tutor: Ing. Alberto Tamayo Ramos

Ciudad de La Habana

Julio de 2009



*"Los sueños de hoy serán las realidades de mañana".
José Martí.*

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado:

Implementación de un Sistema de Control de Pago para el Departamento de Transportaciones Nacionales.

y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Mauris Yadira Cruz Guerrero

Yudisel Santana Pacheco

Ing. Alberto Tamayo Ramos

Ing. Radel Calzada Pando

AGRADECIMIENTOS

Agradecerles a mis padres, a mi hermano, a mis abuelos y a toda mi familia, que siempre me han apoyado en todo lo que he hecho. A todas mis amistades, que han estado a mi lado toda mi vida universitaria. A Yaima Montero Quintana y a Jorge Luis Ramos Medina por apoyarme y estar conmigo en los momentos buenos y malos. Agradecerle aunque ya no estén aquí conmigo, a Arianna Rosabal por ser una de las personas más importantes en mi vida. A mis tutores Radel y Alberto, así como a Dayron Cruz por ayudarme tanto en la realización de este trabajo.

Yudisel

A mi familia por contribuir infinitamente en la realización de mis sueños.

A nuestro siempre Comandante Fidel Castro por permitirnos formar parte de este proyecto que ya es historia y una realidad.

A mis tutores Radel, Alberto y quien considero mi co-tutor Dayron por motivarnos a salir de los esquemas y a realizar un trabajo novedoso del cual me siento muy satisfecha.

A todas las personas que me han apoyado en el desarrollo del presente trabajo y de manera general durante todo el transcurso de vida universitaria. Gracias por influir positivamente en mi formación!

¡¡¡A todos muchas gracias!!!

Mauris

DEDICATORIA

Dedico este trabajo a mi mamá y a mi papá, por ser unas de las personas más grandes que tengo en este mundo, por brindarme su apoyo y amor en cada momento de mi vida y por hacer hasta lo imposible para que yo salga adelante.

❧ Yudisel Santana Pacheco ❧

A mi mamá por ser la persona más maravillosa del mundo que siempre ha estado conmigo apoyándome y dándome ánimos para seguir adelante. Eres lo que más quiero.

A mi padre y mis abuelos espero se sientan orgullosos de cuanto he logrado hasta hoy.

A mi gran amiga Liudmila la que siempre ha estado a mi lado en todos los momentos en que más la he necesitado siendo la hermana que nunca tuve!

A mis tíos y mis primos por ser tan buenos y por ayudarme mucho durante mi vida.

A mis amistades que han compartido conmigo durante estos 5 años los buenos y malos momentos de mi vida universitaria!

❧ Mauris Yadira Cruz Guerrero ❧

RESUMEN

El presente trabajo pretende mejorar todo lo referente a los procesos de pago de dietas y facturas que se realizan en el Departamento de Transportación Nacional de la Universidad de las Ciencias Informáticas. La forma en que se realizan actualmente estos procesos es algo difícil debido a que se realizan de forma manual, por lo que surge la necesidad de informatizar los mismos. Es por ello, que este trabajo persigue como objetivo desarrollar un sistema de control de pago para el Departamento de Transportación Nacional y así brindar una mejora a la hora de la gestión de la información manejada en estos procesos. Para su desarrollo se realizó un amplio estudio de los procesos implicados en el tema y del diseño anteriormente propuesto, con el objetivo de garantizar un mejor funcionamiento del sistema que se implementará. Se tuvieron en cuenta las herramientas y tecnologías actuales a ser usadas, las cuales fueron anteriormente propuestas y además forman parte del grupo de herramientas libres y de código abierto; así como la generación de la documentación asociada durante la implementación de dichos módulos. Finalmente se hicieron pruebas para verificar la calidad, seguridad, confiabilidad, y disponibilidad, con el objetivo de obtener la aceptación del cliente.

PALABRAS CLAVES

DTN: Departamento de Transportaciones Nacionales UCI.

UCI: Universidad de las Ciencias Informáticas.

VR-E: Vicerrector Económico.

ÍNDICE

INTRODUCCIÓN.....	1
CAPÍTULO 1: “FUNDAMENTACIÓN TEÓRICA”.....	4
1.1. INTRODUCCIÓN	4
1.2. CONCEPTOS FUNDAMENTALES.....	4
1.3. SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS A LOS PROCESOS DE PAGO DE DIETAS Y FACTURAS POR CONCEPTO DE TRANSPORTACIÓN	6
1.3.1 Sistemas internacionales.....	6
1.3.2 Sistemas nacionales.....	8
1.4. TENDENCIAS ACTUALES DE LA TECNOLOGÍA EN EL DESARROLLO DE SOFTWARE.....	8
1.4.1 Lenguajes de Programación.....	9
1.4.2 Tecnología para el desarrollo web.....	11
1.4.3 Servidor de Aplicaciones	12
1.4.4 Gestor de Base de datos	12
1.4.5 IDE de Programación	13
1.4.6 Framework.....	14
1.5. PROPUESTA DE SOLUCIÓN	15
1.6. CONCLUSIONES	15
CAPÍTULO 2: “DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA”.....	16
2.1 INTRODUCCIÓN	16
2.2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.....	16
2.2.1 Autenticar usuario.....	16
2.2.2 Gestionar dieta	16
2.2.3 Gestionar factura.....	20
2.2.4 Visualizar resumen	22
2.2.5 Obtener modelos	23
2.2.6 Obtener reportes	24
2.2.7 Obtener cantidad de dietas y facturas mensuales.....	25
2.2.8 Gestionar dieta por bonificación.....	26

2.3	VALORACIÓN CRÍTICA DEL DISEÑO PROPUESTO POR EL ANALISTA	27
2.3.1	Modelo de la BD	28
2.3.2	Requerimientos no funcionales.....	28
2.4	PATRONES O ESTILOS ARQUITECTÓNICO PRESENTE EN LA SOLUCIÓN PROPUESTA.....	30
2.4.1	Patrón Modelo Vista Controlador (MVC).....	30
2.4.2	Arquitectura Cliente/Servidor	31
2.5	MODELO DE IMPLEMENTACIÓN	32
2.5.1	Modelo de despliegue.....	32
2.5.2	Diagrama de componentes.....	33
2.6	SEGURIDAD DEL SISTEMA.....	36
2.7	ANÁLISIS DE POSIBLES IMPLEMENTACIONES Y COMPONENTES O MÓDULOS QUE SON REHUSADOS	37
2.8	DESCRIPCIÓN DE LAS NUEVAS CLASES U OPERACIONES NECESARIAS	37
2.8.1	Clases Controladoras	37
2.8.2	Clases Modelos	40
2.9	CONCLUSIONES	44
CAPÍTULO 3: “PRUEBA DE LA SOLUCIÓN PROPUESTA”		45
3.1	INTRODUCCIÓN	45
3.2	PRUEBAS APLICADAS.....	45
3.3	PRUEBAS DE UNIDAD	45
3.3.1	Pruebas de caja blanca	46
3.3.2	Pruebas de caja negra.....	48
3.4	DESCRIPCIÓN DE LAS PRUEBAS DE CAJA NEGRA REALIZADAS.....	48
3.4.1	Caso de uso: Gestionar dieta.	48
3.4.2	Caso de uso: Gestionar Factura.....	54
3.4.3	Caso de uso: Gestionar bonificación	60
3.5	CONCLUSIONES	65
CONCLUSIONES GENERALES.....		66
RECOMENDACIONES		67

REFERENCIAS BIBLIOGRÁFICAS	68
BIBLIOGRAFÍA.....	69
GLOSARIO DE TÉRMINOS	70
ANEXOS.....	71

ÍNDICE DE TABLAS

TABLA 1: DESCRIPCIÓN DE LA CLASE FACTURACONTROLLER.	38
TABLA 2: DESCRIPCIÓN DE LA CLASE DIETACONTROLLER.....	38
TABLA 3: DESCRIPCIÓN DE LA CLASE BONIFICACIÓNCONTROLLER.....	39
TABLA 4: DESCRIPCIÓN DE LA CLASE REPORTECONTROLLER.....	40
TABLA 5: DESCRIPCIÓN DE LA CLASE USUARIOCONTROLLER.....	40
TABLA 6: DESCRIPCIÓN DE LA CLASE FACTURAMODEL.....	41
TABLA 7: DESCRIPCIÓN DE LA CLASE DIETAMODEL.....	42
TABLA 8: DESCRIPCIÓN DE LA CLASE BONIFICACIÓNMODEL.....	43
TABLA 9: DESCRIPCIÓN DE LA CLASE REPORTEMODEL.....	43
TABLA 10 COMPLEJIDAD CICLOMÁTICA VS EVALUACIÓN DE RIESGO.....	46
TABLA 11 PRUEBA REALIZADA A LA FUNCIONALIDAD INSERTAR DIETA.....	50
TABLA 12 PRUEBA REALIZADA A LA FUNCIONALIDAD MODIFICAR DIETA.....	52
TABLA 13 PRUEBA REALIZADA A LA FUNCIONALIDAD VISUALIZAR DIETA.....	53
TABLA 14 PRUEBA REALIZADA A LA FUNCIONALIDAD ELIMINAR DIETA.....	54
TABLA 15 PRUEBA REALIZADA A LA FUNCIONALIDAD INSERTAR FACTURA.....	56
TABLA 16 PRUEBA REALIZADA A LA FUNCIONALIDAD MODIFICAR FACTURA.....	57
TABLA 17 PRUEBA REALIZADA A LA FUNCIONALIDAD VISUALIZAR FACTURA.....	58
TABLA 18 PRUEBA REALIZADA A LA FUNCIONALIDAD ELIMINAR FACTURA.....	59
TABLA 19 PRUEBA REALIZADA A LA FUNCIONALIDAD INSERTAR BONIFICACIÓN.....	61
TABLA 20 PRUEBA REALIZADA A LA FUNCIONALIDAD MODIFICAR BONIFICACIÓN.....	63
TABLA 21 PRUEBA REALIZADA A LA FUNCIONALIDAD VISUALIZAR BONIFICACIÓN.....	64
TABLA 22 PRUEBA REALIZADA A LA FUNCIONALIDAD ELIMINAR BONIFICACIÓN.....	65

ÍNDICE DE FIGURAS

Figura 1 Autenticar usuario.....	16
Figura 2 Lista de las dietas existentes hasta el momento.....	17
Figura 3 Solicitud para obtener una dieta.....	17
Figura 4 Insertar una dieta según el tipo de dieta deseado (Anticipo o Liquidación).....	18
Figura 5 Modificar una dieta determinada.....	19
Figura 6 Eliminar una dieta determinada.....	19
Figura 7 Lista de las facturas existentes hasta el momento.....	20
Figura 8 Solicitud para obtener una factura determinada.....	20
Figura 9 Insertar factura especificando el tipo de pago (Cheque).....	21
Figura 10 Modificar una dieta existente.....	22
Figura 11 Eliminar una dieta existente.....	22
Figura 12 Obtener resumen de las dietas y facturas existentes hasta el momento.....	23
Figura 13 Visualizar modelo.....	24
Figura 14 Imprimir el modelo resultante.....	24
Figura 15 Buscar todas las dietas y facturas existentes por criterios de área, fecha de ingreso y fecha de pago.....	25
Figura 16 Dietas y facturas en un rango de fecha determinado con el saldo especificado.....	25
Figura 17 Dietas o facturas correspondientes al mes y año especificado.....	26
Figura 18 Dieta o factura registrada en el mes y año especificado.....	26
Figura 19 Dietas existentes hasta el momento.....	27
Figura 20 Despliegue del sistema.....	32
Figura 21 Diagrama de Componentes (Global).....	33
Figura 22 Diagrama Componentes (Modelos).....	34
Figura 23 Diagrama de Componentes (Vistas).....	35
Figura 24 Diagrama de Componentes (Controladoras).....	36

INTRODUCCIÓN

Debido a la creciente evolución y desarrollo del uso y manejo de la información en el mundo actual, se necesita recurrir a las más diversas opciones con el fin de mantener un control eficaz sobre estas. Siendo así indispensable el uso de las Ciencias Informáticas como una tecnología vital para dar solución a los problemas que persisten en la sociedad. Por lo que son necesarias para lograr un perfeccionamiento de las complicaciones existentes de forma sostenible e incremental.

El país no es inmune a estos problemas, por lo que ha tenido que buscar soluciones para lograr un mayor desarrollo tecnológico, contando con el apoyo del capital humano existente. Prueba de ello es la creación de la Universidad de las Ciencias Informáticas (UCI), donde el mayor reto es la formación de jóvenes a profesionales que vinculan el estudio con la producción. Por lo que se ha convertido en un centro clave para la producción de software.

Las empresas e instituciones cubanas realizan diferentes procesos de pago, tradicionalmente de manera manual pues de alguna forma siempre se han visto dificultados debido a la situación de bloqueo que tiene nuestro país, lo cual le imposibilita adquirir algunas herramientas para realizar eficientemente este proceso. El pago puede ser realizado al contado o aplazado (último término que no se realiza en el DTN UCI).

En las instituciones de Educación Superior de nuestro país existen diferentes procesos de pago. Los principales son el pago de dietas a profesores y trabajadores externos al centro que realizan alguna tarea en colaboración con el mismo, y el pago del 50% del pasaje a estudiantes como concepto de dieta por bonificación aprobado luego del VII Congreso de la FEU. Otro proceso de pago importante lo constituye el pago de factura a organismos externos a las universidades, que prestan algún servicio.

La Universidad de las Ciencias Informáticas como institución estudiantil de nuestro país posee una Vicerrectoría Económica que se subdivide en diferentes departamentos de trabajo que de una forma u otra laboran de forma conjunta. Esta Vicerrectoría además de atender la Dirección de Economía y Planificación atiende el DTN. El mismo atiende el movimiento de transporte del centro relacionado con las empresas del país que nos prestan el servicio.

Actualmente estos procesos de pago se realizan de forma no informatizada en el DTN UCI, lo que implica entre otras dificultades, que no se tiene acceso al sistema contable de la dirección de economía del centro, puede existir pérdida de información así como demora en la realización de estos procesos, por todas estas dificultades se informatizará estos procesos mediante el desarrollo de un sistema de control de pago relacionado con los servicios de transportación nacional, brindados a nuestra universidad, partiendo de un diseño ya propuesto.

De la situación anterior se tiene el siguiente **problema**: ¿Cómo mejorar el proceso de gestión de la información relacionada con el control de pago en el DTN de la UCI?

El **objeto de estudio** consiste en él: Proceso de gestión de la información de los procesos de pago. El **campo de acción** está enmarcado en el Proceso de gestión de la información de los procesos de control de pago en el DTN de la UCI.

Idea a defender: El desarrollo de una aplicación informática mejorará el proceso de gestión de la información relacionada con el control de pago en el DTN de la UCI.

Para dar solución a la problemática antes mencionada se ha definido como **objetivo general**: Desarrollar una aplicación informática que mejore el proceso de gestión de la información relacionada con el control de pago en el DTN de la UCI.

De los que se derivan los siguientes **objetivos específicos**:

- Realizar el marco teórico de la investigación.
- Implementar un sistema para informatizar los procesos de registro, control y reporte de facturas y dietas.
- Realizar pruebas unitarias al sistema propuesto.

Dentro de las tareas científicas de la investigación que se propone para dar solución a los objetivos se encuentran:

Tareas Conjuntas:

- Revisar la bibliografía referente a los diferentes sistemas existentes vinculados a los procesos de pagos.
- Evaluar el resultado del trabajo de diploma Diseño de un Sistema de Control de Pago para el Departamento de Transportaciones Nacionales, para dar continuidad al desarrollo de la solución.
- Realizar el análisis de las tecnologías y herramientas propuestas con anterioridad para la implementación del sistema.
- Implementación del sistema relacionada con el control de pago en el DTN de la UCI.
- Elaboración de los casos de pruebas unitarias.

Mauris Yadira Cruz Guerrero:

- Implementación de los módulos de Administración y Dieta.

Yudisel Santana Pacheco:

- Implementación de los módulos de Registro y Factura.

Como **posible resultado** se espera obtener una aplicación informática que mejore el proceso de gestión de la información relacionada con el control de pago en el DTN de la UCI.

Con el sistema que se implementará se obtendrían los siguientes **beneficios**: los procesos de control de pago serán informatizados, favoreciendo el incremento de la rapidez en los procesos de pago que se desarrollan en el DTN.

Métodos de Investigación Científica

Entre los **métodos teóricos** que se emplearán para nuestra investigación está:

- **Analítico-Sintético**: Se estudiaron las diferentes Resoluciones que se utilizan en nuestro centro dentro de la dirección señalada. Se consultará bibliografía especializada referente a temas de pagos de Dietas y Facturas. Se trabajará con la documentación existente de cada una de las formas de pago que ya se han venido realizando.
- **Hipotético deductivo**: La investigación sigue además un método hipotético deductivo porque a partir del problema concreto se plantean objetivos específicos e hipótesis que en el transcurso de la investigación son resueltos siguiendo métodos científicamente bien fundamentados.
- **Sistémico**: En cada caso se plantea el problema como un todo, con el fin de lograr el desarrollo de un sistema lo suficientemente flexible y robusto. Donde la propia dinámica del procedimiento de pago de dietas y facturas se funda en un sistema sostenible e integral.

Como **método empírico** utilizado para cumplir con las tareas se empleará:

- **La entrevista**: Se entrevistarán a los funcionarios de la Dirección de Economía del centro así como a trabajadores del Departamento de Transportaciones Nacionales UCI, con el objetivo de comprender a fondo el negocio en el que está enmarcado el problema a resolver y obtener toda la información acerca de las funcionalidades del sistema según los requerimientos del cliente.
- **Observación científica**: Se utiliza para analizar el curso natural del problema y del objeto de estudio con el fin de extraer mediante la observación las principales características identificativas y fenoménicas.

CAPÍTULO 1: “FUNDAMENTACIÓN TEÓRICA”

1.1. Introducción

En este capítulo se realiza un análisis de las diferentes aplicaciones similares al sistema propuesto tanto a nivel nacional como internacional. Se describen además las herramientas y tecnologías que se emplearon para el desarrollo del mismo, así como la descripción de los principales conceptos asociados al sistema y las clases que se consideran críticas en la implementación de la aplicación.

1.2. Conceptos fundamentales

Proceso de pago: Es un servicio en el cual se ve implicado el sistema financiero de cualquier persona o entidad que participe. Uno de los factores que influirá será la forma de pago que se vaya a liquidar el servicio que se ha dado. El pago puede ser realizado al contado o aplazado.

Proceso de pago de dieta en Cuba

En nuestro país este proceso se rige a partir de normativas puestas por el Ministerio de Finanzas y Precios el cual expone que el pago de Dietas son autorizados a partir de anticipos y liquidación y otros gastos en que incurran los funcionarios, empleados y personas designadas por la entidad en el ejercicio de las funciones que se les encomiende. Este proceso de pago de dietas procede a partir de que debe existir en cada empresa un funcionario que autorice el pago de dieta a través del modelo de Anticipo y Liquidación de gastos, el cual es autorizado a cobrar en la caja de la empresa rigiéndose por los documentos que justifiquen el pago, pues toda dieta solicitada debe tener un justificante que avale el pago; sin este aval no se procede a efectuar el autorizo de pago.

Proceso de pago de factura en Cuba

Las diferentes empresas Cubanas y Extranjeras que radican en Cuba prestan diferentes servicios a los residentes en el país un ejemplo de ellas es ETECSA (Empresa de Telecomunicaciones de Cuba S.A) que presta servicios telefónicos en Cuba pues es extranjera y posee diferentes tipos de acceso de pago de las diferentes facturas que se generan todos los meses por el consumo que se desarrollan por cada línea telefónica existente. Todo el residente en la isla puede acceder a pagar la Factura de cada mes a través de 2 servicios principales que se ofertan: Servicio por Factura en formato duro o Servicio por Pago a través de Tarjetas Magnéticas.

Otras empresas que prestan servicios requieren del pago del servicio de forma presencial y es cuando le prestan un servicio a una entidad o particular y este debe remitirse a la Oficina Comercial de la Empresa que le da el servicio y paga la Factura que se origina.

Cuando se hacen compras a Empresas Extranjeras a distancia se puede efectuar el pago a través de Cheques o Transferencias Bancarias siempre y cuando se tenga la Factura que se emite al servicio

que se solicita y entonces se procede a depositar el consumo a través de alguna forma de pago ya mencionada anteriormente.

Sistema de pago de dietas en la UCI

La universidad paga dietas por diferentes tipos de conceptos:

- Dietas por Gastos Viáticos (Transporte y Bonificación).
- Dietas por Pagos Anticipados.
- Dietas de Liquidación.

Todo el proceso de dieta se rige a partir del tipo de concepto por el cual se solicite. Las dietas por pagos anticipados se realizan a aquellas personas que solicitan que se les ayude con una proporción de dinero de la caja de la universidad para realizar alguna tarea del centro pero tienen la obligación de liquidar el préstamo a su regreso. La liquidación de la dieta se efectúa cuando una persona consume de su bolsillo la tarea que va a cumplir y a su regreso al centro solicita que se le pague todo lo que consumió ya sea por concepto de alimentación o de transporte (Gastos Viáticos) todas estas dietas siempre serán autorizadas por un Vicerrector o persona que esté acreditada en la caja de la universidad para poder cobrar la dieta.

La universidad cuenta con un nuevo servicio de pago de dieta que surge a partir del VII Congreso de la FEU donde sale a relucir el pago del 50% del Pasaje Estudiantil: Bonificación.

La bonificación es un proceso en el cual el estudiante puede viajar todos los meses a su casa si lo desea y solo tiene derecho de realizar por cada mes del curso una bonificación, la cual consiste en reintegrarle al estudiante el 50% del pasaje que consuma. Para poder efectuar la bonificación el departamento de transportaciones posee la Resolución No 300/2007 de la universidad la cual ampara que el estudiante antes de proceder con solicitar la bonificación debe remitirse a ver a su decano el cual emitirá el modelo de bonificación. Es importante que el estudiante conozca la Resolución, pues para proceder en la solicitud del modelo de bonificación y de no perder el dinero consumido por pasarse de tiempo de entrega, debe entregar el modelo de bonificación que emite el decano, más los modelos de comprobantes de viajes dentro de un plazo de 72 horas de su arribo al centro y entonces el Departamento de Transportaciones Nacionales UCI procede en realizar la bonificación.

Para realizar el pago el departamento recepciona durante la semana la cantidad de bonificaciones que se presenten y todos los viernes por la tarde son llevadas al vicerrector Económico de la universidad para que sean firmadas y autorizadas, al terminar este proceso el departamento procede en sacar de la caja de la universidad las bonificaciones y luego en la oficina se le pasa un correo a los estudiantes que recibirán el pago para que pasen a cobrarlo, para esto el estudiante tiene solo una semana para pasar a cobrar porque si no se le reintegra el dinero a la caja y no se puede extraer nuevamente.

Sistema de pago de factura en la UCI

El DTN de la Universidad realiza pago de Factura a diferentes empresas del país que tienen que ver con los procesos de transporte.

Para realizar el proceso de pago la universidad, contrata a través del MITRANS el tipo de transporte que se utilizará, cuando esto sucede las empresas que prestan el servicio emiten una factura por el consumo realizado y esta es enviada al DTN UCI el cual procede a revisar la factura recibida y la presenta a través de un modelo de Solicitud de Pago al comité de compra de la universidad, luego de aprobado el pago la factura pasa a manos de la Dirección de Planificación y Estadística quien rebaja del presupuesto asignado al área de Transporte a la que pertenece este departamento y luego de actualizarse los datos de presupuesto pasa a manos del Departamento de Finanzas quienes se encargan de elaborar el tipo de pago solicitado, pues se pueden hacer de 2 formas, como se explica anteriormente.

Después de realizado todo esto, el DTN UCI espera una semana para confirmar en el Departamento de Finanzas y solicitar los datos del tipo de pago que se realizó y de acuerdo al tipo de pago se procederá a pagarle a la empresa que prestó el servicio.

1.3. Sistemas automatizados existentes vinculados a los procesos de pago de dietas y facturas por concepto de transportación

A nivel internacional como nacional existen sistemas que realizan los procesos de pagos fundamentales que se efectúan en el DTN. Los sistemas a los que se hacen mención a continuación no son factibles para dar solución a la problemática existente debido a que en el departamento se labora con informaciones precisas, es decir, el proceso de pago dieta como se explicaba anteriormente se realiza mediante modelos emitidos por el Ministerio de Finanzas y Precios, igualmente para el proceso de pago de la factura, por lo que estos sistemas no presentan algunos de los datos que se registran mediante los mismos, además las aplicaciones internacionales aunque realizan procesos similares son enfocadas para empresas de transporte de mercancías por carretera, por lo que muestran otras funcionalidades que no son viables para el DTN.

1.3.1 Sistemas internacionales

CSGTR01

CSGTR01 es una solución de gestión para pequeñas y medianas empresas producida por CDS SOFT.

[1] Es una aplicación informática de gestión, de fácil utilización, diseñada especialmente para las empresas de transporte de mercancías por carretera, además de permitir una gran visibilidad sobre los

costes de su flota de vehículos. El sistema permite simplificar la gestión de los procesos de compras, ventas, almacén, gestión de cobros y pagos, así como el control de costes de la flota de vehículos. La facturación pasará a ser un proceso sencillo, acortándose considerablemente los tiempos empleados en esta tarea. Los movimientos relacionados con la facturación, los cobros y pagos generan los correspondientes asientos contables de forma automática. [1]

Meribia Transporte

Es una aplicación orientada al sector del transporte de mercancías por carretera. Contempla todo el ciclo de gestión de este tipo de empresas, ya sean agencias (sin vehículos propios) o empresas de transportes (con vehículos propios).

Presenta como características para el servicio de facturación: [2]

- Permite realizar facturas de cualquier año en cualquier momento gracias al sistema de numeración basado en la serie y el año.
- Posibilidad de facturación en moneda extranjera.
- Facturación automática de las cargas pendientes de facturar a clientes y su posterior impresión.
- Posibilidad de especificar formas de pago: contado, aplazado, a la vista, etc. Tipos definibles por el usuario.
- Contabilización de la facturación.
- Control de los pagos a empresas de transportes cuando se trabaja como agencia de transporte y de los cobros a clientes.
- Definición flexible de formas de pago: contado, pagaré, 30 y 60 días. [2]

CIF-TRANS

Es la aplicación informática de gestión de cargas (cargas completas y cargas partidas). Está destinada a las empresas de transporte por carretera contemplando todo tipo de situaciones posibles. Cuya aplicación está desarrollada en entorno gráfico de Windows para su utilización tanto en mono puesto como en red.

Permite:

- Facturación masiva o selectiva por clientes.
- Facturación general o por tipos de mercancías, por pactos de facturación (horas, km, etc.)
- Emisión de recibos en soporte físico y magnético.
- Confección de remesas, y control de la cartera de efectos pagados descontados, en gestión de cobro y devueltos.

1.3.2 Sistemas nacionales

Casi todas las empresas de nuestro país trabajan con el sistema ASSETS, el cual solo tiene para llevar un control de cifras de pago, es decir, después de efectuado el pago de dietas los departamentos de economía y finanza de las empresas archivan el concepto de gasto: las cifras pagadas en cada dieta o pago menor. Este sistema no registra a la persona que se le ha realizado el pago.

ASSETS, es un sistema flexible, con ayuda en línea, tiene pantallas de entradas de datos con opciones fáciles de interpretar y ejecutar, facilita el uso de la parametrización para adaptarse a las exigencias de cada cliente que lo utilice, con la emisión de varios reportes que tendrán la forma y el contenido que el usuario defina. También facilita la ejecución de auditorías contables para localizar errores de compatibilidad de datos, así como eliminar estas incorrecciones, garantizando la fidelidad de su información. En este sistema siempre se registra por parte de los clientes que lo trabajan las cifras que se presenten ante cualquier tipo de procedimiento así como el concepto por el que se utiliza la cifra, nunca se registra por concepto de dieta al proveedor que la solicita siendo así diferente en facturas que si se registra el cliente que solicita el pago del servicio prestado pero no los datos económicos de las empresas a las que se les va a pagar un servicio.

1.4. Tendencias actuales de la tecnología en el desarrollo de software

Al analizar todos los problemas existentes en el DTN y estudiar cómo debe fluir la información, se comienza un estudio de las tendencias actuales del desarrollo de software para definir las herramientas y tecnologías a utilizar. Después de analizar una amplia información sobre el tema y estudiar la propuesta de las herramientas plasmadas en el diseño de sistema que se va a implementar, se define la realización de una aplicación Web. A continuación se muestra un conjunto de **ventajas** de este tipo de aplicaciones: [3]

- No requieren instalación, pues usan tecnología Web, lo cual permite el aprovechamiento de todas las características del Internet.
- Son fáciles de usar ya que no se requieren complicadas combinaciones de Hardware/Software para utilizar estas aplicaciones. Solo un computador con un buen navegador web.
- Alta disponibilidad, ya que puede realizar consultas en cualquier parte del mundo donde tenga acceso a Internet y a cualquier hora.
- Una empresa puede migrar de sistema operativo o cambiar el Hardware libremente sin afectar el funcionamiento de las aplicaciones de servidor.
- Se facilita el trabajo a distancia.
- Actualizar o hacer cambios en el Software es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor lo que implica que no hay que distribuirla entre los demás computadores. El proceso es rápido y limpio.

Estos tipos de aplicaciones presentan **desventajas** como:

- Acceso limitado, la necesidad de conexión permanente y rápida a Internet hacen que el acceso a estas aplicaciones no esté al alcance de todos.
- La interactividad no se produce en tiempo real, en las aplicaciones web cada acción del usuario conlleva un tiempo de espera excesivo hasta que se obtiene la reacción del sistema.
- Elementos de interacción muy limitados. En comparación con el software de escritorio, las posibilidades de interacción con el usuario que ofrecen las aplicaciones web (mediante formularios principalmente) son muy escasas.
- Diferencias de presentación entre plataformas y navegadores. La falta de estándares ampliamente soportados dificulta el desarrollo de las aplicaciones.[3]

1.4.1 Lenguajes de Programación

1.4.1.1 PHP

[4] Es un lenguaje de script interpretado en el lado del servidor, utilizado para la generación de páginas Web dinámicas, embebido en páginas HTML y ejecutado en el servidor. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl, con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como es HTML, XML o WML [4]. PHP es un lenguaje que se ejecuta en el servidor por lo que no es necesario que su navegador lo soporte, es independiente del navegador. Al ser un lenguaje libre es una alternativa de fácil acceso para todos y presenta características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas:

- El precio para utilizar PHP es cero, por lo que es gratuito y se puede descargar desde www.php.net.
- [5] Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MSSQL Server, Sybase mSQL, Informix.
- El código se pone al día continuamente con mejoras y extensiones del lenguaje, con el objetivo de ampliar las capacidades de PHP.
- Se puede hacer cualquier cosa que podemos realizar con un *script CGI*, como el procesamiento de información en formularios, foros de discusión, manipulación de *cookies* y páginas dinámicas.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas de la web de fácil programación.
- Más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.

- Goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.[5]
- Es multiplataforma. Funciona en toda máquina que sea capaz de compilar su código, entre ellas diversos sistemas operativos para PC y diversos Unix. El código escrito en PHP en cualquier plataforma funciona exactamente igual en otra.

[6] En cuanto a seguridad es un potente lenguaje y el intérprete, tanto incluido en el servidor Web, como módulo o ejecutado como un binario CGI, puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor. Todas estas características hacen que lo que se ejecute en el servidor Web sea seguro por defecto. PHP ha sido diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI, Perl o C y con la correcta selección de las opciones de configuración de tiempo de compilación y ejecución se consigue la exacta combinación de libertad y seguridad que se necesita. Ya que existen diferentes modos de utilizar PHP, existe también una multitud de opciones de configuración que permiten controlar su funcionamiento. Una gran selección de opciones garantiza que se pueda usar PHP para diferentes aplicaciones, pero también significa que existen combinaciones de estas opciones y configuraciones del servidor que producen instalaciones inseguras. [6]

1.4.1.2 Javascript

Es un lenguaje de scripts del lado del cliente desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML además porque es el navegador el que soporta la carga de procesamiento. Fundamentalmente se utiliza para realizar validaciones de datos del lado del cliente. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Con Javascript se pueden crear efectos especiales sobre páginas web, como crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo, además se pueden definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

Javascript es el siguiente paso, después del HTML, que puede dar un programador de la web que decida mejorar sus páginas y la potencia de sus proyectos. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.

[7]La característica principal de Javascript, de hecho, es la de ser un lenguaje de *scripting*, pero, sobre todo, la de ser el lenguaje de scripting por excelencia y, sin lugar a dudas, el más usado [7].

1.4.2 Tecnología para el desarrollo web

1.4.2.1 Ajax

Acrónimo de Asynchronous JavaScript And XML (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente (en el navegador del usuario) y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad. AJAX propone un nuevo modelo de interacción Web combinando las tecnologías siguientes: [8]

- Document Object Model (DOM) para visualizar dinámicamente e interactuar con la información presentada.
- XML, XSLT para intercambiar y manipular datos.
- CSS para definir el aspecto del documento.
- JSON y JSON-RPC pueden ser alternativas a XML/XSLT
- XMLHttpRequest para recuperar datos de forma asincrónica.
- Javascript como nexo de unión de todas estas tecnologías. [8]

Ajax no es una tecnología en sí, sino que es un término que engloba a un grupo de tecnologías que trabajan conjuntamente. Las aplicaciones que utilizan esta técnica son más interactivas, ya que como se dijo anteriormente, responden a las interacciones del usuario más rápidamente, al estilo aplicaciones de escritorio.

Es usado para multitud de tareas como actualizar o eliminar registros, expandir formularios web, devolver peticiones simples de búsqueda; todo sin tener la necesidad de tener que recargar toda la página de HTML cada vez que se realiza un cambio. Generalmente sólo requiere enviar pequeñas peticiones al servidor, y se devuelven respuestas relativamente cortas.

Los navegadores que permiten Ajax son Microsoft Internet Explorer para Windows versión 5.0 y superiores. Navegadores basados en Gecko como Mozilla, Mozilla Firefox, SeaMonkey, Camino, Flock, Epiphany, Galeon y Netscape versión 7.1 y superiores. Navegadores con el API KHTML versión 3.2 y superiores implementado, incluyendo Konqueror versión 3.2 y superiores, y el Web Browser para S60 de Nokia tercera generación y posteriores. Opera versión 8.0 y superiores, incluyendo Opera Mobile Browser versión 8.0 y superiores.

[9] Los navegadores que no permiten Ajax son Opera 7 y anteriores. Microsoft Internet Explorer para Windows versión 4.0 y anteriores. Microsoft Internet Explorer para Macintosh (todas las versiones). Navegadores basados en texto como Lynx y Links. Navegadores para incapacitados visuales (braille) [9].

1.4.3 Servidor de Aplicaciones

1.4.3.1 Apache

Apache es un servidor web gratuito, potente y que ofrece un servicio estable y sencillo de mantener y configurar. Es indiscutiblemente uno de los mayores logros del Software Libre. El servidor Apache se desarrolla dentro del proyecto HTTP Server (http) de la Apache Software Foundation.

Se destacan las siguientes características:

- Es multiplataforma, aunque idealmente está preparado para funcionar bajo Linux.
- Muy sencillo de configurar.
- Es Open-Source.
- Muy útil para proveedores de Servicios de Internet que requieran miles de sitios pequeños con páginas estáticas.
- Amplias librerías de PHP y Perl a disposición de los programadores.
- Posee diversos módulos que permiten incorporarle nuevas funcionalidades, estos son muy simples de utilizar.
- Es capaz de utilizar lenguajes como PHP, TCL, Pitón, entre otros.

Apache presenta además entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido.

1.4.4 Gestor de Base de datos

Un sistema gestor de base de datos (SGBD) un conjunto de programas que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad.

1.4.4.1 PostgreSQL

PostgreSQL es un servidor de base de datos relacional orientado a objetos de software libre. Como muchos otros proyectos, es de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo.

Algunas de sus principales características son la alta concurrencia. Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras

un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último que se actualizó.

PostgreSQL presenta características como: [10]

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.[10]

1.4.5 Entorno de desarrollo integrado de programación

Un entorno de desarrollo integrado (IDE) es un programa compuesto por un conjunto de herramientas para un programador.

Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación.

1.4.5.1 Zend Studio

Sirve como editor de texto para páginas PHP y proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

Diferentes expertos consideran a Zend Studio como el entorno IDE más maduro y con más características útiles. Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP.

Todo el programa está escrito en Java, por lo que a veces no funciona tan rápido como otras aplicaciones de uso diario, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones

del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

[11] Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración [11].

1.4.6 Framework

Los frameworks simplifican el desarrollo de las aplicaciones mediante la automatización de muchas de las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al programador a crear código más legible y más fácil de mantener.

1.4.6.1 CodeIgniter

Es un framework de PHP basado en la arquitectura Modelo-Vista-Controlador (MVC), de la cual se abordará en el capítulo siguiente, está diseñado fundamentalmente para desarrolladores que necesiten crear aplicaciones web en poco tiempo y con un alto rendimiento. Es ligero y flexible. Puede ser tan sólo VC (Vista-Controlador) pues no fuerza al usuario a utilizar una base de datos para un desarrollo.

[12]CodeIgniter es un Framework para desarrollo de Aplicaciones- un *toolkit* para personas que quieran construir sitios usando PHP. Su objetivo es poder hacer los proyectos más rápidos de lo que usted puede hacerlos escribiendo código desde el principio, suministrando un rico conjunto de librerías para tareas comunes, con una simple interface y estructura lógica para acceder a estas librerías. CodeIgniter deja que su creatividad se centre en el proyecto, minimizando la cantidad de código necesitado para ejecutar una tarea [12].

Entre sus características, se encuentra la compatibilidad con PHP 4 y 5, posee soporte para múltiples bases de datos entre las que se pueden mencionar PostgreSQL, MySQL, MSSQL, además de que se lo pueden incluir drivers para otros gestores de bases de datos que usted programe, incluye también plantillas, validaciones y no requiere instalación.

Una de las características más interesantes de CodeIgniter es el elevado número de clases que incluye para trabajar con distintos objetos: calendario, bases de datos, correo electrónico, manipulación de imágenes, FTP, lenguaje, tablas, sesiones, compresión ZIP, entre otros. A diferencia de otros frameworks, CodeIgniter cuenta con una documentación excelente que permite conocer todos los

secretos de este entorno de trabajo. Está liberado bajo la licencia de código abierto Apache-BSD, lo que significa que es totalmente libre y puede ser usado.

1.5. Propuesta de solución

La aplicación que se desarrollará facilitará al Departamento de Transportaciones Nacionales UCI un control de todos los procesos de pago que se realicen, desglosando sus funcionalidades en cuatro módulos fundamentales que son: Administración, Dieta, Factura y Reportes.

El mismo debe permitir a los administradores que lo trabajen, llevar un mejor control de pago de las dietas que se realicen de acuerdo al tipo de concepto por el que se realizan y que permita obtener cifras y datos estadísticos de las personas que efectúan la solicitud de pago por concepto de dieta. Lo mismo debe suceder con el pago de factura, que se tenga un mejor control de las facturas que se pagan para que a la Dirección de Economía le sea accesible tener información para todas las tareas que desarrollan en cuanto a pagos.

Para la elaboración de la propuesta de solución se partió de las herramientas y tecnologías propuestas en el diseño de la aplicación por lo que se plantea desarrollar una aplicación Web la cual permitirá que el sistema pueda ser utilizado desde distintos lugares y sin más requerimientos que una computadora con navegador Web y conexión a la red, por lo que fue diseñado con una arquitectura Cliente/Servidor, de la cual se darán detalles posteriormente. Se usará PHP como lenguaje de programación del lado del servidor, como Sistema Gestor de Base de Datos Postgres y Apache como servidor de aplicación Web, todo esto por las potencialidades que ofrecen, así como por formar parte del grupo de software de código abierto, solución por la que se aboga en la UCI. Se utilizará Javascript del lado del cliente para lograr la interactividad con el usuario en el navegador y Ajax como tecnología de desarrollo web, ya que la misma permite actualizar parte de una página en cualquier momento, dándoles a los usuarios una respuesta instantánea a sus ingresos y consultas. Además se utilizará como IDE de desarrollo Zend Studio y como framework el CodeIgniter.

1.6. Conclusiones

En este capítulo se profundizó en el conocimiento de algunos conceptos necesarios para la comprensión de este trabajo. En la elaboración del capítulo se esboza el uso de las tecnologías y herramientas usadas en la confección del trabajo y un marco para poder guiarse mientras se trabajaba en el presente proyecto. Es de mucha utilidad el conocer las ventajas que traería el uso de las antes mencionadas para la elaboración del trabajo.

CAPÍTULO 2: “DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA”

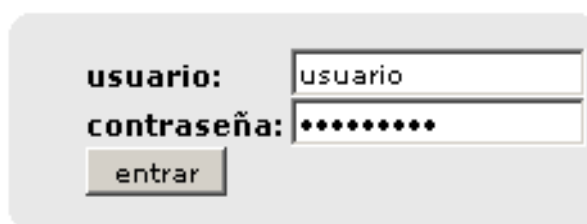
2.1 Introducción

Una vez analizado y estudiado toda la documentación referente al framework de desarrollo que utilizamos para la implementación y todo lo referente a los procesos de pago de dietas y facturas, entonces se abordará en este capítulo detalles de la solución propuesta. Se describirán las funcionalidades, clases u operaciones necesarias, así como el patrón o estilo arquitectónico presente en el sistema, se realizará una valoración crítica del diseño ya propuesto y un análisis de posibles implementaciones y componentes o módulos que son rehusados. Además en el siguiente capítulo se generan los artefactos que propone el RUP para el flujo de trabajo de implementación.

2.2 Descripción de la solución propuesta

2.2.1 Autenticar usuario

El usuario introduce sus datos en el formulario que se muestra en la figura 1, el mismo envía los datos introducidos, se verifican si son los correctos, luego se buscan los datos del usuario en la base de datos, y se le asigna el nivel de acceso dentro del sistema.



El formulario de autenticación de usuario se muestra en un recuadro gris con los siguientes elementos:

- Etiqueta **usuario:** con un campo de texto que contiene el texto "usuario".
- Etiqueta **contraseña:** con un campo de texto que contiene ocho caracteres de puntos, representando una contraseña oculta.
- Botón **entrar** situado debajo de los campos de texto.

Figura 1 Autenticar usuario.

2.2.2 Gestionar dieta

Cuando se selecciona en el menú el vínculo Módulo Dieta, se le mostrará al usuario la opción de registrar una dieta y en un formulario se le mostrará las dietas existentes hasta el momento como se muestra en la figura 2, dándole la posibilidad de que puedan obtener una dieta determinada mediante el buscador para luego modificarla y eliminarla como se muestra en la figura 3. Si se va a insertar una dieta se deben de llenar los campos correspondientes como se muestra en la figura 4 y se debe de especificar el tipo de dieta, es decir, si es Dieta por liquidación o Dieta por anticipo, debido a que cada tipo de dieta tienen algunas características diferentes, después se envían los datos, se verifican si los mismos son correctos y se almacena la información en la base de datos. Si se desea modificar los datos de una dieta determinada, se selecciona mediante la selección descrita anteriormente, se actualizan los datos correspondientes como se muestra en la figura 5, y se realiza el mismo proceso de envío y verificación expuesto anteriormente. A la hora de eliminar una dieta determinada, se

Sistema de Control de Pago de Dietas y Facturas

selecciona, se muestra la información que se desea eliminar, el sistema mostrará un mensaje si desea eliminar, como se muestra en la figura 6, de acuerdo a la respuesta del cliente se borran sus datos o no.

Panel de búsqueda

Cadena de búsqueda

Resultados de la búsqueda



















Solicitante	Fecha	Dirección			
Rosmel Perez	2009-06-02	Facultad 3			
Radel Calzada	2009-05-13	Direccion de Informatizacion			
Julio Cesar	2009-05-21	UCIFAR			
Radel Calzada	2009-05-16	Informatizacion			
alberto tamayo	2009-03-25	ip			

Figura 2 Lista de las dietas existentes hasta el momento.

Panel de búsqueda

Cadena de búsqueda

Resultados de la búsqueda







Solicitante	Fecha	Dirección			
Rosmel Perez	2009-06-02	Facultad 3			

Figura 3 Solicitud para obtener una dieta.

Dirección del solicitante

Nombre del solicitante

Fecha de confección del modelo

Motivos del viaje

Fecha de salida estimada

Fecha de salida real

Fecha de regreso real

Días de viaje

Estimado

Real

Nro. de solicitud

Tipo de dieta




- Anticipo
- Liquidación

Figura 4 Insertar una dieta según el tipo de dieta deseado (Anticipo o Liquidación).

Panel de búsqueda

Cadena de búsqueda

Resultados de la búsqueda

Solicitante	Fecha	Dirección	
Rosmel Perez	2009-06-02	Facultad 3	  

Direccion del solicitante
 Nombre del solicitante
 Fecha de confección del modelo
 Motivos del viaje
 Fecha de salida estimada
 Fecha de salida real
 Fecha de regreso real
 Días de viaje
 Estimado
 Real
 Recibido
 Liquidado
 Custodio
 Nro. de solicitud
 Numero cajera
 Tipo de dieta

	Almuerzo	Hospedaje	Desayuno	Transporte	Total
Entregado	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text"/>
Utilizado	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text"/>
Devuelto	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text"/>
A entregar	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text" value="12"/>	<input type="text"/>

Figura 5 Modificar una dieta determinada.

Panel de búsqueda

Cadena de búsqueda

Resultados de la búsqueda

Solicitante	Fecha	Dirección	
Rosmel Perez	2009-06-02	Facultad 3	

Figura 6 Eliminar una dieta determinada.

2.2.3 Gestionar factura

Cuando se selecciona en el menú el vínculo Módulo Factura, se le mostrará al usuario la opción de registrar una factura y en un formulario se le mostrará las facturas existentes hasta el momento como se muestra en la figura 7, dándole la posibilidad de que puedan obtener una factura determinada mediante el buscador para luego modificarla y eliminarla como se muestra en la figura 8. Si se va a insertar una factura se deben de llenar los campos correspondientes como se muestra en la figura 9 y se debe de especificar la forma de pago, es decir, si es por transferencia bancaria o por cheque, debido a que cada forma de pago agrega nuevos datos a guardar, después se envían los datos, se verifican si los mismos son correctos y se almacena la información en la base de datos. Si se desea modificar los datos de una factura determinada, se selecciona mediante la selección descrita anteriormente, se actualizan los datos correspondientes como se muestra en la figura 10, y se realiza el mismo proceso de envío y verificación antes expuesto. A la hora de eliminar una factura determinada, se selecciona mediante una búsqueda la factura que se desea eliminar, se muestra la información, el sistema mostrará un mensaje si desea eliminar, como se muestra en la figura 11, de acuerdo a la respuesta del cliente se borran sus datos o no.

Panel de búsqueda

Criterio de búsqueda

Resultados de la búsqueda

A favor de	Fecha	Importe	
Yudisel	2009-12-31	21	<input type="button" value="editar"/> <input type="button" value="eliminar"/> <input type="button" value="detalle"/>
yu	2009-06-02	12	<input type="button" value="editar"/> <input type="button" value="eliminar"/> <input type="button" value="detalle"/>

Figura 7 Lista de las facturas existentes hasta el momento.

Panel de búsqueda

Criterio de búsqueda

Resultados de la búsqueda

A favor de	Fecha	Importe	
Yudisel	2009-12-31	21	<input type="button" value="editar"/> <input type="button" value="eliminar"/> <input type="button" value="detalle"/>

Figura 8 Solicitud para obtener una factura determinada.

Área solicitante	<input type="text"/>
Fecha confección	<input type="text"/>
Tipo de solicitud	<input type="text"/>
Tipo de pago	<input type="text"/>
Tipo de moneda	<input type="text"/>
Forma de pago	<input type="text" value="Cheque"/>
Plan aprobado	<input type="text"/>
Plan disponible	<input type="text"/>
Importe	<input type="text"/>
Disponibilidad	<input type="text"/>
A favor de	<input type="text"/>
Fundamentación de solicitud	<input type="text"/>
Número de cheque	<input type="text"/>
Fecha	<input type="text"/>
	<input type="button" value="registrar"/>

Figura 9 Insertar factura especificando el tipo de pago (Cheque).

Área solicitante	<input type="text" value="876543"/>
Fecha confección	<input type="text" value="31/12/2009"/>
Tipo de solicitud	<input type="text" value="Compras"/>
Tipo de pago	<input type="text" value="Servicio"/>
Tipo de moneda	<input type="text" value="CUC"/>
Forma de pago	<input type="text" value="Cheque"/>
Plan aprobado	<input type="text" value="123"/>
Plan disponible	<input type="text" value="321"/>
Importe	<input type="text" value="21"/>
Disponibilidad	<input type="text" value="32"/>
A favor de	<input type="text" value="Yudisel"/>
Fundamentación de solicitud	<input type="text" value="El pasado mes de abril, estudiantes de la facultad 6 del grupo 6304 visitaron la pizzería de la universidad para satisfacer algunas inquietudes respecto al proceso de elaboración de"/>
Número de cheque	<input type="text" value="342534657"/>
Fecha	<input type="text" value="27/12/2009"/>
<input type="button" value="guardar cambios"/>	

Figura 10 Modificar una dieta existente.

Panel de búsqueda	
Criterio de búsqueda	<input type="text" value="yudisel"/>
<input type="button" value="buscar"/>	
Resultados de la búsqueda	
A favor de	Fecha Importe
Yudisel	2009-12-31 21

Figura 11 Eliminar una dieta existente.

2.2.4 Visualizar resumen

Cuando se selecciona en el menú el Módulo de reportes, se le mostrará al usuario la opción de obtener el resumen y en un formulario se le mostrará el rango de fecha a establecer, luego de enviada esta información se le mostrará todas las facturas y las dietas por concepto de bonificación, liquidación y

anticipo existentes hasta el momento como se muestra en la figura 12. Se le posibilitará la opción de imprimir el resultado obtenido.

Fecha inicio Fecha fin

Dietas

Solicitante	Fecha	Dirección
Rosmel Perez	2009-06-02	Facultad 3
Radel Calzada	2009-05-13	Direccion de Informatizacion
Julio Cesar	2009-05-21	UCIFAR
Radel Calzada	2009-05-16	Informatizacion

Bonificaciones

Dirección	Nombre solicitante	Codigo
Facultad 3	Mauris Yadira	56789

Facturas

A favor de	Fecha	Importe
yu	2009-06-02	12

Figura 12 Obtener resumen de las dietas y facturas existentes hasta el momento.

2.2.5 Obtener modelos

Para obtener los modelos de una dieta o de una factura, el usuario trabajará en el módulo que desee, ya sea en el de dieta o el de factura, luego de seleccionado el módulo, procederá mediante el proceso de búsqueda descrito en las funcionalidades 2.2.2 y 2.2.3, luego de obtener la dieta o la factura, se procederá a seleccionar la opción de visualizar el modelo, como se muestra en la figura 13. Luego podrá ver todos los datos de la selección en su modelo correspondiente como se muestra en la figura 14 se le posibilitará la opción de imprimir el resultado obtenido.

Panel de búsqueda

Cadena de búsqueda

Resultados de la búsqueda

Solicitante	Fecha	Dirección
Julio Cesar	2009-05-21	UCIFAR

Figura 13 Visualizar modelo.

UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS		ANTICIPO Y LIQUIDACION DE GASTOS DE VIAJES			MODELO	
					1.03	
Dirección:UCIFAR		Código:		D	M	A
Nombre y Apellidos: Julio Cesar		21	05	2009		
Motivo del Viaje:		Fecha		D	M	A
Este es un viaje		Salida Estimada		22	05	2009
		Salida		24	05	2009
		Regreso		28	05	2009
		Días de viaje 5				
AUTORIZADO		Estimado 5				
Entrega	Liquidación		D	M	A	Real
			12	05	2009	Hospedado
Recibo	D	M	A	Concepto	Total	Alimentación
	12	05	2009	Entregado	0	0
Liquidado	D	M	A	Utilizado	20	5
	12	05	2009	Devuelto	0	0
Custodio	D	M	A	A Entregar	20	5
	12	05	2009	Solicitud No.	Anotado	
				5	Número	
					87654	

Figura 14 Imprimir el modelo resultante.

2.2.6 Obtener reportes

Para la búsqueda de todas las dietas y facturas existentes por criterios de área, fecha de ingreso y fecha de pago, el usuario la podrá realizar mediante el buscador implantado en cada módulo y obtendrá todas las dietas y facturas registradas, como se muestra en la figura 15. En caso de querer el

reporte por criterio de saldo, el usuario de remitirá al módulo de reportes y seleccionará la opción de visualizar reporte, en un formulario se le mostrará el rango de fecha a establecer y el valor del saldo, luego de enviada esta información se le mostrará todas las facturas y las dietas existentes con este saldo, como se muestra en la figura 16.



Panel de búsqueda

Cadena de búsqueda

Figura 15 Buscar todas las dietas y facturas existentes por criterios de área, fecha de ingreso y fecha de pago.



Fecha inicio Fecha fin

Saldo

Dietas

No hay resultados

Bonificaciones

No hay resultados

Facturas

A favor de	Fecha	Importe
yu	2009-06-02	12

Figura 16 Dietas y facturas en un rango de fecha determinado con el saldo especificado.

2.2.7 Obtener cantidad de dietas y facturas mensuales

Para la búsqueda de todas las dietas y facturas existentes en un mes determinado, el usuario debe recurrir al módulo en el que desea trabajar, es decir si es factura o dieta, posteriormente en el buscador especificará el año y el mes, como se muestra en la figura 17, después de procesada la información, se le mostrará al usuario todas dietas o facturas registradas hasta el momento, como se muestra en la figura 18.

Panel de búsqueda

Cadena de búsqueda

Figura 17 Dietas o facturas correspondientes al mes y año especificado.

Panel de búsqueda

Cadena de búsqueda

Resultados de la búsqueda




Solicitante	Fecha	Dirección	
Rosmel Perez	2009-06-02	Facultad 3	  

Figura 18 Dieta o factura registrada en el mes y año especificado.

2.2.8 Gestionar dieta por bonificación

Cuando se selecciona en el menú el vínculo Módulo Bonificación, se le mostrará al usuario la opción de registrar una dieta por concepto de bonificación y en un formulario se le mostrará las dietas existentes hasta el momento como se muestra en la figura 19, dándole la posibilidad de que puedan obtener una dieta determinada mediante el buscador para luego modificarla y eliminarla. Si se va a insertar una dieta se deben de llenar los campos correspondientes. Después se envían los datos, se verifican si los mismos son correctos y se almacena la información en la base de datos. Si se desea modificar los datos de una dieta determinada, se selecciona mediante la selección descrita anteriormente, es decir, mediante el buscador, se actualizan los datos correspondientes y se realiza el mismo proceso de envío y verificación expuesto anteriormente. A la hora de eliminar una dieta determinada, se selecciona y se muestra la información que se desea eliminar, el sistema mostrará un mensaje si desea eliminar, de acuerdo a la respuesta del cliente se borran sus datos o no.

Panel de búsqueda

Criterio de búsqueda

buscar

Resultados de la búsqueda




Dirección	Nombre solicitante	Código	
Facultad 3	Mauris Yadira	56789	  

Figura 19 Dietas existentes hasta el momento.

2.3 Valoración crítica del diseño propuesto por el analista

El diseñador del sistema ha realizado una propuesta teniendo en cuenta el sistema que se quiere desarrollar. Primero en el momento de implementar el sistema y luego a la hora de ponerlo en funcionamiento.

En el diseño propuesto, para lograr una mejor comprensión de la lógica de los elementos del diseño, se propone una estructuración a través de paquetes del diseño, que contienen de manera lógica, las diferentes clases del negocio según su responsabilidad en la aplicación. También se cuenta con una serie de diagramas de interacción, que explican la relación entre las clases y como son llamados los métodos y sentencias dentro de cada una de ellas, generando así toda la información necesaria para conocer el orden de implementación de las acciones.

A medida que se fue desarrollando el software se encontraron detalles en las clases que se propusieron en el diseño, en cuanto a los atributos de las mismas, debido a que hay datos que ya no son necesarios guardar en el sistema, por lo que esto conlleva a una nueva modelación de la base de datos.

2.3.1 Modelo de la BD

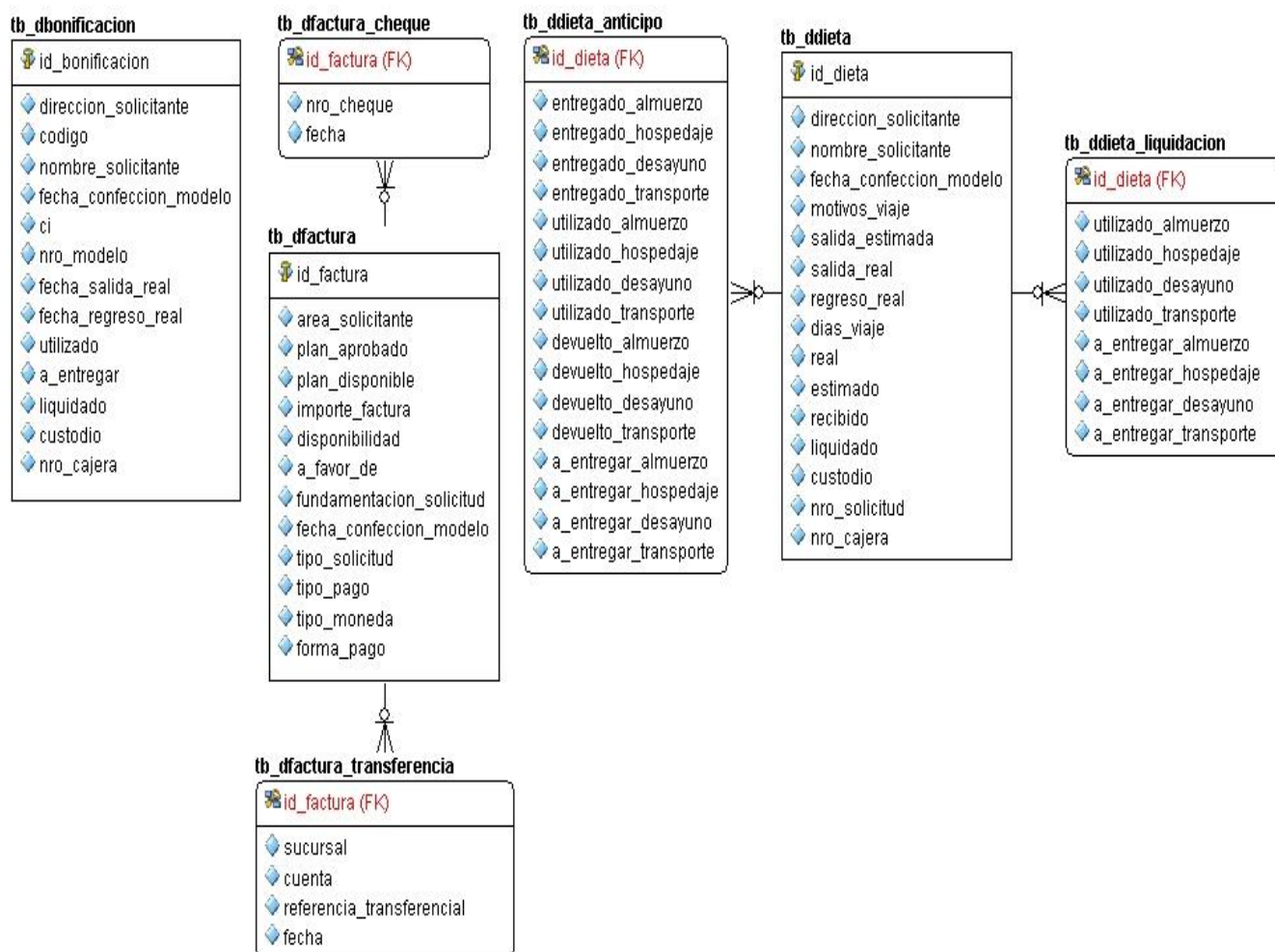


Figura 20 Modelo de la Base de Datos.

2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, estas pueden ser características que hagan al producto atractivo, usable, rápido o confiable. Especifican propiedades del sistema como restricciones de ambiente y desarrollo, performance, dependencias de plataformas, y confiabilidad.

A continuación se mostrarán los requisitos no funcionales del sistema que el analista propuso:

Apariencia o interfaz externa

El producto final debe tener una interfaz fácil de usar y amigable con un ambiente acorde a los principios de trabajo de la Dirección de Economía y el Departamento de Transportaciones Nacionales.

Estará diseñado para la resolución deseada por el usuario, aunque debe de soportar un estándar de 800 x 600 píxeles.

Debe tener imágenes acordes a las funciones que ejerce el Departamento de Transportaciones Nacionales.

Usabilidad

- El sistema ha de ser de una sencillez tal que pueda ser usado por personas que tengan un conocimiento básico en el manejo de las computadoras.
- El sistema deberá estar disponible las 24 h del día.
- El sistema deberá contar con Menús con las funciones más importantes del sistema.

Rendimiento

- Las funcionalidades deben de estar divididas en secciones, de modo que no se sobrecarguen los pedidos.
- Se debe soportar el paginado cuando sea mucha la densidad de una sección.
- Las respuestas no deben tardar en ser procesadas más de 3 segundos.
- El hardware donde corra la aplicación debe tener suficiente memoria RAM para soportar más de 100 peticiones simultáneas.
- Se necesita un servidor de bases de datos que soporte grandes volúmenes de datos.

Soporte

- Se tendrá una documentación adecuada que permita un entendimiento del funcionamiento del software.

Políticos culturales

- El producto no debe contener palabras en otros Idiomas.
- El producto debe respetar los términos empleados normalmente por los especialistas en el tema de las organizaciones que represente.
- Debe contener información acorde a los principios éticos puestos en vigor en la Dirección de Economía y el Departamento de Transportaciones Nacionales.

Portabilidad

- El sistema será multiplataforma (Linux o Windows).

Seguridad

- El usuario debe autenticarse antes de entrar al sistema.

Confiabilidad

- Deben establecerse los mecanismos necesarios para el restablecimiento del sistema ante fallos de comunicación u otros, los tiempos mínimos para ellos no deben exceder de 10 minutos.
- Deben montarse sistemas de respaldo eléctrico en los locales de los servidores para mantener la vitalidad de los servicios.
- Deben hacerse una copia semanal de los datos hacia una zona segura, para garantizar que no se pierdan.

2.4 Patrones o estilos arquitectónico presente en la solución propuesta.

2.4.1 Patrón Modelo Vista Controlador (MVC)

Para la implementación de la aplicación como se explicó anteriormente se utilizó como framework el codeigniter, el cual está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por 3 niveles: Modelo-Vista-Controlador. MVC es una aproximación al software que separa la lógica de la aplicación de la presentación. En la práctica, permite que sus páginas web contengan mínima codificación ya que la presentación es separada del código PHP.

El **Modelo** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. Accede a la capa de almacenamiento de datos o acceso a datos. Especialmente sus clases de modelo contendrán funciones que lo ayudarán a recuperar, insertar y actualizar información en su base de datos.

La **Vista** transforma el modelo en una página web que permite al usuario interactuar con ella, es decir que no es más que la información que es presentada al usuario. La Vista comúnmente será una página web, pero en CodeIgniter, una vista también puede ser un fragmento de una página como un encabezado o un pie de página. También puede ser una página RSS, o cualquier otro tipo de "página".

El **Controlador** sirve como un intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para procesar la petición HTTP y generar una página web.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. CodeIgniter tiene un enfoque bastante flexible del MVC, ya que los Modelos no son requeridos. Si no necesita agregar separación, o descubre que mantener los modelos requiera más complejidad de la que quería, puede ignorarlos y

construir su aplicación mínimamente usando Controladores y Vista. CodeIgniter también le permite incorporar sus códigos existentes, o incluso desarrollar librerías de núcleo para el sistema, habilitándolo a trabajar en una forma que hace que tenga más sentido para usted.

2.4.2 Arquitectura Cliente/Servidor

Como se expuso anteriormente se utilizará una arquitectura Cliente/Servidor en el sistema a desarrollar y que posteriormente se observará gráficamente en el modelo de despliegue, con el objetivo de que todas las informaciones que sean procesadas se puedan dividir en procesos independientes que cooperen entre sí para poder intercambiar informaciones, servicios o recursos. Este modelo permitiría a la aplicación que se pueda dividir de forma que el servidor contenga la parte que debe ser compartida por varios usuarios, y en el cliente permanezca sólo lo particular de cada usuario.

En la parte cliente se realizan funciones como:

- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.
- Manejo de la interfaz de usuario.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Enlaces de comunicaciones con otras redes de áreas local o externa.
- Gestión de periféricos compartidos.
- Control de acceso concurrente a bases de datos compartidas.

Cada vez que un cliente necesite un servicio lo solicita al servidor correspondiente y éste le responderá proporcionándole lo que solicita. Pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Las principales características de la arquitectura cliente/servidor se pueden destacar las siguientes:

- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

2.5 Modelo de Implementación

2.5.1 Modelo de despliegue

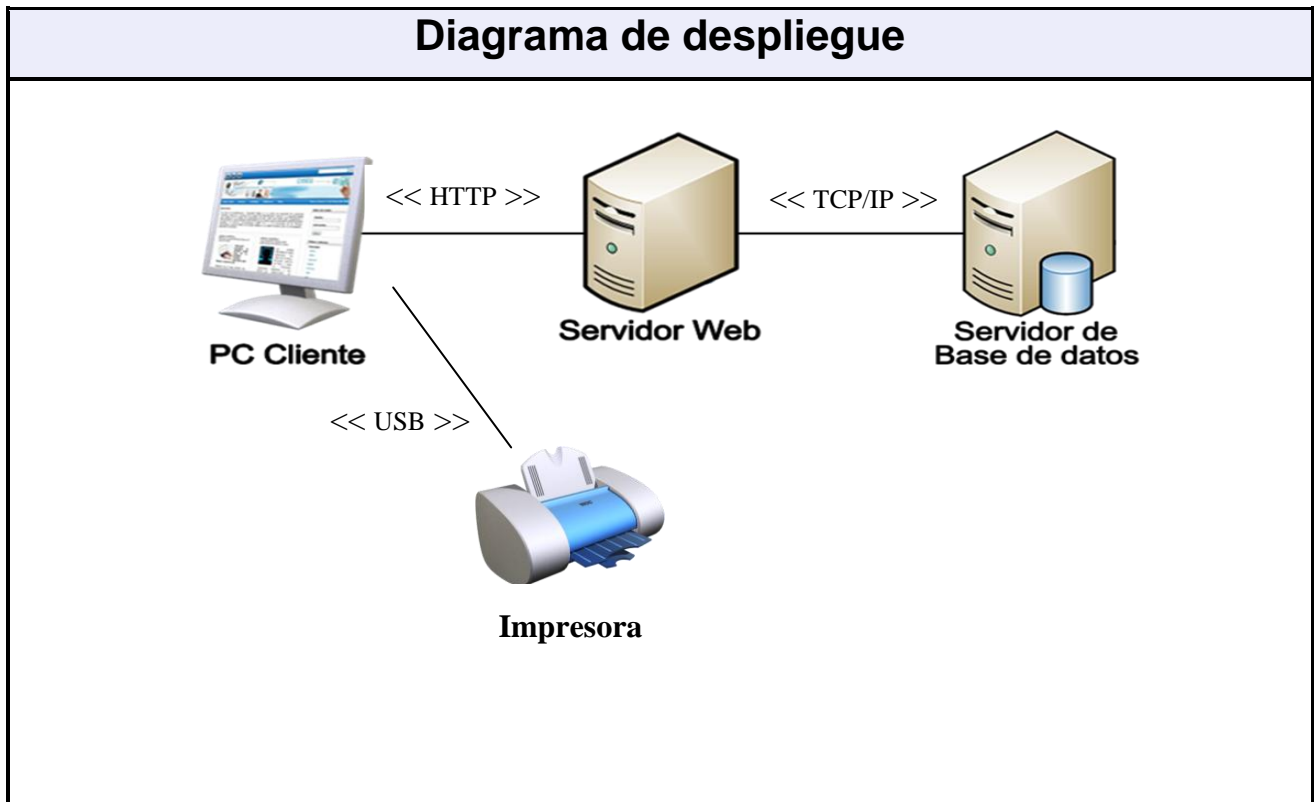


Figura 20 Despliegue del sistema.

2.5.2 Diagrama de componentes

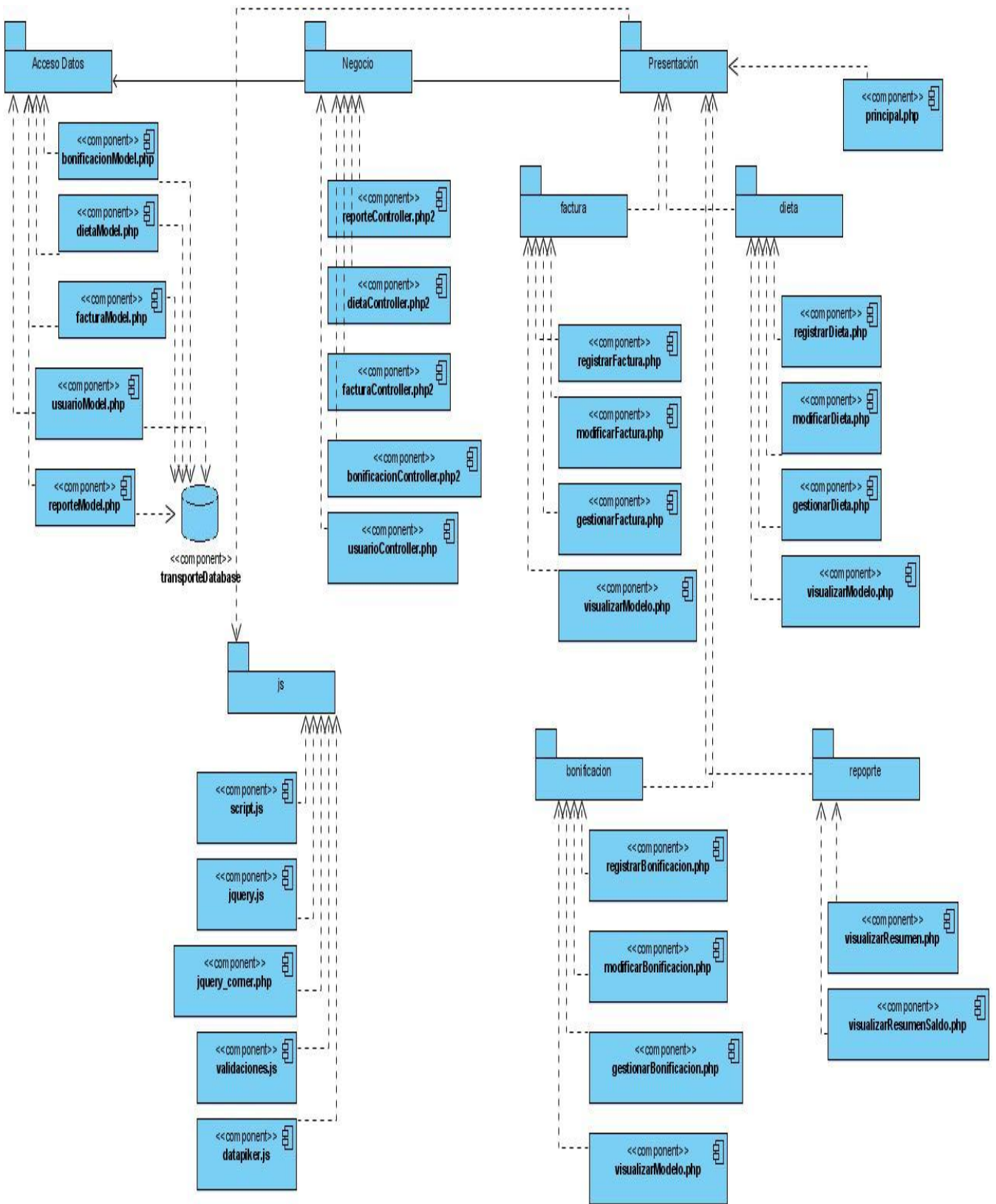


Figura 21 Diagrama de Componentes (Global).

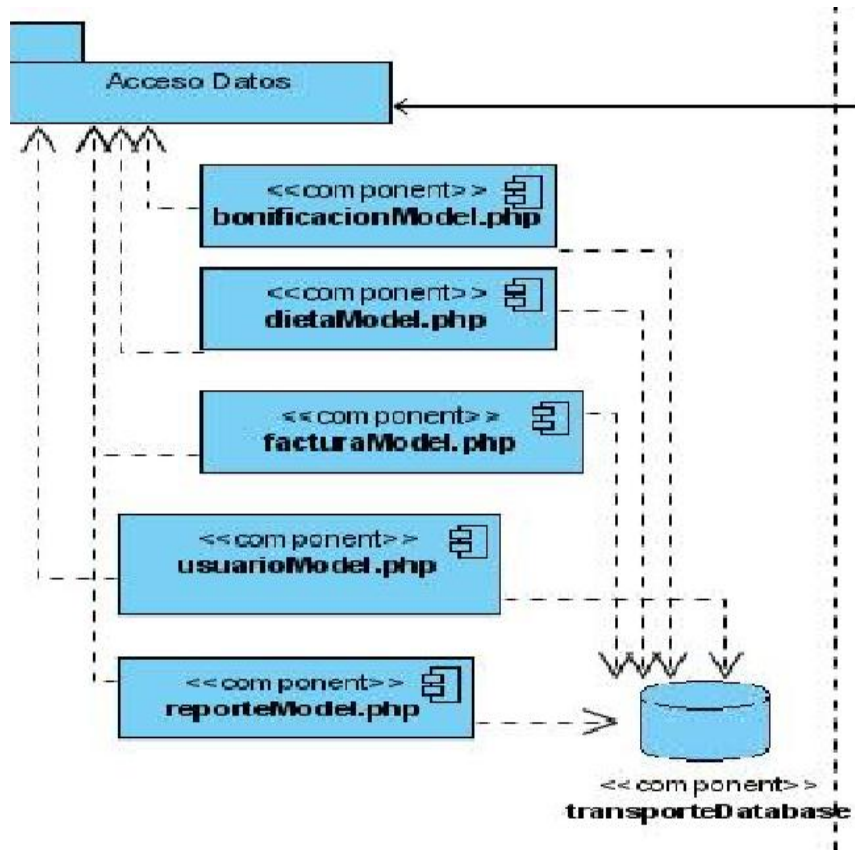


Figura 22 Diagrama Componentes (Parte de Acceso a Datos).

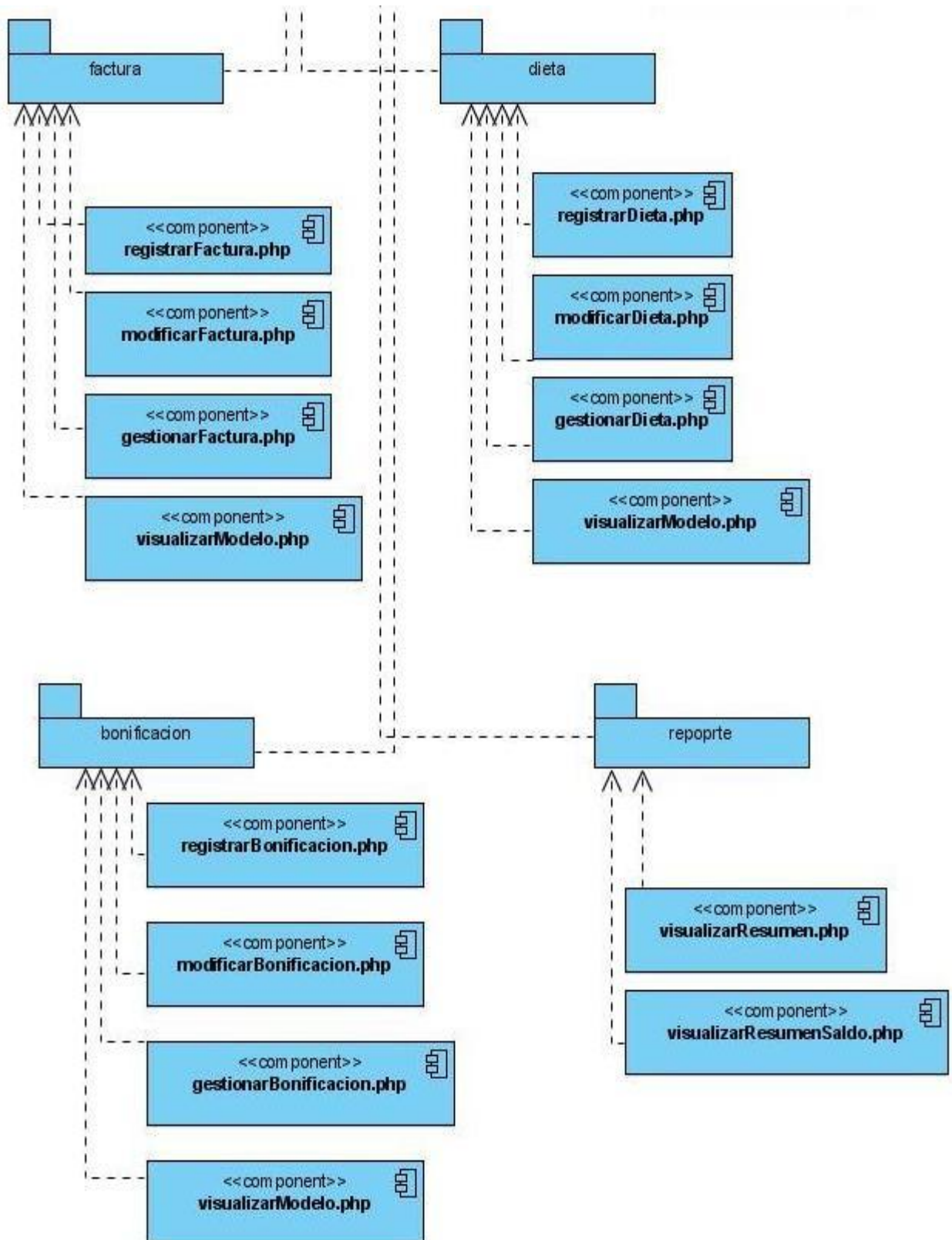


Figura 23 Diagrama de Componentes (Vistas).

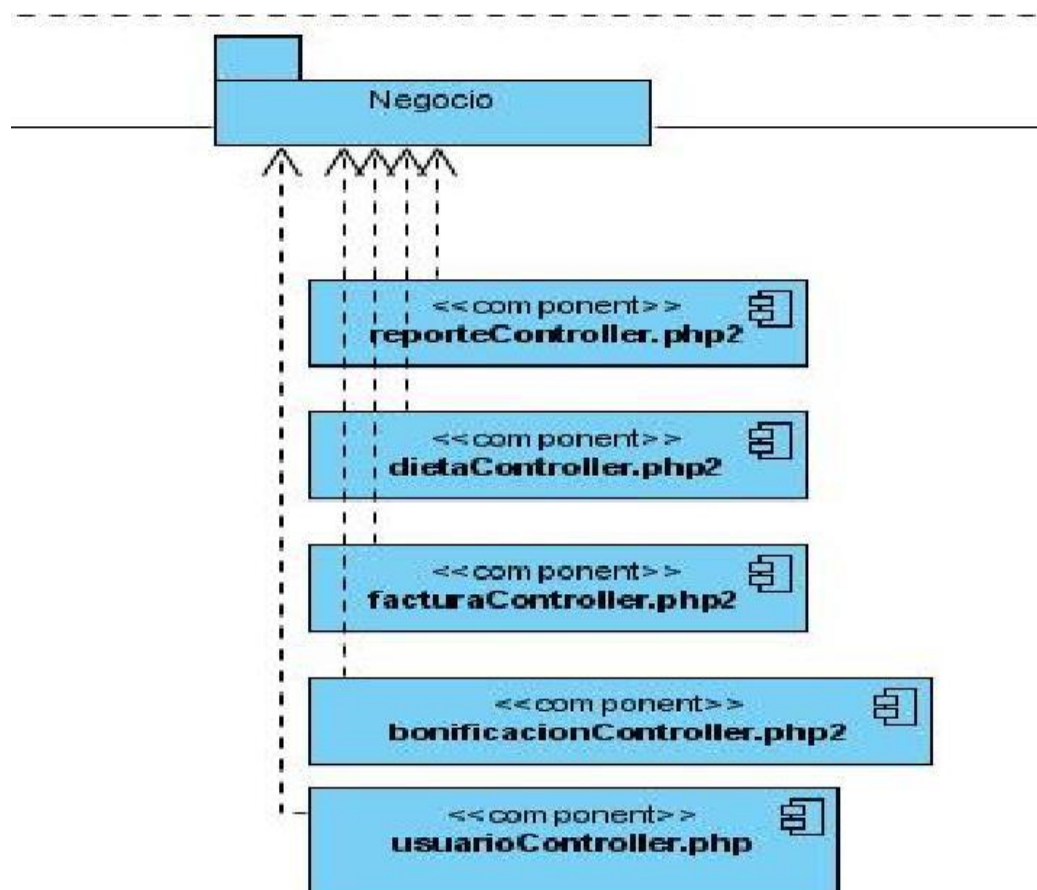


Figura 24 Diagrama de Componentes (Controladoras).

2.6 Seguridad del sistema

La seguridad es un factor de peso en la implementación de cualquier aplicación, esta verifica que la información no pueda ser visualizada por personal no autorizado, o utilizada con fines impropios o contrarios a las políticas del cliente. En la actualidad no se concibe un software de ningún tipo sin pensar en un mecanismo de seguridad que lo respalde.

El sistema, cuenta con un solo usuario, el cual es el único que trabajará con la aplicación, una vez que este se autentica en el sistema, se registra una variable con los datos del usuario, que tendrá un tiempo de vida determinado y que se destruirá una vez que este se desloguee del sistema. En CodeIgniter para tener acceso a un método solamente es necesario escribir en la barra de direcciones del navegador la dirección URL del sitio y seguido el nombre de la clase controladora y el método que se desea invocar, por lo que para determinar si un usuario posee o no permiso para acceder a determinado método se implementó en el constructor de todas las clases controladoras, un mecanismo de verificación.

2.7 Análisis de posibles implementaciones y componentes o módulos que son rehusados

Para la realización de la aplicación se utilizaron componentes que facilitarían el trabajo, debido a que pueden ser rehusados. Se trabajó con la librería JQuery que se utilizó en la capa de presentación, la misma nos permite mejor la calidad en la presentación de los elementos al usuario. Usamos un componente asociado a la librería que se llama DatePicker, para visualizar elementos de tipo calendario, componente que puede ser usado en todas las interfaces sin necesidad de volver a programarlo.

2.8 Descripción de las nuevas clases u operaciones necesarias

Para que la aplicación cumpliera con las funcionalidades que requiere y en correspondencia con el modo que el framework estipula, se implementaron 5 clases controladoras y 4 modelos las cuales se detallan a continuación

2.8.1 Clases Controladoras

Nombre: facturaController	
Tipo de clase: Controladora	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase.
Nombre:	gestionarFactura(\$link,\$icon)
Descripción:	Muestra la vista que permitirá buscar una factura determinada.
Nombre:	obtenerFacturas()
Descripción:	Muestra los datos correspondientes a las facturas registradas.
Nombre:	registrarFactura(\$link,\$icon)
Descripción:	Muestra la vista que permitirá registrar una factura.
Nombre:	insertarFactura(\$link,\$icon)
Descripción:	Inserta los datos de una factura.
Nombre:	eliminarFactura(\$idFactura)
Descripción:	Permite eliminar una factura dada.
Nombre:	cargaModificar (\$idFactura,\$link,\$icon)
Descripción:	Muestra todos los datos que van a modificarse de una factura determinada.
Nombre:	modificarFactura(\$link,\$icon)
Descripción:	Modifica una factura.
Nombre:	buscarFactura(\$cadenaBusqueda)
Descripción:	Busca una factura determinada.

Nombre:	visualizarModelo(\$idFactura,\$link,\$icon)
Descripción:	Visualiza el modelo de la factura seleccionada.

Tabla 1: Descripción de la clase facturaController.

Nombre: dietaController	
Tipo de clase: Controladora	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase.
Nombre:	gestionarDieta(\$link,\$icon)
Descripción:	Muestra la vista que permitirá buscar una dieta determinada.
Nombre:	obtenerDietas()
Descripción:	Muestra los datos correspondientes a las dietas registradas.
Nombre:	registrarDieta(\$link,\$icon)
Descripción:	Muestra la vista que permitirá registrar una dieta.
Nombre:	insertarDieta(\$ (\$link,\$icon)
Descripción:	Inserta los datos de una dieta.
Nombre:	eliminarDieta(\$idDieta)
Descripción:	Permite eliminar una dieta dada.
Nombre:	cargaModificar (\$idDieta,\$link,\$icon)
Descripción:	Muestra todos los datos que van a modificarse de una dieta determinada.
Nombre:	modificarDieta(\$link,\$icon)
Descripción:	Modifica una dieta.
Nombre:	visualizarModelo(\$idDieta,\$link,\$icon)
Descripción:	Visualiza el modelo con todos los datos de la dieta seleccionada.
Nombre:	function buscarDieta(\$cadenaBusqueda)
Descripción:	Busca una dieta determinada.

Tabla 2: Descripción de la clase dietaController.

Nombre: bonificacionController	
Tipo de clase: Controladora	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase.
Nombre:	gestionarBonificacion (\$link,\$icon)
Descripción:	Muestra la vista que permitirá buscar una dieta por concepto de bonificación.
Nombre:	obtenerBonificacion()
Descripción:	Muestra los datos correspondientes a las dietas por bonificación registradas.
Nombre:	registrarBonificacion (\$link,\$icon)
Descripción:	Muestra la vista que permitirá registrar una dieta por concepto de bonificación.
Nombre:	insertarBonificacion (\$ (\$link,\$icon)
Descripción:	Inserta los datos de una dieta por concepto de bonificación.
Nombre:	eliminarBonificacion (\$idDieta)
Descripción:	Permite eliminar una dieta dada por concepto de bonificación.
Nombre:	cargaModificar(\$idDieta,\$link,\$icon)
Descripción:	Muestra todos los datos que van a modificarse de una dieta por bonificación determinada.
Nombre:	modificarBonificacion(\$link,\$icon)
Descripción:	Modifica una dieta por concepto de bonificación.
Nombre:	visualizarModelo(\$idDieta,\$link,\$icon)
Descripción:	Visualiza el modelo con todos los datos de dieta por bonificación seleccionada.
Nombre:	buscarBonificacion(\$cadenaBusqueda)
Descripción:	Busca una dieta por bonificación determinada.

Tabla 3: Descripción de la clase bonificaciónController.

Nombre: reporteController	
Tipo de clase: Controladora	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase.
Nombre:	moduloReporte(\$link,\$icon)

Descripción:	Muestra la vista que permitirá seleccionar el tipo de reporte que desea.
Nombre:	visualizarResumen (\$link,\$icon)
Descripción:	Muestra la vista que permitirá realizar la búsqueda por fechas determinadas.
Nombre:	visualizarDatosResumen (\$fechaInicio,\$fechaFin)
Descripción:	Muestra el resultado de la búsqueda dado dos fechas determinadas
Nombre:	reporteSaldo(\$link,\$icon)
Descripción:	Muestra la vista que permitirá realizar la búsqueda por fechas determinadas y por un saldo específico.
Nombre:	visualizarResumenSaldo(\$fechaInicio,\$fechaFin,\$saldo)
Descripción:	Visualiza el resumen por saldo.

Tabla 4: Descripción de la clase reporteController.

Nombre: usuarioController	
Tipo de clase: Controladora	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase.
Nombre:	index()
Descripción:	Muestra la vista que permitirá autenticarse.
Nombre:	inicioSistema()
Descripción:	Dará inicio a la sección después de haber introducidos el usuario y la contraseña.

Tabla 5: Descripción de la clase usuarioController.

2.8.2 Clases Modelos

Nombre: facturaModel	
Tipo de clase: Modelo	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase.
Nombre:	obtenerFacturas()
Descripción:	Método para obtener todas las facturas registradas.

Sistema de Control de Pago de Dietas y Facturas

Nombre:	eliminarFactura(\$idFactura)
Descripción:	Método para eliminar una factura determinada.
Nombre:	registrarFactura(\$area_solicitante,\$plan_aprobado,\$plan_disponible,\$importe_factura,\$disponibilidad,\$a_favor_de,\$fundamentacion_solicitud,\$fecha_confeccion_modelo,\$tipo_solicitud,\$tipo_pago,\$tipo_moneda,\$forma_pago)
Descripción:	Método para insertar una factura.
Nombre:	registrarFacturaTransferencia(\$idFactura,\$sucursal,\$referenciaTransferencial,\$fecha,\$cuenta)
Descripción:	Método para insertar los datos de la entidad a una factura cuando se realiza el pago por transferencia bancaria.
Nombre:	registrarFacturaCheque(\$idFactura,\$nroCheque,\$fecha)
Descripción:	Método para insertar otros datos de la factura cuando se realiza el pago por cheque.
Nombre:	modificarFactura(\$idFactura,\$area_solicitante,\$plan_aprobado,\$plan_disponible,\$importe_factura,\$disponibilidad,\$a_favor_de,\$fundamentacion_solicitud,\$fecha_confeccion_modelo,\$tipo_solicitud,\$tipo_pago,\$tipo_moneda,\$forma_pago)
Descripción:	Método para modificar los datos de una factura determinada.
Nombre:	modificarFacturaTransferencia(\$idFactura,\$sucursal,\$referenciaTransferencial,\$fecha,\$cuenta)
Descripción:	Método para modificar los datos de la entidad de una factura determinada.
Nombre:	modificarFacturaCheque(\$idFactura,\$nroCheque,\$fecha)
Descripción:	Método para modificar otros datos de una factura determinada cuando presenta el cheque como forma de pago.
Nombre:	obtenerFacturaDadoldFactura(\$idFactura)
Descripción:	Método para obtener los datos de una factura determinada.
Nombre:	eliminarFacturaCheque(\$idFactura)
Descripción:	Método para eliminar otros datos de una factura determinada cuando presenta el cheque como forma de pago.
Nombre:	eliminarFacturaTransf(\$idFactura)
Descripción:	Método para eliminar los datos de la entidad de una factura determinada.
Nombre:	buscarFacturas(\$cadenaBusqueda)
Descripción:	Busca una factura determinada.

Tabla 6: Descripción de la clase facturaModel.

Sistema de Control de Pago de Dietas y Facturas

Nombre: dietaModel	
Tipo de clase: Modelo	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase, por defecto.
Nombre:	obtenerDietas()
Descripción:	Método para obtener todas las dietas registradas.
Nombre:	eliminarDieta(\$idDieta)
Descripción:	Método para eliminar una dietas determinada.
Nombre:	registrarDieta(\$direccion_solicitante,\$nombre_solicitante,\$fecha_confeccion_modelo,\$motivos_viaje,\$salida_estimada,\$salida_real,\$regreso_real,\$dias_viaje,\$real,\$estimado,\$nro_solicitud)
Descripción:	Método para insertar una dieta.
Nombre:	registrarDietaAnticipo(\$id_dieta,\$entregado_almuerzo,\$entregado_hospedaje,\$entregado_desayuno,\$entregado_transporte,\$utilizado_almuerzo,\$utilizado_hospedaje,\$utilizado_desayuno,\$utilizado_transporte,\$devuelto_almuerzo,\$devuelto_hospedaje,\$devuelto_desayuno,\$devuelto_transporte,\$a_entregar_almuerzo,\$a_entregar_hospedaje,\$a_entregar_desayuno,\$a_entregar_transporte)
Descripción:	Método para insertar los datos de una dieta por concepto de anticipo.
Nombre:	registrarDietaLiquidacion(\$id_dieta,\$utilizado_almuerzo,\$utilizado_hospedaje,\$utilizado_desayuno,\$utilizado_transporte,\$a_a_entregar_almuerzo,\$a_a_entregar_hospedaje,\$a_a_entregar_desayuno,\$a_a_entregar_transporte)
Descripción:	Método para insertar otros datos de una dieta por concepto de liquidación.
Nombre:	modificarDieta(\$id_dieta,\$direccion_solicitante,\$nombre_solicitante,\$fecha_confeccion_modelo,\$motivos_viaje,\$salida_estimada,\$salida_real,\$regreso_real,\$dias_viaje,\$real,\$estimado,\$recibido,\$liquidado,\$custodio,\$nro_solicitud,\$nro_cajera)
Descripción:	Método para modificar los datos de una dieta determinada.
Nombre:	obtenerDietaDadoldDieta(\$idDieta)
Descripción:	Método para obtener los datos de una dieta determinada.
Nombre:	eliminarDietaAnticipo(\$id_dieta)
Descripción:	Método para eliminar una dieta determinada por concepto de anticipo.
Nombre:	eliminarDietaLiquidacion(\$id_dieta)
Descripción:	Método para eliminar una dieta determinada por concepto de liquidación.
Nombre:	buscarDietas(\$cadenaBusqueda)
Descripción:	Busca una dieta determinada.

Tabla 7: Descripción de la clase dietaModel.

Nombre: bonificacionModel	
Tipo de clase: Modelo	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase, por defecto.
Nombre:	obtenerBonificaciones()
Descripción:	Método para obtener todas las dietas por bonificación registradas.
Nombre:	eliminarBonificacion(\$idBonificacion)
Descripción:	Método para eliminar una dieta por bonificación determinada.
Nombre:	registrarBonificacion(\$dietaBonificacion)
Descripción:	Método para insertar una dieta por bonificación.
Nombre:	modificarBonificacion(\$dietaBonificacion)
Descripción:	Método para modificar los datos de una dieta por bonificación determinada.
Nombre:	obtenerBonificacionDadoldBonificacion(\$idBonificacion)
Descripción:	Método para obtener los datos de una dieta por bonificación determinada.
Nombre:	buscarBonificaciones(\$cadenaBusqueda)
Nombre:	Busca una dieta por bonificación determinada.

Tabla 8: Descripción de la clase bonificaciónModel.

Nombre: reporteModel	
Tipo de clase: Modelo	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	function __construct()
Descripción:	Constructor de la clase.
Nombre:	visualizarResumen(\$fechaInicio,\$fechaFin)
Descripción:	Visualiza todas las dietas y facturas en un rango de fecha dado.
Nombre:	visualizarResumenSaldo(\$fechaInicio,\$fechaFin,\$saldo)
Descripción:	Visualiza todas las dietas y facturas en un rango de fecha de dado y presenten ese saldo..

Tabla 9: Descripción de la clase reporteModel.

2.9 Conclusiones

En este capítulo queda implementado el sistema y descritas todas las clases utilizadas, cumpliendo con los objetivos específicos planteados y gran parte de las tareas propuestas para la elaboración del sistema, a partir de una breve descripción de como tener acceso a las funcionalidades implementadas; el sistema al estar completamente funcional, se concluye expresando que debería pasar a la etapa de pruebas y realizar los ensayos necesarios para asegurar el sistema libre de no conformidades y con la debida calidad al cliente.

CAPÍTULO 3: “PRUEBA DE LA SOLUCIÓN PROPUESTA”

3.1 Introducción

Uno de los mayores problemas que se afrontan en la actualidad con el desarrollo de software es la calidad de los mismos. Por lo que el proceso de prueba es sin duda uno de los aspectos fundamentales para medir el estado de calidad de una aplicación informática, por lo que se desarrolló este capítulo, con el objetivo de especificar las pruebas que serán hechas al software y describir todo lo encontrado en las mismas.

3.2 Pruebas aplicadas

La prueba de software se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas (configuración de la prueba), registrándose los resultados obtenidos.

Con la realización de estas pruebas se pretende encontrar y documentar los defectos que puedan afectar la calidad del software, validar y probar los requisitos que debe cumplir el software y a su vez que estos fueron implementados correctamente.

Es necesario analizar que las pruebas no pueden asegurar la ausencia de defectos sino que permiten demostrar que existen defectos en el software, que cada prototipo que se quiera entregar al final de una iteración debe ser probado y evaluado.

Todo producto puede ser probado de las siguientes formas:

- Minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- Pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

3.3 Pruebas de unidad

[13] La producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial. (BECK, 2000) [13].

3.3.1 Pruebas de caja blanca

Las pruebas de caja blanca normalmente se denominan pruebas de cobertura o pruebas de caja transparente. Al total de pruebas de caja blanca se le llama cobertura. La cobertura es un número porcentual que indica cuanto código del programa se ha probado. Básicamente la idea de pruebas de cobertura consiste en diseñar un plan de pruebas en las que se vaya ejecutando sistemáticamente el código hasta que haya corrido todo o la gran mayoría de él. Mediante la prueba de caja blanca se puede obtener casos de prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Análisis de complejidad

La Complejidad Ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Se basa en la representación gráfica del flujo de control del programa. De dicho análisis se desprende una medida cuantitativa de la dificultad de prueba y una indicación de la fiabilidad final. Se suele comparar la complejidad ciclomática obtenida contra un conjunto de valores límite como se observa en la tabla 3.1.

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Programa Simple, sin mucho riesgo.
11-20	Más complejo, riesgo moderado.
21-50	Complejo, Programa de alto riesgo.
+50	Programa no testeable, Muy alto riesgo.

Tabla 10 Complejidad ciclomática vs evaluación de riesgo.

Para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del mismo, para hacer dicho cálculo es necesario primero tener el código o el diseño del algoritmo, luego enmarcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo. El código al que se le realiza la prueba se muestra en el Anexo 1.

Después de este paso, es necesario representar el grafo de flujo asociado Figura 21, en el cual se representan distintos componentes como son los círculos que se denominan nodos y representan una o varias sentencias procedimentales. Las flechas se denominan aristas y representan flujo de control.

Una arista debe terminar en un nodo, aún cuando éste no represente ninguna sentencia procedimental. Las áreas delimitadas por aristas y nodos se denominan regiones.

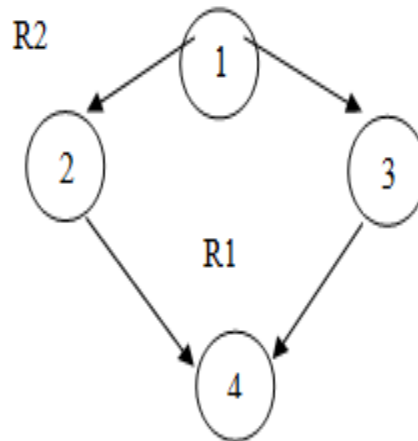


Figura 25 Grafo del flujo del algoritmo.

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías, para concluir que fueron correctos es necesario que el resultado sea el mismo, las fórmulas para calcular son las siguientes:

$$V(G) = A - N + 2$$

Donde A es el número de aristas en el grafo, N es el número de nodos. V se refiere al número ciclomático en teoría de grafos y G indica que la complejidad es una función del grafo.

$$V(G) = 4 - 4 + 2 = 2.$$

$$V(G) = P + 1$$

Siendo "P" la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

$$V(G) = 1 + 1 = 2.$$

$$V(G) = R$$

Siendo "R" la cantidad total de regiones, para cada fórmula "V (G)" representa el valor del cálculo.

$$V(G) = 2.$$

Después de haber realizado el cálculo por las tres vías antes expuestas, se puede concluir que el algoritmo representado anteriormente tiene una complejidad ciclomática de 2, dando una visión de que existen a lo sumo 2 caminos lógicos por donde recorrer el algoritmo. Al tener una complejidad de 2, la evaluación de riesgo dice que es un programa simple, sin mucho riesgo.

3.3.2 Pruebas de caja negra

Se realizará la prueba de Caja Negra ya que esta se centra principalmente en los requisitos funcionales del software. La misma permite obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa. Con este tipo de prueba se ignora la estructura de control, concentrándose en los requisitos funcionales y ejercitándolos.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Dentro del método de Caja Negra se utilizará la técnica de la Partición de Equivalencia ya que es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

3.4 Descripción de las pruebas de caja negra realizadas

3.4.1 Caso de uso: Gestionar dieta.

Descripción general

El caso de uso se inicia cuando el funcionario selecciona el codificador Gestionar Dieta. Este puede insertar, modificar, eliminar o visualizar una determinada dieta, posibilitando que estas sean configurables.

Insertar dieta

Descripción de la funcionalidad

Esta funcionalidad permite adicionar una nueva dieta a las existentes en la base de datos, para ello es necesario que el actor seleccione la opción "Registrar dieta", introduzca los datos para crear la dieta y que los mismos se encuentren libres de errores. Cuando ya el sistema realiza la adición de la nueva dieta notifica que la acción se realizó y muestra una lista con todas las dietas existentes.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de “Módulo de dieta” y seleccionar “Registrar dieta”.
- Introducir los datos necesarios y presionar el botón “registrar” para insertar la nueva dieta.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones:

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Todos los campos del formulario están escritos correctamente.		Se inserta la dieta en la base de datos y se regresa a la página donde se muestra el listado de las dietas.	Se inserta la dieta en la base de datos.	
	1.1. Uno de los campos del formulario esta en blanco.	El sistema muestra un mensaje de error indicando que no puede dejar campos de entrada en blanco.	Por favor, debe llenar todos los campos.	
	1.2. Aparecen caracteres extraños en cualquiera de los campos del formulario. Tales como “@, {, }, [,], *, ^, %, \$, &”	El sistema muestra un mensaje de error.	Por favor, no debe introducir caracteres especiales.	
	1.4. Algunos de los campos de solo letras contienen números o caracteres especiales.	El sistema muestra un mensaje de error.	Por favor, verifique los datos introducidos.	

Tabla 11 Prueba realizada a la funcionalidad Insertar dieta.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Modificar dieta

Descripción de la funcionalidad

Esta funcionalidad permite modificar una dieta de las existentes en la base de datos, para ello es necesario que el actor seleccione la opción modificar dieta, introduzca los datos que desea modificar de la dieta seleccionada previamente y que los mismos se encuentren libres de errores. Cuando ya el sistema actualiza la dieta modificada muestra una lista con todas las dietas existentes.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de Modulo de Dietas y presionar en el icono modificar para la dieta que se desea modificar en el listado de dietas.
- Después de hacer los cambios presionar el botón “registrar cambios” para modificar la dieta seleccionada.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones:

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Todos los campos del formulario están escritos correctamente.		Se inserta la dieta en la base de datos y se regresa a la página donde se muestra el listado de las dietas.	Se inserta la dieta en la base de datos.	
	1.1. Uno de los campos del formulario esta en blanco.	El sistema muestra un mensaje de error indicando que no puede dejar campos de entrada en blanco.	Por favor, debe llenar todos los campos.	

Sistema de Control de Pago de Dietas y Facturas

	1.2. Aparecen caracteres extraños en cualquiera de los campos del formulario. Tales como “@, {, }, [,], *, ^, %, \$, &”	El sistema muestra un mensaje de error.	Por favor, no debe introducir caracteres especiales.	
	1.4. Algunos de los campos de solo letras contienen números o caracteres especiales.	El sistema muestra un mensaje de error.	Por favor, verifique los datos introducidos.	
1. Todos los campos del formulario están escritos correctamente.		Se inserta la dieta en la base de datos y se regresa a la página donde se muestra el listado de dietas.	Se inserta la dieta en la base de datos.	

Tabla 12 Prueba realizada a la funcionalidad Modificar dieta.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Visualizar dieta

Descripción de la funcionalidad

Esta funcionalidad permite visualizar todas las dietas existentes en la base de datos.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de “Modulo de Dieta”.
- Ver listado de actividades que aparece.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Se listaron correctamente todos los datos presentes en la base de datos.		El sistema muestra la lista de las dietas existentes.	Se listaron correctamente todos los datos.	
	1.1. Se introduce en el panel de búsqueda un criterio.	No existe ninguna dieta que cumpla con el criterio de búsqueda introducido.	No hay resultados.	

Tabla 13 Prueba realizada a la funcionalidad Visualizar dieta.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Eliminar dieta

Descripción de la funcionalidad

Esta funcionalidad permite eliminar una dieta de las existentes en la base de datos, para ello es necesario que el funcionario seleccione la dieta que desea eliminar y pulse el icono eliminar. El sistema le pregunta si es seguro que desea eliminar la dieta y en caso de ser afirmativo el sistema elimina la dieta seleccionada de su base de datos notificando que la acción se realizó y mostrando el listado de las dietas actualizado.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a “Modulo de Dietas”.
- En la lista de dietas presionar en el icono eliminar de la dieta que se ese eliminar.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Se eliminó correctamente la dieta de la base de datos.		El sistema vuelve a la página donde se muestra el listado de las dietas actualizado.	La dieta se eliminó de la base de datos.	

Tabla 14 Prueba realizada a la funcionalidad Eliminar dieta.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

3.4.2 Caso de uso: Gestionar factura

Descripción general

El caso de uso se inicia cuando el funcionario selecciona el codificador Gestionar Factura. Este puede insertar, modificar, eliminar o visualizar una determinada factura, posibilitando que estas sean configurables.

Insertar factura

Descripción de la funcionalidad

Esta funcionalidad permite adicionar una nueva factura a las existentes en la base de datos, para ello es necesario que el actor seleccione la opción “Registrar factura”, introduzca los datos para crear la

factura y que los mismos se encuentren libres de errores. Cuando ya el sistema realiza la adición de la nueva factura notifica que la acción se realizó y muestra una lista con todas las facturas existentes.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de “Módulo de Factura” y seleccionar “Registrar factura”.
- Introducir los datos necesarios y presionar el botón “registrar” para insertar la nueva factura.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Todos los campos del formulario están escritos correctamente.		Se inserta la factura en la base de datos y se regresa a la página donde se muestra el listado de factura.	Se inserta la factura en la base de datos.	<Observaciones>
	1.1. Uno de los campos del formulario esta en blanco.	El sistema muestra un mensaje de error indicando que no puede dejar campos de entrada en blanco.	Por favor, debe llenar todos los campos.	
	1.2. Aparecen caracteres extraños en cualquiera de los campos del formulario Tales como “@, {, }, [,], *, ^, %, \$, &”	El sistema muestra un mensaje de error.	Por favor, no debe introducir caracteres especiales.	

	1.4. Algunos de los campos de solo letras contienen números o caracteres especiales.	El sistema muestra un mensaje de error.	Por favor, verifique los datos introducidos.	
--	--	---	--	--

Tabla 15 Prueba realizada a la funcionalidad Insertar factura.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Modificar factura

Descripción de la funcionalidad

Esta funcionalidad permite modificar una factura de las existentes en la base de datos, para ello es necesario que el actor seleccione la opción modificar factura, introduzca los datos que desea modificar de la factura seleccionada previamente y que los mismos se encuentren libres de errores. Cuando ya el sistema actualiza la factura modificada muestra una lista con todas las facturas existentes.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de Modulo de Factura y presionar en el icono modificar para la factura que se desea modificar en el listado de factura.
- Después de hacer los cambios presionar el botón “registrar cambios” para modificar la factura seleccionada.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Todos los campos del formulario están escritos correctamente.		Se inserta la factura en la base de datos y se regresa a la página donde se muestra el listado de las facturas.	Se inserta la factura en la base de datos.	
	1.1. Uno de los campos del formulario esta en blanco.	El sistema muestra un mensaje de error indicando que no puede dejar campos de entrada en blanco.	Por favor, debe llenar todos los campos.	
	1.2. Aparecen caracteres extraños en cualquiera de los campos del formulario. Tales como “@, {, }, [,], *, ^, %, \$, &”	El sistema muestra un mensaje de error.	Por favor, no debe introducir caracteres especiales.	
	1.4. Algunos de los campos de solo letras contienen números o caracteres especiales.	El sistema muestra un mensaje de error.	Por favor, verifique los datos introducidos.	
1. Todos los campos del formulario están escritos correctamente.		Se inserta la factura en la base de datos y se regresa a la página donde se muestra el listado de las facturas.	Se inserta la factura en la base de datos.	

Tabla 16 Prueba realizada a la funcionalidad Modificar factura.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Visualizar factura

Descripción de la funcionalidad

Esta funcionalidad permite visualizar todas las facturas existentes en la base de datos.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de “Modulo de Factura”.
- Ver listado de factura que aparece.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones:

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Se listaron correctamente todos los datos presentes en la base de datos.		El sistema muestra la lista de las facturas existentes.	Se listaron correctamente todos los datos.	
	1.1. Se introduce en el panel de búsqueda un criterio.	No existe ninguna factura que cumpla con el criterio de búsqueda introducido.	No hay resultados.	

Tabla 17 Prueba realizada a la funcionalidad Visualizar factura.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Eliminar Factura

Descripción de la funcionalidad

Esta funcionalidad permite eliminar una factura de las existentes en la base de datos, para ello es necesario que el funcionario seleccione la factura que desea eliminar y pulse el icono eliminar. El sistema le pregunta si es seguro que desea eliminar la factura y en caso de ser afirmativo el sistema elimina la factura seleccionada de su base de datos notificando que la acción se realizó y mostrando el listado de las facturas actualizado.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a "Modulo de Facturas".
- En la lista de facturas presionar en el icono eliminar de la dieta que se ese eliminar.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Se eliminó correctamente la factura de la base de datos.		El sistema vuelve a la página donde se muestra el listado de las facturas actualizado.	La factura se eliminó de la base de datos.	

Tabla 18 Prueba realizada a la funcionalidad Eliminar factura.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

3.4.3 Caso de uso: Gestionar bonificación

Descripción general

El caso de uso se inicia cuando el funcionario selecciona el codificador Gestionar Bonificación. Este puede insertar, modificar, eliminar o visualizar una determinada bonificación, posibilitando que estas sean configurables.

Insertar bonificación

Descripción de la funcionalidad

Esta funcionalidad permite adicionar una nueva bonificación a las existentes en la base de datos, para ello es necesario que el actor seleccione la opción “Registrar bonificación”, introduzca los datos para crear la bonificación y que los mismos se encuentren libres de errores. Cuando ya el sistema realiza la adición de la nueva bonificación notifica que la acción se realizó y muestra una lista con todas las bonificaciones existentes.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de “Módulo de bonificación” y seleccionar “Registrar bonificación”.
- Introducir los datos necesarios y presionar el botón “registrar” para insertar la nueva bonificación.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Todos los campos del formulario están escritos correctamente.		Se inserta la bonificación en la base de datos y se regresa a la página donde se muestra el listado de las bonificaciones.	Se inserta la bonificación en la base de datos.	
	1.1. Uno de los campos del formulario esta en blanco.	El sistema muestra un mensaje de error indicando que no puede dejar campos de entrada en blanco.	Por favor, debe llenar todos los campos.	
	1.4. Algunos de los campos de solo letras contienen números o caracteres especiales.	El sistema muestra un mensaje de error.	Por favor, verifique los datos introducidos.	

Tabla 19 Prueba realizada a la funcionalidad Insertar bonificación.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Modificar bonificación

Descripción de la funcionalidad

Esta funcionalidad permite modificar una bonificación de las existentes en la base de datos, para ello es necesario que el actor seleccione la opción “Modificar bonificación”, introduzca los datos que desea modificar de la bonificación seleccionada previamente y que los mismos se encuentren libres de errores. Cuando ya el sistema actualiza la bonificación modificada muestra una lista con todas las bonificaciones existentes.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de “Modulo de Bonificación” y presionar en el icono modificar para la bonificación que se desea modificar en el listado de las bonificaciones.
- Después de hacer los cambios presionar el botón “registrar cambios” para modificar la bonificación seleccionada.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Todos los campos del formulario están escritos correctamente.		Se inserta la bonificación en la base de datos y se regresa a la página donde se muestra el listado de las bonificaciones.	Se inserta la bonificación en la base de datos.	
	1.1. Uno de los campos del formulario esta en blanco.	El sistema muestra un mensaje de error indicando que no puede dejar campos de entrada en blanco.	Por favor, debe llenar todos los campos.	
	1.2. Aparecen caracteres extraños en cualquiera de los campos del formulario. Tales como “@, {, }, [,], *, ^, %, \$, &”	El sistema muestra un mensaje de error.	Por favor, no debe introducir caracteres especiales.	

	1.4. Algunos de los campos de solo letras contienen números o caracteres especiales.	El sistema muestra un mensaje de error.	Por favor, verifique los datos introducidos.	
--	--	---	--	--

Tabla 20 Prueba realizada a la funcionalidad Modificar bonificación.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Visualizar bonificación

Descripción de la funcionalidad

Esta funcionalidad permite visualizar todas las bonificaciones existentes en la base de datos.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a la parte de “Modulo de Bonificación”.
- Ver listado de bonificaciones que aparece.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Se listaron correctamente todos los datos presentes en la base de datos.		El sistema muestra la lista de las bonificaciones existentes.	Se listaron correctamente todos los datos.	

	1.1. Se introduce en el panel de búsqueda un criterio.	No existe ninguna bonificación que cumpla con el criterio de búsqueda introducido.	No hay resultados.	
--	--	--	--------------------	--

Tabla 21 Prueba realizada a la funcionalidad Visualizar bonificación.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

Eliminar Bonificación

Descripción de la funcionalidad

Esta funcionalidad permite eliminar una bonificación de las existentes en la base de datos, para ello es necesario que el funcionario seleccione la bonificación que desea eliminar y pulse el icono eliminar. El sistema le pregunta si es seguro que desea eliminar la bonificación y en caso de ser afirmativo el sistema elimina la bonificación seleccionada de su base de datos notificando que la acción se realizó y mostrando el listado de las bonificaciones actualizado.

Flujo central

- Autenticarse con el usuario y contraseña correspondiente para entrar al sistema.
- En las opciones del menú ir a “Modulo de Bonificación”.
- En la lista de bonificaciones presionar en el icono eliminar de la bonificación que se quiera eliminar.

Condiciones de ejecución

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
1. Se eliminó correctamente la bonificación de la base de datos.		El sistema vuelve a la página donde se muestra el listado de las bonificaciones	La bonificación se eliminó de la base de datos.	

		actualizado.		
--	--	--------------	--	--

Tabla 22 Prueba realizada a la funcionalidad Eliminar bonificación.

Registro de defectos y dificultades detectados

Primeramente debemos concretar que el objetivo de la prueba realizada no es asegurar la ausencia de defectos en la aplicación, sino demostrar que pueden existir defectos en la misma. El objetivo ha sido diseñar una prueba que saque a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo, satisfactoriamente no se han encontrado defectos ni dificultades.

3.5 Conclusiones

Es importante y relevante recalcar que todas las pruebas que se le pueda realizar a un sistema no lo hace libre de errores ni exento de los mismos, pero con las pruebas realizadas, se garantiza que los módulos de dieta y factura quede limpio de errores, muestra de ello son las pruebas realizadas y sus resultados. En todos los casos las pruebas devolvieron el resultado esperado, asegurando el funcionamiento del sistema para el desarrollo de un sistema con buena calidad.

CONCLUSIONES GENERALES

Una vez concluido el trabajo, se dio cumplimiento al objetivo y las tareas planteadas. Con el sistema completamente implementado se da solución a los problemas existentes en el DTN de la UCI, dando lugar a un mayor control de los procesos de pago. La aplicación desarrollada facilita obtener datos estadísticos así como un mejor manejo de toda la información referente a los procesos de pagos de dietas y facturas.

RECOMENDACIONES

Luego de la presentación del estudio realizado que finaliza con la implementación del Sistema de control de pago para el DTN, se recomienda:

- Instalar el sistema en las facultades regionales (Artemisa, Ciego de Avila y Manzanillo).
- Insertar como nueva funcionalidad a la aplicación la contabilidad del cobro del pasaje a profesores internos (Pase semestral).

REFERENCIAS BIBLIOGRÁFICAS

- [1] Sistemas internacionales (CSGTR01) www.cds-soft.es/documentos/CDSSGI_TRANSPORTE.pdf (15/3/2009).
- [2] Sistemas internacionales (Meribia Transporte) www.unicom.es/software/tir/tir_bloques_factura.html (15/3/2009).
- [3] Aplicaciones Web, (Ventajas y Desventajas) www.avidos.net/blogold/aplicaciones-web/ (17/3/2009).
- [4] Lenguajes de programación (PHP) <http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP> (18/3/2009).
- [5] Ídem a la Referencia 4.
- [6] Ídem a la Referencia 4.
- [7] Lenguajes de programación (javascript) http://www.htmlpoint.com/javascript/corso/js_02.htm (19/3/2009).
- [8] Tecnologías web (Ajax) <http://www.monografias.com/trabajos43/ajax/ajax2.shtml> (19/3/2009).
- [9] Ídem a la Referencia 9.
- [10] Sistema gestor de base dato (Postgres) http://www.netpecos.org/docs/mysql_postgres/x15.html (20/3/2009).
- [11] IDE de Programación (zendstudio) <http://www.maestrosdelweb.com/editorial/zendstudio/> (21/3/2009).
- [12] Framework (codeigniter) http://codeigniter.com/user_guide/ (21/3/2009).
- [13] BECK, K. Extreme Programming Explained. 2000. (20/5/2009).

BIBLIOGRAFÍA

- Glosario de términos <http://www.tripod.lycos.es/support/glossary/C/>
- Resolución No 330/2007 Rector UCI. Control de pago del 50% del Pasaje Estudiantil: Bonificación.
- Proceso de Pago, <http://pdf.rincondelvago.com/el-proceso-de-pago.html>
- Sistema contable ASSETS <http://assets.co.cu/>
- Codeigniter http://codeigniter.com/user_guide/
- Tipos de Prueba <http://eva.uci.cu/mod/resource/view.php?id=14103>
- Modelo de implementación <http://eva.uci.cu/mod/resource/view.php?id=14099>

GLOSARIO DE TÉRMINOS

- **Cookies:** Los archivos de texto descargados en el disco duro del ordenador de un visitante destinado a almacenar las acciones del visitante con miras personalizar mejor sus próximas visitas.
- **CGI (Common Gateway Interface):** El programa o script del lado del servidor utilizado para tratar los datos introducidos en un formulario para llenar.
- **CSS:** Hojas en estilo de cascada aplicables a documentos HTML que contiene diferentes estilos y son fáciles de cambiar y diseñar.
- **DOM (Document Object Model):** Una especificación W3C para interfaces de programa de aplicación para el acceso al contenido de los documentos HTML y XML.
- **HTTP (Hypertext Transfer Protocol):** Protocolo de transmisión del hipertexto.
- **Internet:** Red de computadoras alrededor de todo el mundo que comparten información unas con otras por medio de páginas o sitios.
- **Toolkit:** Es una colección de herramientas integradas que permiten automatizar un conjunto de tareas de algunas de las fases del ciclo de vida del sistema.
- **Páginas webs dinámicas:** Un sitio web que permite interactuar con el visitante, de modo que cada usuario que visita la página vea la información modificada para propósitos particulares.
- **Script CGI:** En español Interfaz común de puerta de enlace. No es en realidad un lenguaje o un protocolo. Es solo es un conjunto de variables y convenciones, nombradas comúnmente, para pasar información en ambos sentidos entre el servidor y el cliente. En sí, es un método para la transmisión de información hacia un compilador instalado en el servidor. Su función principal es la de añadir una mayor interacción a los documentos web que por medio del HTML se presentan de forma estática.
- **XML (Extensible Markup Language):** un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium.

ANEXOS

Funcionalidad “Insertar Dieta”.

```

function insertarDieta($link,$icon) {
    $data['link'] = $link;          (1)
    $data['icon'] = $icon;         (1)
    extract($_POST);(1)
    $this->dietaAD->iniciarTransaccion();      (1)
    $idDieta=$this->dietaAD-
>registrarDieta($direccion_solicitante,$nombre_solicitante,$fecha_modelo,utf8_encode($motivos_viaje
),$f_salida_est,$f_salida_real,$f_regreso_real,$dias_viaje,$real,$estimado,$nro_solicitud);      (1)
    if ($tipo_dieta == 'Anticipo'){          (2)
        $this->dietaAD-
>registrarDietaAnticipo($idDieta,$a_entregado_almuerzo,$a_entregado_hospedaje,$a_entregado_des
ayuno,$a_entregado_transporte,$a_utilizado_almuerzo,$a_utilizado_hospedaje,$a_utilizado_desayuno
,$a_utilizado_transporte,$a_devuelto_almuerzo,$a_devuelto_hospedaje,$a_devuelto_desayuno,$a_de
vuelto_transporte,$a_a_entregar_almuerzo,$a_a_entregar_hospedaje,$a_a_entregar_desayuno,$a_a
entregar_transporte);          (2)
    }
    else if ($tipo_dieta == 'Liquidación') {      (3)
        $this->dietaAD-
>registrarDietaLiquidacion($idDieta,$l_utilizado_almuerzo,$l_utilizado_hospedaje,$l_utilizado_desayun
o,$l_utilizado_transporte,$l_a_entregar_almuerzo,$l_a_entregar_hospedaje,$l_a_entregar_desayuno,$
l_a_entregar_transporte);      (3)
    }
    $this->dietaAD->concluirTransaccion();      (4)
    $this->gestionarDieta($link,$icon);        (4)
}

```