Universidad de las Ciencias Informáticas

Facultad 10

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Sistema de Control de Pagos. Implementación de los módulos de anticipo y liquidación, factura y reporte para el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI.

Autores:

Anael Villalobos Rodríguez.

Luis Felipe Elías Gutiérrez.

Tutor:

Lic. José Manuel de León Cano.

Ciudad de La Habana, 15 de Junio de 2010
"Año del 52 Aniversario del Triunfo de la Revolución"

Dedicatoria

Luis Felipe

A mi madre Cristina y mi padre Luis Felipe que siempre me dieron su apoyo a lo largo de toda la vida de estudiante.

A Leonor y Luis Ernesto por formar parte también de mi familia y me apoyaron siempre.

A mis amigos Anael Villalobos, Yisel Pacheco, Radel Calzada, Camilo Trujillo, Alberto Tamayo, Ronny Zamora, Adrian Garrido, a todos los que empezamos en primer año (10106).

A José Manuel de León que me ayudo a la formación como profesional a lo largo de la carrera.

A todas las personas que me ayudaron día a día, tanto en el trabajo como en el estudio.

Anael

A mis padres Antonia Rodríguez y Flores Villalobos por su apoyo y sus ánimos que me han dado la fe para seguir adelante en cada uno de los momentos de mi vida como estudiantes y por sus consejos que me han convertido en la persona que soy.

A mis segundos padres Jorge Mondejar, Rolando Aparicio, Caridad Guerra y Fredys Vinda por el amor y el cariño que siempre he encontrado en ellos.

A mis hermanos Oisdel Villalobos, Alay Rodríguez y Anay Rodríguez por ayudarme a vencer los obstáculos que me han frenado en momentos difíciles.

A mis segundos hermanos Kendry Portal, Eduan Valdés y Oslay Rodríguez por estar juntos desde pequeños en las buenas y en las malas.

A mis amigos Radel Calzada, Luis Felipe Elías, Camilo Trujillo, Alberto Tamayo, Ronny Zamora, y a todos los que empezamos en primer año (10106) por dedicar parte de sus preciados tiempos a compartir conmigo.

A mis queridos suegros Ismary Duque y Ricardo Pacheco por su atención y dedicación desde que llegué a sus vidas.

En especial a mi querida y amada esposa Yisel Pacheco Duque, que desde que la conocí me ha llenado de momentos de felicidad y ha formado parte de este logro que tanto esperábamos todos.

Agradecimientos

A nuestro tutor José Manuel de León Cano por ayudarnos a superarnos en nuestra estancia durante los cinco años de la carrera.

Al resto del equipo que trabaja en transportaciones que fue una idea que se fue fomentando y ya hoy tiene experiencia en el trabajo.

A todas las personas que nos apoyaron y permitieron el buen desempeño de este trabajo de diploma entre los que se encuentra la Dirección de Gestión Universitaria y al Centro de Informatización Universitaria.

Declaración de autoría

Por este medio declaramos que Anael Villalobos Rodríguez y Luis Felipe Elías Gutiérrez somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Anael Villalobos Rodríguez

_____
Firma del Autor

Luis Felipe Elías Gutiérrez

_____
Firma del Autor

Lic. José Manuel de León Cano

_____
Firma del Tutor

Resumen

La presente investigación consiste en un sistema de control de pagos, implementación de los módulos de anticipo a justificar y liquidación, factura y reporte para el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI. El objetivo de dicho sistema es que sea capaz de agilizar la gestión del proceso de control de pago con el propósito de convertir la información que se genera actualmente de forma manual en dicho grupo a forma digital y automatizar los procesos necesarios para así tener un control más eficiente. Todo esto viene dado a partir de investigaciones realizadas por el Grupo de Gestión y Control de Transportaciones Nacionales en la Dirección de Economía de la Universidad.

El sistema manejará gran cantidad de información que es vital para el funcionamiento de pagos de anticipos a justificar, liquidaciones y facturas emitiendo además reportes de los mismos. Esto se viene haciendo de forma manual por los compañeros que laboran en el Grupo de Gestión y Control de Transportaciones Nacionales y lo que se quiere es informatizar este proceso.

Palabras Claves: UCI, VRE y Dieta (Anticipo a justificar y liquidación).

Índice

Introducción

Desde los inicios el hombre siempre ha buscado la forma de hacer la mayor cantidad de operaciones en el menor tiempo posible. Las eras han ido pasando y el hombre ha ido mejorando su forma de pensar, de protegerse, ha perfeccionado sus herramientas de trabajo para no quedar fuera del desarrollo de las tecnologías de la informática y las comunicaciones.

Hoy en día, una de las herramientas más usadas son las computadoras, que sobrepasan el billón de operaciones por segundo. En la actualidad con la modernización de sistemas de redes, se hace factible una serie de servicios muchos más fáciles de obtener y brindar, con esto los usuarios cada día tienen accesos a realizar muchas tareas con menos tramitación y mucho más rapidez de lo que normalmente se está acostumbrado cuando se realizan por otras vías.

Entre los principales pagos que se realizan en la Universidad de las Ciencias Informáticas son el pago de anticipo a justificar, liquidaciones y liquidación por concepto de bonificación a estudiantes, que no es más que el pago del 50% del pasaje aprobado en el VII Congreso de la FEU, así como anticipos a justificar y liquidaciones a profesores y trabajadores del centro que realizan alguna tarea en colaboración con el mismo. Otro proceso de pago importante lo constituye el pago de facturas a organismos externos de la universidad, que prestan algún servicio a esta.

La Universidad de las Ciencias Informáticas como institución estudiantil de nuestro país posee una Vicerrectoría Económica que se subdivide en diferentes áreas de trabajos que de una forma u otra laboran en conjunta. Esta Vicerrectoría además de atender la Dirección de Contabilidad y Finanzas, Dirección de Planificación y Estadísticas, Dirección de Gestión Energía, Dirección de Auditoría y Control atiende el Grupo de Gestión y Control de Transportaciones Nacionales al cual hacemos referencia en este trabajo, el mismo es el encargado de todo el movimiento de transporte del centro relacionado con las empresas del país que nos prestan el servicio: por ómnibus (Astro), Trenes (FerroCuba), Barco (Viajero) y Dirección Provincial de Transporte (ómnibus Escolares DPT).

Actualmente estos procesos de pago se realizan de forma no automatizada en el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI, lo que implica entre otras dificultades que no se tiene acceso al sistema contable de la dirección de economía del centro y es por eso que se propone automatizar estos procesos mediante la propuesta de un sistema de registro, de control y de reporte de pago de anticipo a justificar, liquidaciones, liquidación de bonificación y facturas relacionadas con los servicios de transportación nacional brindados a nuestra universidad, señalándose que no se quiere suplantar en este tipo de gestión y control todo el proceso económico que lleva la universidad, sino que se quiere realizar un sistema para controlar desde este grupo de gestión y control en conjunto con la Dirección de Economía todos los procesos que se desarrollen.

Por todo lo planteado anteriormente se asume como **problema de la investigación**: ¿Cómo implementar los módulos de anticipo a justificar y liquidación, facturas y reportes para el Sistema de Control de Pagos que automatice los procesos de control de las Transportaciones Nacionales de la Universidad de las Ciencias Informáticas?

El **objeto de estudio** se centra en los procesos de control de anticipo a justificar y liquidación, facturas y reportes por concepto de transportación nacional del Grupo de Gestión y Control de Transportaciones Nacionales de la Universidad de las Ciencias Informáticas.

El **campo de acción** consiste la implementación de los módulos de anticipo a justificar y liquidación, factura y reporte, para la aplicación del Grupo de Gestión y Control de transportaciones nacionales de la Universidad de las Ciencias Informáticas.

Este trabajo de desarrollo tiene como **objetivo general** implementar los módulos anticipo a justificar y liquidación, factura y reporte, para un sistema que controle los pagos que se realizan por concepto de transportaciones nacionales de la Universidad de las Ciencias Informáticas.

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

- Determinar métodos de búsquedas por diferentes parámetros de entrada para consultar la información almacenada en el sistema.

- Estudiar la metodología RUP (Rational Unified Process) en los flujos de trabajo de implementación, prueba y despliegue.
- Estudiar frameworks de desarrollo.
- Identificar las herramientas a utilizar en el desarrollo del sistema.
- Analizar las funcionalidades e importancia de la arquitectura MVC (Modelo Vista Controlador) que propone el framework Codeigniter.

Métodos de Investigación Científica:

Entre los métodos teóricos que se emplearon para nuestra investigación están:

Histórico - Lógico:

- Se estudiaron las diferentes resoluciones que se utilizan en nuestro centro dentro de las direcciones señaladas anteriormente.
- Se buscó en internet algunas documentaciones referentes a temas de pagos de anticipo a justificar, liquidaciones y facturas.
- Se trabajó con la documentación existente de cada una de las formas de pago que ya se han venido realizando.
- Se analizaron algunas resoluciones del Ministerio de Economía y Planificación (MEP) y el Ministerio de Finanzas y Precios.

Como método empírico utilizado para cumplir con las tareas se empleó:

La entrevista:

Se entrevistó al encargado del diseño del sistema de control de pago con el objetivo de comprender a fondo los casos de uso y obtener toda la información acerca de las funcionalidades del sistema según el levantamiento de requisitos que se llevó a cabo con el cliente.

Capítulo 1 “Fundamentación Teórica”

Introducción

Este capítulo abordará el estado actual de desarrollo de las aplicaciones de proceso de pago en el ámbito nacional y en la universidad. Se describirá brevemente como se lleva a cabo el proceso de pago de anticipo a justificar, liquidaciones y facturas en la universidad y se planteará una propuesta de solución para resolver el problema detectado. Además se analizará la propuesta de herramientas y tecnologías a utilizar para el desarrollo del sistema.

1.1 Sistema de pago de anticipo a justificar y liquidación.

El sistema de pago de anticipo a justificar y liquidación se realiza a través de un modelo cual se expide en triplicado por el funcionario autorizado a confeccionar dicho modelo, remitiendo el duplicado al responsable del control de anticipos a los efectos de que éste efectúe las anotaciones correspondientes en el modelo SNC-3-02 – Modelo de Anticipo y Liquidación de Gastos de Viajes **(Anexo 1).**

Estos modelos se envían al cajero para que proceda a efectuar el pago, el cual para poder ser pagado debe tener la firma que autoriza el cobro de anticipo a justificar o liquidación, firma la cual es emitida por algún funcionario autorizado del centro.

Estos pagos de anticipo a justificar y liquidación se pueden hacer de varias formas, por anticipo a utilizar y por liquidación, anticipo a utilizar no es más que una dieta por préstamo de efectivo el cual en algún momento debe ser liquidado y la liquidación no es más que una dieta que se hace de un inicio por liquidación de efectivo el cual no debe ser después justificado, pues ya sale completamente de la caja sin dejar deuda en ella.
Las anotaciones en este modelo, se efectúan en todos los casos a tinta, lápiz-tinta o máquina de escribir.

En el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI se utilizan diferentes conceptos de pagos de anticipo a justificar y liquidación de acuerdo con el tipo que se solicite, es decir, se emplean por:

- Pago por anticipo.
- Pago de liquidación (y dentro de este tipo se encuentran las liquidaciones por bonificación).
- Pagos por consumo de Alimentación y Transporte.

1.1.1 Sistema de Pagos de Anticipos y Liquidaciones existentes en Cuba.

El proceso de pagos de anticipos y liquidaciones en nuestro país se rige a partir de normativas puestas por el Ministerio de Economía y Planificación (MEP) y el Ministerio de Finanzas y Precios los cuales exponen que los pagos de anticipo a justificar y liquidación son autorizados a los funcionarios, empleados y personas designadas por la entidad en el ejercicio de las funciones que se les encomiende.

Este proceso de pago de anticipo a justificar y liquidación procede a partir de que debe existir en cada empresa un funcionario que autorice el pago de la misma 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 todo anticipo a justificar y liquidación solicitado debe tener un justificante que avale el pago sin este aval no se procede a efectuar el autorizo de pago.

Es importante conocer que los pagos por efecto de anticipo a justificar y liquidación no pueden exceder de $500.00 en moneda nacional y en caso de pagos a particulares no se puede exceder de $200.00, todo esto lo ampara la resolución conjunta del Banco Nacional de Cuba y el Ministerio de Finanzas y Precios dictado el 3 de junio de 1998.

Hay empresas como el MITRANS que abala dentro de sus procesos de trabajo las diferentes entidades transportistas existentes en Cuba como son Astro – Ferro Cuba – Viajero y Cubana de Aviación la cuales por motivos de trabajo, muchos de sus empleados solicitan pago de anticipo a justificar y liquidación a partir de algún tipo de tarea que preste a la empresa, es importante conocer que estas empresas pagan por consumo de anticipo a justificar y liquidación a los choferes que realizan viajes fuera de la capital o de cualquier provincia a otra pagándoles una tarifa fija que no excede de más de $16.00 por día.

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 anticipo a justificar y liquidación los departamentos de economía y finanzas de las empresas archivan el concepto de gasto: las cifras pagadas en cada anticipo a justificar y liquidación o pago menor. Este sistema no registra a la persona que se le ha realizado el pago.

1.1.2 Sistema de Pago de Dietas en la Universidad de las Ciencias Informáticas.

La universidad paga un anticipo a justificar o una liquidación por diferentes tipos de conceptos los que fueron mencionados anteriormente. Todo el proceso de anticipo a justificar y liquidación se rige a partir del tipo de concepto por el cual se solicite. Los anticipos a justificar y liquidaciones por pagos anticipados se realizan a aquellas personas que solicitan se les ayude con una proporción de dinero de la caja de la universidad para realizar alguna tarea de la universidad pero tienen la obligación de liquidar el préstamo a su regreso al centro. La liquidación 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 page todo lo que consumió ya sea por efecto de alimentación o de transporte (Gastos Viáticos) todos estos pagos siempre serán autorizadas por un Vicerrector o persona que esté acreditada en la caja de la universidad para poder cobrar un anticipo a justificar o una liquidación.

En la universidad existe un nuevo servicio de pago de liquidación que surge a partir del VII Congreso de la FEU donde sale a relucir el pago del 50% del Pasaje Estudiantil: Bonificación.

1.2 Sistema de Pago de Facturas.

En el Mundo con los diferentes adelantos que existen se pagan las facturas a empresas y a diferentes particulares a partir de tarjetas de créditos cosa que no existía antes de surgir las tarjetas magnetizadas.
Ante se pagaba todo de forma personal, donde cada persona que tenía un servicio que pertenecía a alguna empresa o entidad en específico debía remitirse a las oficinas comerciales a pagar el servicio que estaba recibiendo pero hoy ya se evitan esto con tan solo recibir por email o por formato duro el consumo que debe pagar del tipo que sea y con tan solo acceder a algún cajero automático y teniendo una tarjeta magnética puede liquidar la deuda que posee.

1.2.1 Sistema de Pago de Facturas existente 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 que 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 dos servicios principales que se ofertan: Servicio por factura en formato duro o Servicio por pago a través de tarjetas magnéticas.

Otras empresas que nos 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.

1.2.2 Sistema de Pago de Facturas existente en la UCI.

El sistema de pago de factura en nuestra universidad es un proceso importante para el desarrollo de la misma, pues procede cuando el centro solicita a una entidad o empresa algún tipo de servicio y esta después de ratificar que ayudará con lo solicitado entonces emite una factura la cual será pagada de varias formas de acuerdo con la solicitud del propietario, es decir, si la empresa que preste el servicio quiere que se le page en cheque o transferencia bancaria.

1.3 Pago por anticipo a justificar y liquidación.

1.3.1 Pagos de Anticipo a justificar.

El anticipo a justificar procede cuando una persona va a viajar y realizará consumo por gasto de alimentación o de transporte o solo una de ellas y requiere que se le preste una cantidad de dinero de la caja de la universidad que para ser extraído de la misma, la dirección a la que pertenece la persona que solicita la dieta debe emitir tres modelos de anticipo a justificar y liquidación (Anexo 1) los cuales lleva el nombre de la dirección a la que pertenece la persona que cobrará el dinero, el nombre de la persona que cobrará, el carné de identidad, la fecha de emitido el modelo y las fechas de viaje de ida y regreso así como el consumo que va a realizar. Cómo el anticipo a justificar es un dinero que se da sin justificación desde un inicio entonces la persona debe liquidar en la caja de la universidad a su regreso del dinero que extrajo, para esto la persona que recibió el dinero tiene 72 horas después de su regreso al centro para remitirse a la caja de la universidad y efectuar la liquidación. Es importante conocer que cuando se le da el anticipo a justificar este modelo es autorizado con una firma por el vicerrector del área a la que pertenece la dirección en la casilla de entrega y luego cuando va a liquidar es firmado en la casilla de liquidación y entonces la cajera de la universidad procede a ver si el dinero que sacó de un inicio con los comprobantes que traiga lo liquida todo o si liquida solo una parte y devuelve la otra o que si el consumo que realizó se fue por encima de la cantidad de dinero que extrajo de un inicio, si esto sucede entonces la cajera procede a entregarle el completo que falta el cual fue puesto por la persona de su bolsillo.

1.3.2 Pago por liquidación.

El pago por liquidación procede de la misma forma que se realizan los pagos de anticipados pero tiene algunos cambios incluyendo que las liquidaciones que se hacen de Bonificación siempre son por concepto de liquidación.

Cuando una persona viaja por motivos de trabajo y consume de su bolsillo ya sea por transporte o por gastos de alimentación entonces a su regreso tiene la potestad de solicitar el reintegro siempre y cuando tenga una justificante es decir, la dirección a la que pertenece solicita al vicerrector que los atiende que le sean firmado los tres modelos de anticipo a justificar y liquidación para efectuar el cobro en la caja del consumo y es entonces cuando el vicerrector firma en liquidado y la cajera de la universidad procede a realizar el pago de todo el consumo que haya efectuado la persona a partir de los comprobantes que

abalen el pago. Es importante conocer que el Ministerio de Economía y Planificación tiene una tarifa fija para el pago cuando es por motivos de gastos por alimentación: por desayuno se paga $2.00 por almuerzo $3.50 y por comida $10.00 es decir por día se paga $15.50 en moneda nacional cuando es por divisa entonces se paga de acuerdo con el tipo de servicio que se solicita motivo el cual en el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI no se realiza este tipo de pago en divisa.

En la universidad como se explicó anteriormente se paga el 50% del pasaje estudiantil: Bonificación y este pago se le hace a los estudiantes como proceso de liquidación partir de un mecanismo expuesto en la resolución No 306/2009 emitida por el rector de la universidad y que se debe proceder de la siguiente forma:

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 Grupo de Gestión y Control de Transportaciones Nacionales posee la resolución No 306/2009 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 (Anexo 5). 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 Grupo de Gestión y Control de Transportaciones Nacionales de la UCI procede en realizar la bonificación.

Para realizar el pago, el Grupo de Gestión y Control de Transportaciones Nacionales 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 Grupo de Gestión y Control de Transportación Nacional 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.

Todos estos tipos de procesos de pago de anticipo a justificar y liquidación los realiza Grupo de Gestión y Control de Transportaciones Nacionales de la UCI, pagando los anticipos a justificar y las liquidaciones a los efectos de medios de transporte que sean ómnibus, trenes y barcos, nunca se paga por efecto de viaje en avión a profesores y estudiantes solo se procede a pagar a los vicerrectores del centro, decanos de las Facultades Regionales y al rector de la universidad.

1.4 Pago de Facturas.

El Grupo de Gestión y Control de Transportaciones Nacionales de la UCI realiza pago de factura a diferentes empresas del país que tienen que ver con los procesos de transporte es decir a Astro, FerroCuba, Viajero, Dirección Provincial de Transporte (DPT) y Transmetro.

Para realizar el proceso de pago la universidad contrata a través del MITRAN como empresa rectora de transporte en Cuba el tipo de transporte que se utilizará, cuando esto sucede las empresas que nos presten el servicio emiten una factura por el consumo realizado y esta factura es enviada a la universidad al Grupo de Gestión y Control de Transportaciones Nacionales de la UCI, el cual procede en revisar la factura recibida y la presenta a través de un modelo de solicitud de pago en el cual pone el nombre de la empresa que prestó el servicio y por qué su pago, el valor de la factura a pagar, el concepto por el cual se paga la factura. Después de elaborado este documento se procede a presentar en el comité de compra de la universidad que sesiona dos veces por semana y es el encargado de aprobar el pago solicitado, esta aprobación la da la persona que está al frente de este comité y 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 realizar el tipo de pago solicitado, pues se pueden hacer de dos formas, en cheques o en transferencia bancaria.

Después de realizado todo esto el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI espera una semana para confirmar en el departamento de finanzas y solicitar los datos del tipo de pago que se realizó, es decir, si se hizo un cheque este es recogido y entregado a la empresa que se le paga el servicio, la empresa firma el bauche (documento que ampara que ha sido pagado el cheque a la entidad que prestó el servicio) y entonces el Grupo de Gestión y Control de Transportaciones Nacionales de la

UCI se encarga de entregar en finanzas el bauche. Si lo que se hizo fue una transferencia bancaria entonces se recoge el número referativo de la transferencia que es el que lo identifica así como a la sucursal y número de cuenta a la que se hizo el depósito. Para comprobar que la transferencia se cobró, el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI llama por teléfono a la empresa a la que se le paga el servicio y le da los datos enunciados anteriormente, en caso que la transferencia no llegue después de un mes hecha la empresa debe avisar al Grupo de Gestión y Control de Transportaciones Nacionales de la UCI que no han podido cobrar la transferencia porque no les ha llegado y el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI procede a reclamar en finanzas la transferencia y se comprueba si en el estado de cuenta de la universidad se tiene actualizado el pago.

Todos los datos de pagos que se realizan en el Grupo de Gestión y Control de Transportaciones Nacionales de la UCI se registran en una tabla para llevarse el control de todos los pagos que se realicen.

1.5 Propuesta de solución.

Con la realización de este trabajo se creará un sistema que facilite al Grupo de Gestión y Control de Transportaciones Nacionales de la UCI un control de todos los procesos de pago que se realicen.

El sistema permitirá a los administradores que lo trabajen, llevar un mejor control de pago de anticipo a justificar y liquidación que se realicen de acuerdo con el tipo de concepto por el que se realizan y que permita obtener cifras y datos estadísticos de las personas que realizan la solicitud de pago por concepto de anticipo a justificar y liquidación. Lo mismo sucederá con el pago de facturas que se tenga un mejor control de las facturas que se pagan a provincias desglosándose por municipio y año en que estas se pagan para así tener un mejor control de todo el presupuesto que se va llevando a cabo por año y que para la dirección de economía le sea accesible tener todas estas informaciones para todas las tareas que desarrollan en cuanto a pagos.

1.6 Herramientas de apoyo para la elaboración del sistema: Marco de Trabajo.

Para la elaboración de la propuesta de solución que se plantea en esta investigación se desarrollará 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. Se usará PHP como lenguaje de programación del lado del servidor, PostgreSQL como Sistema Gestor de Base de Datos 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 Java Script del lado del cliente para lograr la interactividad con el usuario en el navegador y específicamente la técnica Jquery, ya que la misma permite actualizar parte de una página en cualquier momento, dándole a los usuarios una respuesta instantánea a sus ingresos y consultas.

1.6.1 Arquitectura Cliente/Servidor.

Se utilizará una arquitectura Cliente/Servidor en el sistema a desarrollar 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 las aplicaciones 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 del 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 ordenadores. Los clientes se suelen situar en ordenadores personales o estaciones de trabajo y los servidores en ordenadores 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.

1.6.2 PostgreSQL.

[1] PostgreSQL se emplea en el sistema a realizar con el objetivo de tener un gestor de base de datos de código abierto que va a ofrecer control de concurrencia multi-versión que va a soportar casi todas las versiones SQL (incluyendo subconsultas, transacciones y tipos de funciones definidas por el usuario) y que permitiría la conexión con diferentes lenguajes de programación como son: C++, Java, PERL, tcl y Python.

1.6.3 PHP.

[2] PHP se utiliza con el sistema a realizar con el objetivo de cumplir varias funciones como son:

- [3] Es rápido en la integración con Bases de Datos como MySQL, MS, SQL, Oracle, Informix, PostgreSQL y el servidor Apache, le permite constituirse como una de las alternativas más atractivas del mercado.
- Su librería: es realmente amplia, lo que permite reducir los llamados ´´costes ocultos´´, uno de los principales defectos de ASP.
- Es suficientemente versátil y potente: como para hacer tanto aplicaciones grandes que necesiten acceder a recursos a bajo nivel del sistema como pequeños scripts que envíen por correo electrónico un formulario rellenado por el usuario.
- Tiene una de las comunidades más grandes en Internet: con lo que no es complicado encontrar ayuda, documentación, artículos, noticias y más recursos.
- Está revolucionando Internet a la vez que evoluciona a pasos gigantescos.

1.6.4 Servidor Web Apache.

Un servidor de páginas Web es un programa que permite acceder a páginas Web alojadas en un ordenador es por eso que se puede decir que hoy Apache es el servidor web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma.

Servidor Web Apache surgió en abril de 1996 y ya en julio del 2002 era utilizado por el 57% de los sitios Web de Internet.

Tiene capacidad de:

- Servir como página con contenido estático.
- Servir como página de contenido dinámico.

A través de otras herramientas puede soportal:

- Bases de Datos.
- Ficheros.

Se puede considerar entonces que es muy potente y altamente configurable y se puede decir que los servidores Web pueden soportar además:

- Protocolos de Transferencia de Hipertexto como HTTP (HyperText Transfer Protocol).

Por mencionar un ejemplo de la forma de trabajo de los servidores web se puede mencionar que cuando un usuario hace clic sobre un enlace, el servidor Web recibe esta solicitud y suministra los datos que le han sido solicitado o bien puede devolver un mensaje de error y no cumpliría el usuario con la solicitud hecha por él.

1.6.5 JavaScript.

[4] Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. 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 en las páginas y 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.

Entre las acciones típicas que se pueden realizar con JavaScript se tienen dos vertientes.

- Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo.
- Por el otro, JavaScript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

1.6.6 JQuery.

[5] JQuery es un framework de JavaScript para facilitar, entre otros, el acceso a los elementos del DOM, los efectos, interactuar con los documentos HTML, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. JQuery podría ser otro framework más como script.aculo.us, MooTools, YUI pero en cambio disponemos una gran potencia con una facilidad mucho mayor que sus competidores.

JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o JQuery().

1.6.7 NetBeans.

[6] NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans.La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo.

Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

1.6.8 CodeIgniter.

[7] 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.
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.

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.

Conclusiones

Con este capítulo se ha hecho un resumen de los aspectos con los cuales labora el Grupo de Gestión y control de Transportaciones Nacionales de la UCI y que sirven para las sedes universitarias en provincia, presentándose una propuesta del sistema que se desea realizar. Se puede ver a simple vista las diferentes tecnologías que pueden ser utilizadas para la elaboración del sistema que se desea realizar.

Capítulo 2 “Descripción y Análisis de la Solución Propuesta”

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 anticipos a justificar, liquidaciones 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.

2.1 Descripción de la solución propuesta.

2.1.1 Autenticar Usuario.

El usuario introduce sus datos en el formulario que se muestra en la figura 2.1, 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 acceso al sistema.

Figura 2.1 Autenticar Usuario.

2.1.2 Gestionar Dieta.

Cuando se selecciona en el menú el vínculo Gestionar Dietas, se le mostrará al usuario insertar una dieta se deben de llenar los campos correspondientes como se muestra en la figura 2.2 y se debe de especificar el tipo de dieta, es decir, si es dieta por liquidación, dieta por anticipo o por concepto de Bonificación como se muestra en la figura 2.3, 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, al seleccionar la opción de modificar le aparecerá un campo para entrar los datos según el criterio de búsqueda como se muestra en la Figuera 2.4 y 2.5, posteriormente aparecerá la dieta la cual se le harán las modificaciones, se actualizan los datos correspondientes y se realiza el mismo proceso de envío y verificación expuesto anteriormente. Para eliminar una dieta determinada, aparecerá un campo para entrar por un criterio de búsqueda la dieta que desea eliminar, el sistema muestra la dieta y mostrará un mensaje informándole si desea eliminar sí o no, de acuerdo con la respuesta del cliente se realiza la operación según muestra la figura 2.6. También puede obtener un reporte de las dietas que se han hecho según el parámetro de búsqueda que desee según muestra la figura 2.7. Se le posibilitará la opción de imprimir el resultado obtenido.

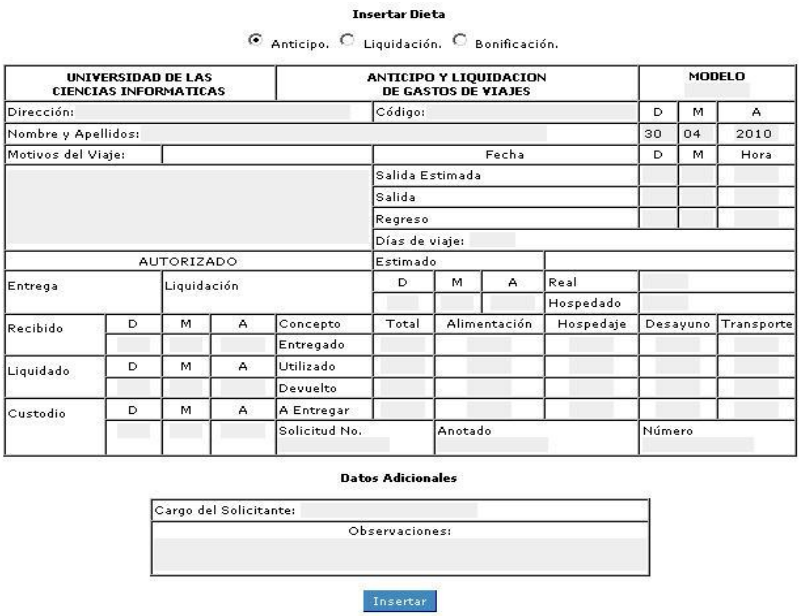

Figura 2.2 Gestionar Dietas (Anticipo o Liquidación).

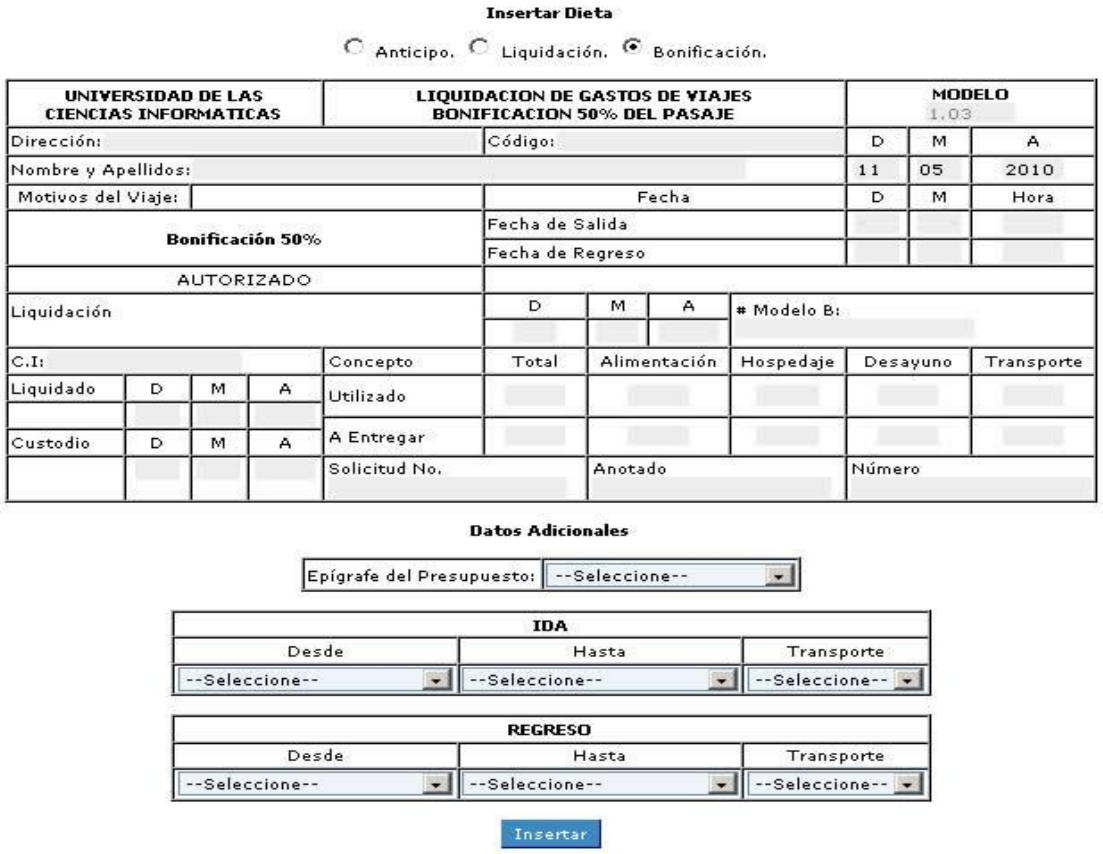

Figura 2.3 Gestionar Dietas (Bonificación).

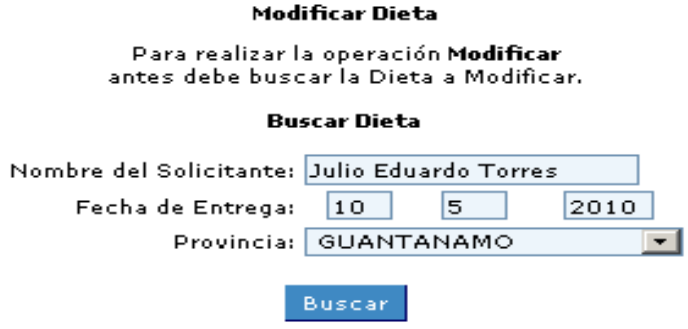

Figura 2.4 Modificar Dieta (Anticipo o Liquidación).

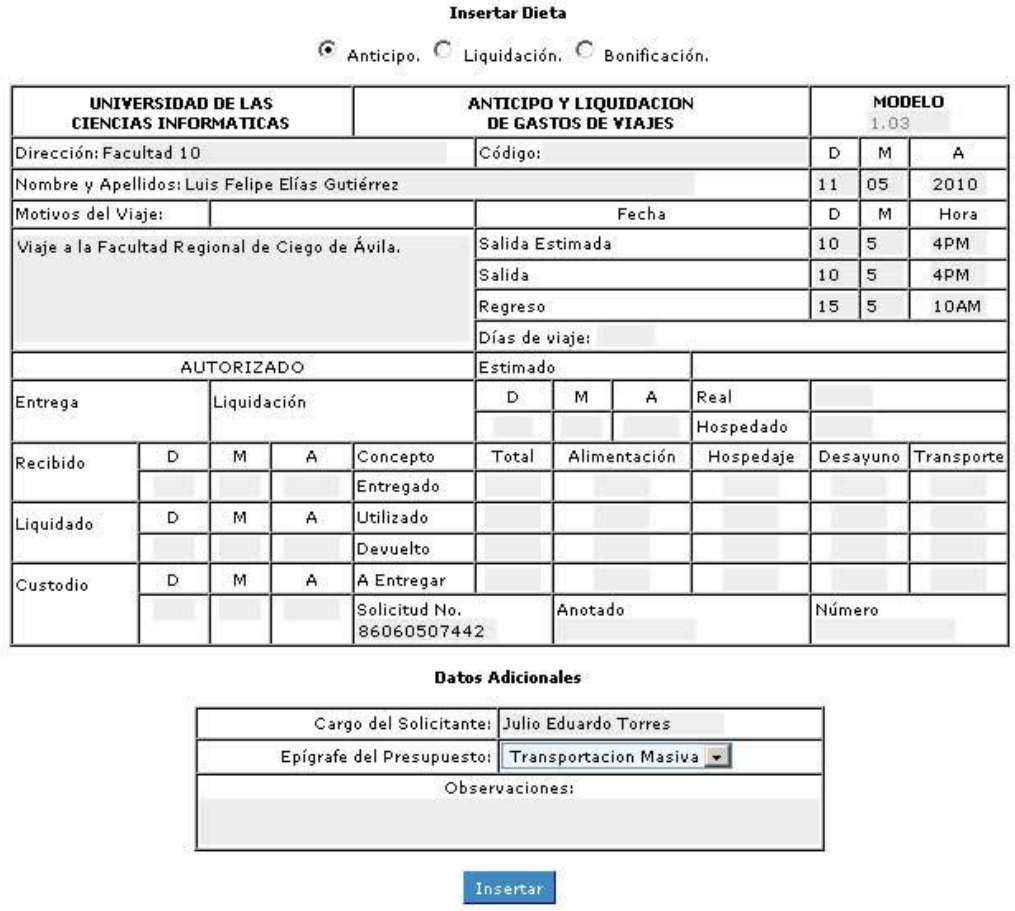

Figura 2.4 Modificar Dieta (Anticipo o Liquidación).

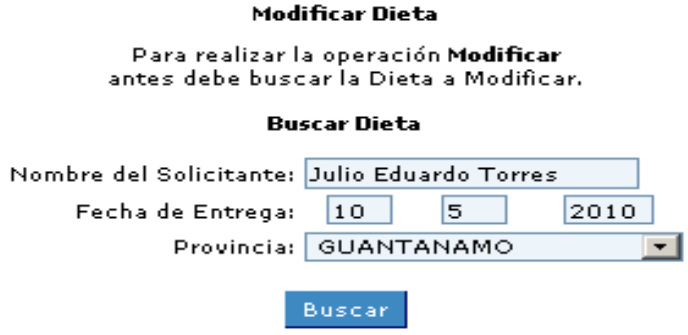

Figura 2.5 Modificar Dieta (Bonificación).

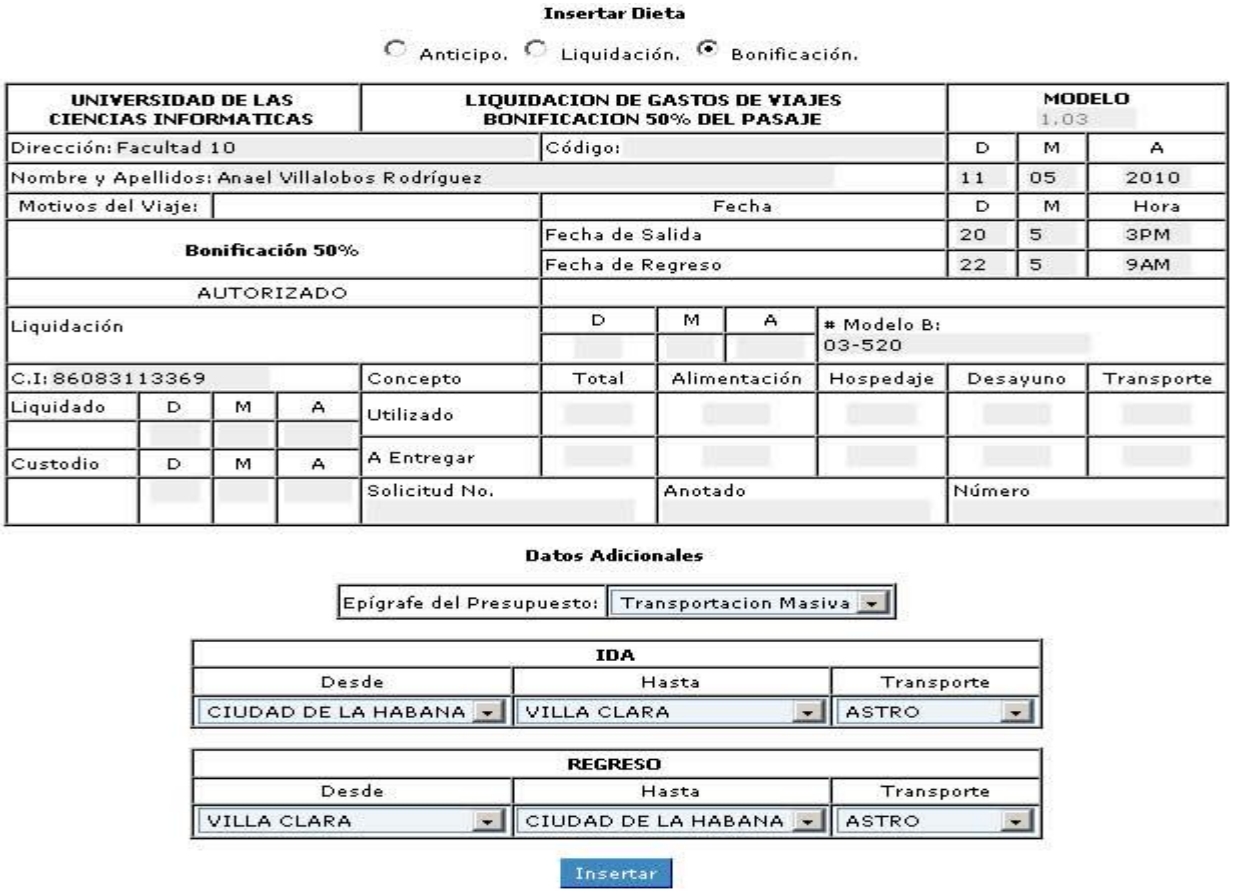

Figura 2.5 Modificar Dieta (Bonificación).

Figura 2.6 Eliminar Dieta.

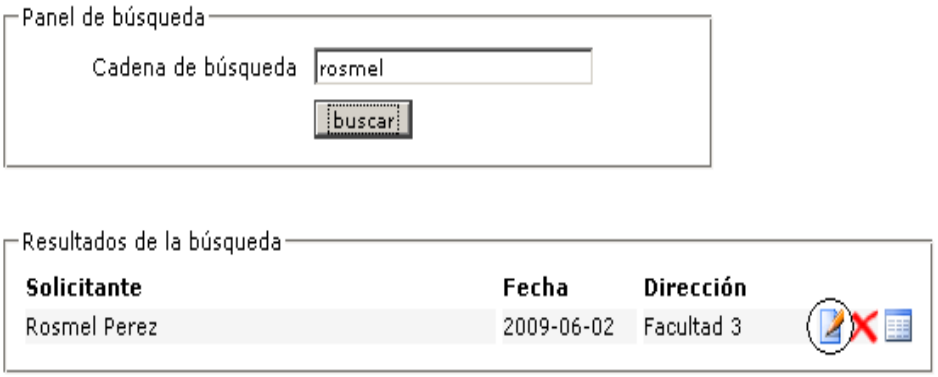

Figura 2.7 Reporte de Dietas.

2.1.3 Gestionar Factura.

Cuando se selecciona en el menú el vínculo Gestionar Facturas, se le mostrará al usuario el formulario para gestionar la factura como se muestra en la figura 2.8. Si se va a insertar una factura se deben de llenar los campos correspondientes como se muestra en la figura y se debe de especificar el estado de la factura si es pendiente, terminada o cancelada, 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, al seleccionar la opción modificar, se muestra un campo para buscar la factura por dos criterios, la cual después de realizada la búsqueda le aparecerá la factura que va a ser modificada como muestra la figura 2.9, una vez seleccionada se actualizan los datos y se realiza el mismo proceso de envío y verificación expuesto anteriormente. También se pueden obtener los reportes de las facturas hechas entrando los distintos parámetros de búsqueda según como muestra la figura 2.8. Para eliminar una factura determinada debe entrar el parámetro de búsqueda la cual le aparece un alista con distintas facturas y usted selecciona la que será eliminada, el sistema mostrará un mensaje si desea eliminar o no, como se muestra en la figura 2.10, de acuerdo con la respuesta del cliente se borran sus datos o no. También se puede obtener una lista de facturas hechas entrando un parámetro como muestra la figura 2.11. Se le posibilitará la opción de imprimir el resultado obtenido.

Figura 2.8 Gestionar Facturas.

Figura 2.9 Modificar Factura.

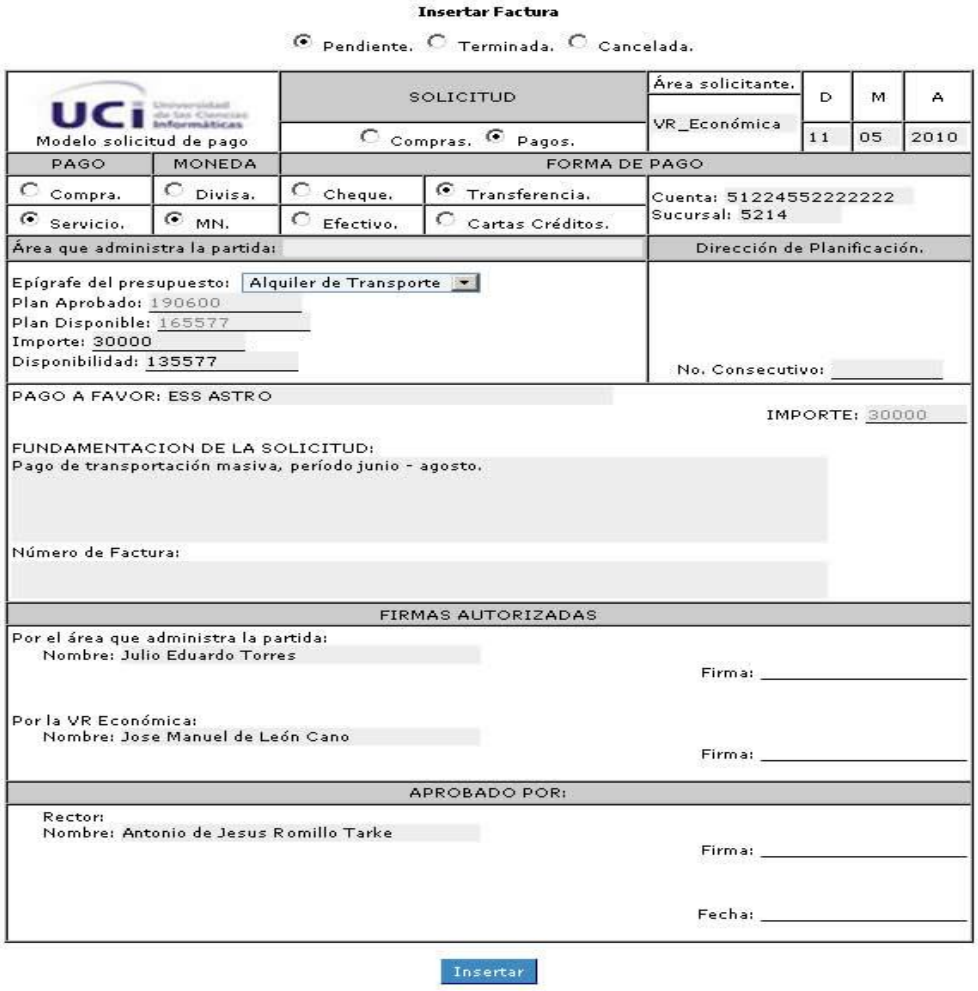

Figura 2.9 Modificar Factura.

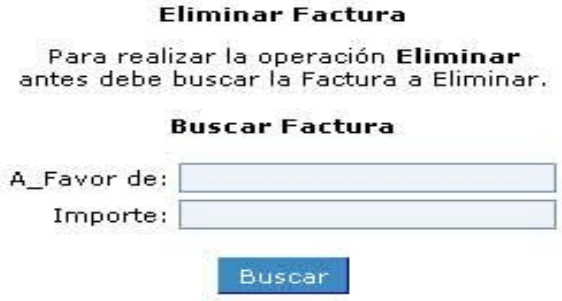

Figura 2.10 Eliminar Factura.

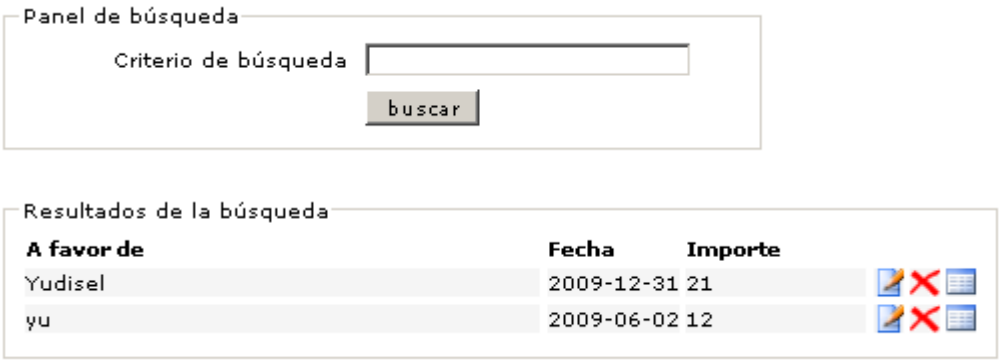

Figura 2.11 Reporte de Factura.

2.1.4 Obtener Reportes.

Cuando se selecciona en el menú el vínculo Reportes, se le mostrará al usuario la opción de obtener el resumen como muestra la figura. En un formulario se le mostrará el rango de fecha a establecer y en una lista de chequeo se le mostrarán los demás criterios de búsquedas que al darle click a alguno se incorpora a la búsqueda como se muestra en la figura 2.12, luego se verifica que los datos sean correctos y se envía esta información y se le mostrará todas las facturas y las dietas por concepto de bonificación, liquidación y anticipo existentes que cumplan con los criterios establecidos anteriormente, como se muestra en la figura 2.13. Se le posibilitará a opción de imprimir el resultado obtenido.

Figura 2.12 Criterios de Búsqueda.

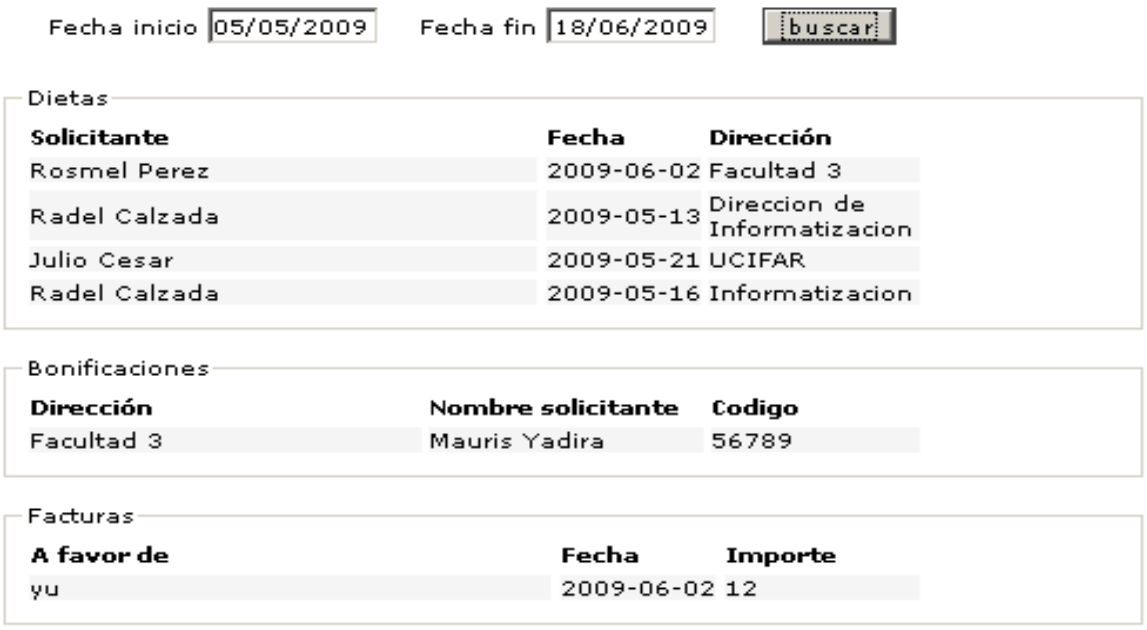

Figura 2.13 Reporte General.

2.2 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 Modelo de la Base de Datos.

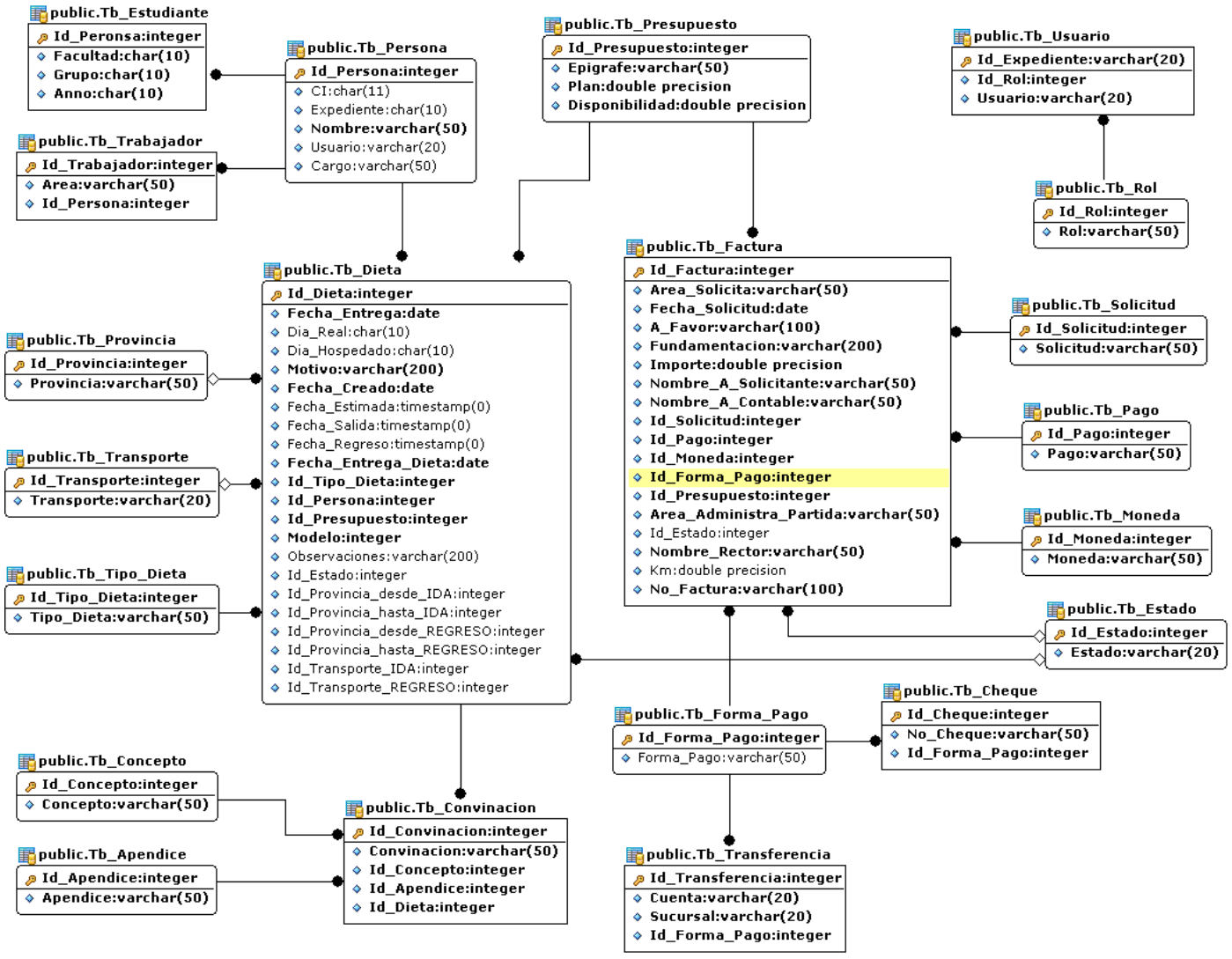

Figura 2.14 Modelo de la Base de Datos.

2.4 Requerimientos Funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Deben ser comprensibles por clientes, usuarios y desarrolladores. Deben tener una sola interpretación y estar definidos en forma medible y verificable. Se mantienen invariables sin importar con que propiedades o cualidades se relacionen. No alteran la funcionalidad del producto.

Seguidamente se hará referencia a los requisitos funcionales del sistema que se propone realizar:

R1 Gestionar usuarios.

- R1.1 Insertar Usuario.
- R1.2 Modificar Usuario.
- R1.3 Eliminar Usuario.
- R1.4 Mostrar Usuarios.

R2 Autenticar usuario.

R3 Gestionar Dieta.

- **R3.1 Insertar Dieta.**
  - R3.1.1 Insertar Dieta por Liquidación.
  - R3.1.2 Insertar Dieta por Anticipo.
  - R3.1.3 Insertar Dieta por Bonificación.
- **R3.2 Modificar Dieta.**
  - R3.2.1 Modificar Dieta por Liquidación.
  - R3.2.2 Modificar Dieta por Anticipo.
  - R3.2.3 Modificar Dieta por Bonificación.
- **R3.3 Eliminar Dieta.**
  - R3.3.1 Eliminar Dieta por Liquidación.
  - R3.3.2 Eliminar Dieta por Anticipo.
  - R3.3.3 Eliminar Dieta por Bonificación.

R4 Registrar Datos de la Bonificación.

R5 Gestionar Factura.

- R5.1 Insertar Factura.
- R5.2 Modificar Factura.
- R5.3 Eliminar Factura.

R6. Visualizar Modelos.

- R6.1 Modelo de Dieta.
- R6.2 Modelo de Factura.
- R6.3 Modelo de Bonificación.

R7. Mostrar Reportes.

- R7.1 Reporte de Dietas.
- R7.2 Reporte de Facturas.
- R7.3 Reporte General.

2.5 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:

2.5.1 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 Grupo de Gestión y Control de Transportaciones Nacionales de la UCI.

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 Grupo de Gestión y Control de Transportaciones Nacionales de la UCI.

2.5.2 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 horas del día.
- El sistema deberá contar con menús con las funciones más importantes del sistema.

2.5.3 Rendimiento.

- Las funcionalidades deben de estar divididas en secciones, de modo que no se sobrecarguen los pedidos.
- 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.

2.5.4 Soporte.

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

2.5.5 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 Grupo de Gestión y Control de Transportaciones Nacionales de la UCI.

2.5.6 Portabilidad.

- El sistema será multiplataforma (Linux o Windows).

2.5.7 Seguridad.

- El usuario debe autenticarse antes de entrar al sistema.

2.5.8 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.

2.6 Patrones o estilos arquitectónicos presente en la solución propuesta.

2.6.1 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 píe 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.

2.6.2 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 del 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.

2.7 Modelo de Implementación.

2.7.1 Modelo de despliegue.

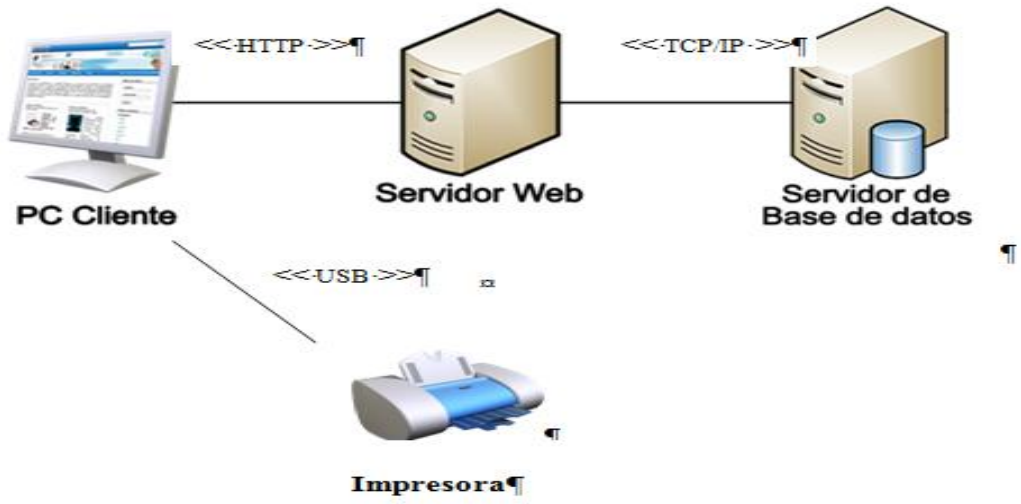

Figura 2.15 Despliegue del sistema.

2.7.2 Diagrama de componentes.

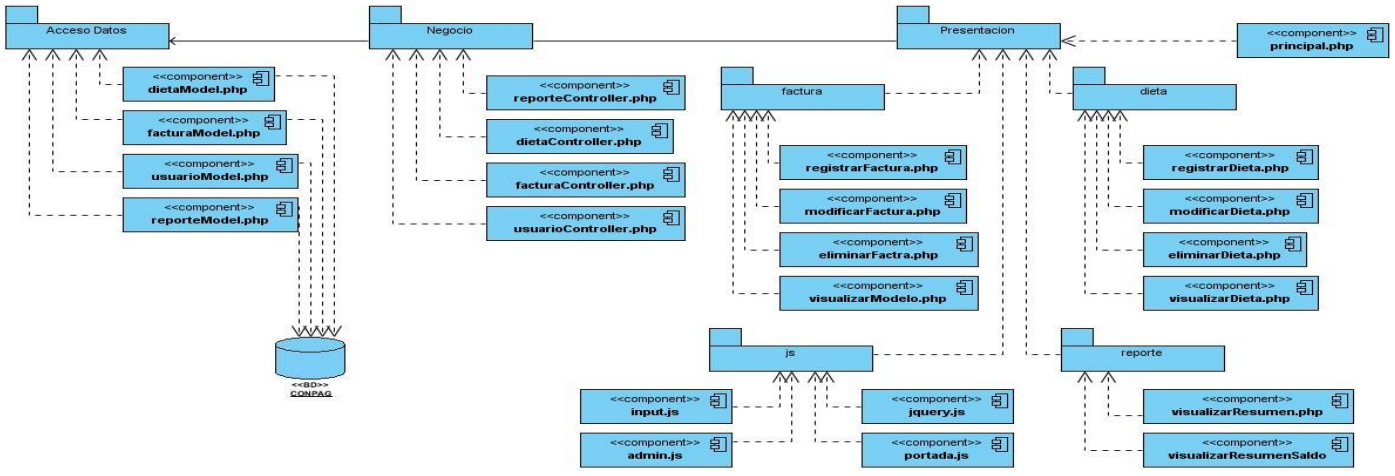

Figura 2.16 Diagrama de Componentes (Global).

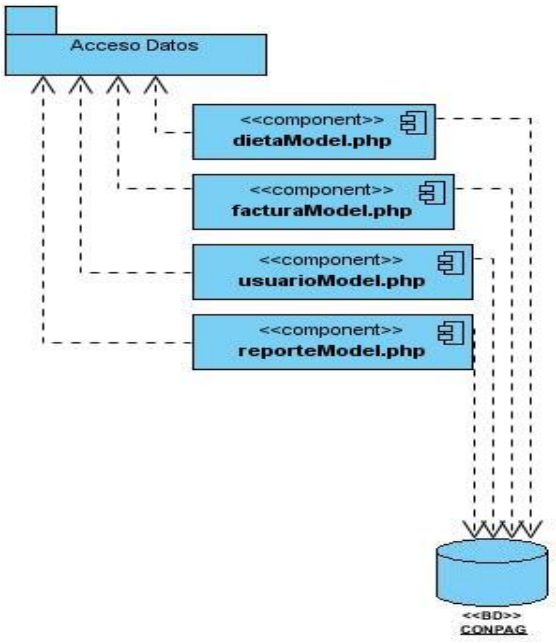

Figura 2.17 Diagrama Componentes (Parte de Acceso a Datos).

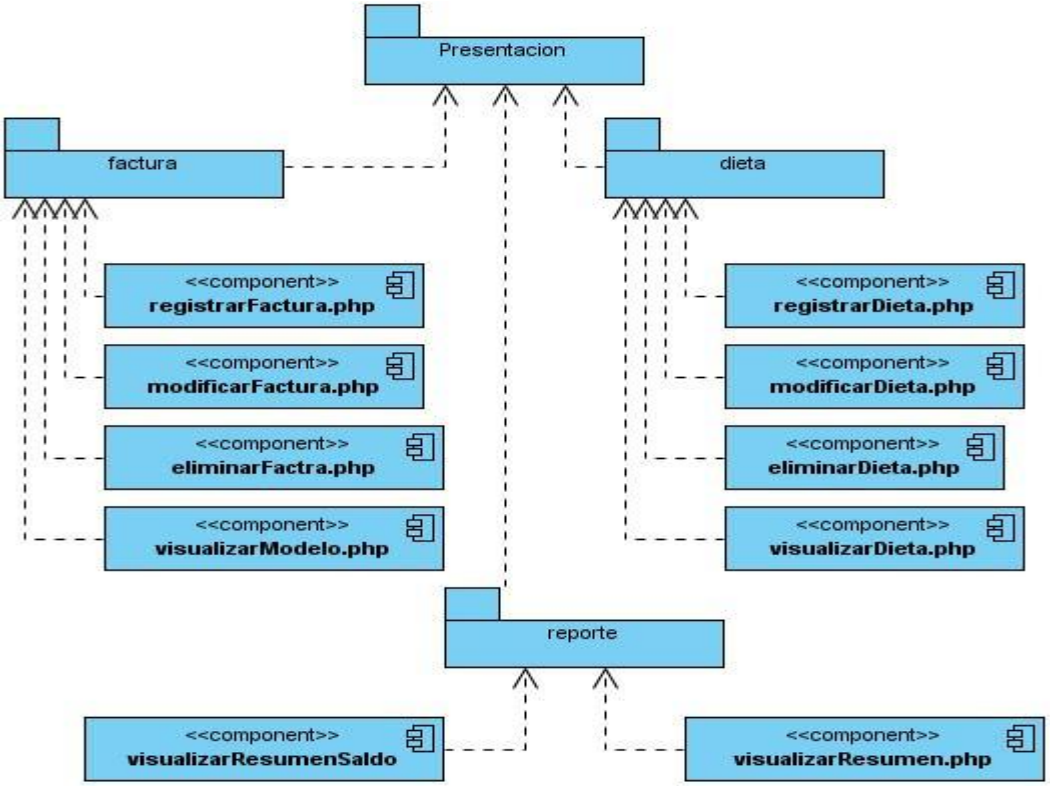

Figura 2.18 Diagrama de Componentes (Vistas).

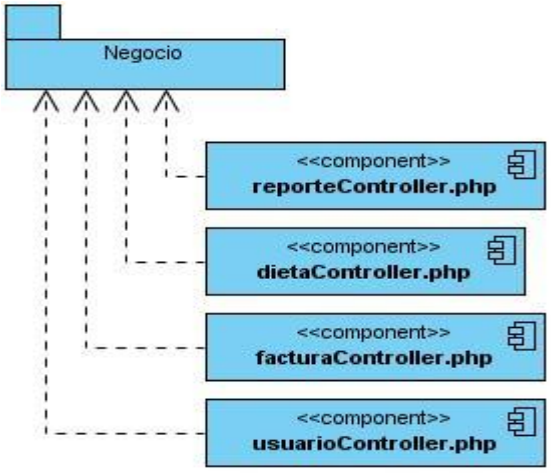

Figura 2.19 Diagrama de Componentes (Controladoras).

2.8 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 deje de estar logueado en el 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.

2.9 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 DataPicker, para visualizar elementos de tipo calendario, componente que puede ser usado en todas las interfaces sin necesidad de volver a programarlo.

2.10 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 4 clases controladoras y 3 modelos las cuales se detallan a continuación.

2.10.1 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 2.1 Descripción de la clase facturaControllers.

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 2.2: Descripción de la clase dietaController.

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 2.3: 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 2.4: Descripción de la clase usuarioController.

2.10.2 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.
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:	obtenerFacturaDadoIdFactura($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 2.5: Descripción de la clase facturaModel.

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($idDieta)
Descripción:	Método para eliminar una dieta determinada.
Nombre:	registrarDieta($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:	obtenerDietaDadoIdDieta($idDieta)
Descripción:	Método para obtener los datos de una dieta determinada.
Nombre:	eliminarDietaAnticipo($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 2.6: Descripción de la clase dietaModel.

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 2.7: Descripción de la clase reporteModel.

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 “Validación de la Solución Propuesta”

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.

3.1 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.

3.1.1 Pruebas de unidad.

[8] 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) [8].

3.1.2 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.

3.1.2.1 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 3.1 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 2.16, 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.

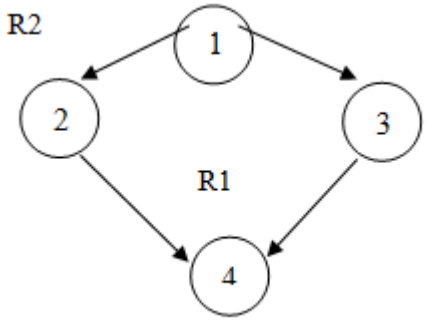

Figura 3.2 Grafo del flujo del algoritmo.

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.

3.1.3 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.

3.2 Descripción de las pruebas de caja negra realizadas.

3.2.1 Caso de uso: Gestionar Dieta.

3.2.1.1 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.

3.2.1.2 Insertar Dieta.

3.2.1.3 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.

3.2.1.4 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 “Insertar dieta”.
- Introducir los datos necesarios y presionar el botón “insertar” para insertar la nueva dieta.

3.2.1.5 Condiciones de ejecución.

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

3.2.1.6 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 a 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.	
	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 3.2 Prueba realizada a la funcionalidad Insertar dieta.

3.2.1.7 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.

3.2.2 Modificar Dieta.

3.2.2.1 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.

3.2.2.2 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 “guardar” para modificar la dieta seleccionada.

3.2.2.3 Condiciones de ejecución.

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

3.2.2.4 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 a 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.	
	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 dieta en a 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 3.3 Prueba realizada a la funcionalidad Modificar dieta.

3.2.2.5 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.

3.2.3 Visualizar Dieta.

3.2.3.1 Descripción de la funcionalidad.

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

3.2.3.2 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.

3.2.3.3 Condiciones de ejecución.

- Para que se pueda ejecutar correctamente este caso de prueba es necesario que el funcionario este autenticado previamente en el sistema y entre el parámetro de búsqueda.

3.2.3.4 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 3.4 Prueba realizada a la funcionalidad Visualizar dieta.

3.2.3.5 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.

3.2.4 Eliminar Dieta.

3.2.4.1 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.

3.2.4.2 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 desee eliminar.

3.2.4.3 Condiciones de ejecución.

Para que se pueda ejecutar correctamente este caso de prueba es necesario que el actor sea un editor nacional autenticado previamente en el sistema y entre el parámetro de búsqueda.

3.2.4.4 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 dietas actualizado.	La dieta se eliminó de la base de datos.	

Tabla 3.5 Prueba realizada a la funcionalidad Eliminar dieta.

3.2.4.5 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.

3.3 Caso de uso: Gestionar Factura.

3.3.1 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.

3.3.2 Insertar factura.

3.3.2.1 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.

3.3.2.2 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 “insertar”.
- Introducir los datos necesarios y presionar el botón “registrar” para insertar la nueva factura.

3.3.2.3 Condiciones de ejecución.

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

3.3.2.4 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 de 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 3.6 Prueba realizada a la funcionalidad Insertar factura.

3.3.2.5 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.

3.3.3 Modificar factura.

3.3.3.1 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.

3.3.3.2 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 “guardar” para modificar la factura seleccionada.

3.3.3.3 Condiciones de ejecución.

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

3.3.3.4 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.	
	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 factura.	Se inserta la factura en la base de datos.	

Tabla 3.7 Prueba realizada a la funcionalidad Modificar factura.

3.3.3.5 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.

3.3.4 Visualizar factura.

3.3.4.1 Descripción de la funcionalidad.

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

3.3.4.2 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.

3.3.4.3 Condiciones de ejecución.

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

3.3.4.4 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 3.8 Prueba realizada a la funcionalidad Visualizar factura.

3.3.4.5 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.

3.3.5 Eliminar Factura.

3.3.5.1 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.

3.3.5.2 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 desee eliminar.

3.3.5.3 Condiciones de ejecución.

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

3.3.5.4 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 facturas actualizado.	La factura se eliminó de la base de datos.	

Tabla 3.9 Prueba realizada a la funcionalidad Eliminar factura.

3.3.5.5 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.

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 queden limpios 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 proceso con buena calidad.

Conclusiones

Con la solución propuesta se cumplieron las principales expectativas en cuanto a configuración, adaptabilidad y eficiencia en los procesos analizados.

- Se logró concebir el módulo anticipo a justificar y liquidación, factura y reporte con la capacidad de procesar 100 solicitudes en 3 segundos desde los servidores asignados donde se demostró la eficiencia y la adaptabilidad a las condiciones de la UCI.
- Se logró implementar el módulo reporte con las facilidades de proporcionar reportes e información necesaria para el control de los pagos evitando la utilización de recursos, tiempo y esfuerzo innecesario en los complejos procesos de pagos.
- Se realizó un estudio de las herramientas informáticas que posibilitarían la implementación de la solución propuesta.
- Se documentaron las funcionalidades de los módulos.

En el desarrollo se optimizaron las funcionalidades al máximo para lograr un producto con la calidad requerida. De una manera u otra se logra dar un uso extremo a la mencionada aplicación siendo la misma capaz de soportarlo.

Con el presente trabajo se demuestra la solución del problema a resolver relacionado con los procesos de gestión de pagos de anticipo a justificar y liquidación, factura y reporte del Grupo de Gestión y Control de Transportaciones Nacionales UCI en la Universidad de las Ciencias Informáticas.

Recomendaciones

Una vez vencidos los objetivos de esta investigación, y teniendo en cuenta las experiencias obtenidas a lo largo de su desarrollo se recomienda:

- Seguir extendiendo el módulo Reportes para lograr brindar la información mucho más completa, personalizada y abarcadora.

Llegar a una solución con las características genéricas necesarias para ser impuesta en cualquier empresa o institución que necesite gestionar pagos de diversos tipos y en diversas condiciones, donde se dejara de pensar un poco en las comodidades específicas del cliente actual y se pensara en algo más abarcador que pueda ser incluso comerciable mundialmente.

Referencias Bibliográficas

[1] Sistema gestor de Base de Datos (Postgres) http://www.netpecos.org/docs/mysql_postgres/x15.html (20/3/2009).

[2]Lenguajes de programación (PHP) http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP (18/3/2009).

[3] Ídem a la Referencia 4.

[4] Lenguajes de programación (JavaScript) http://www.htmlpoint.com/javascript/corso/js_02.htm (19/3/2009).

[5] Tecnologías web (jQuery) http://www.xanelaweb.com/tag/jquery/ (8/2/2010).

[6] IDE de Programación (Netbeans) http://www.maestrosdelweb.com/editorial/netbeans/ (10/2/2010).

[7] Framework (CodeIgniter) http://codeigniter.com/user_guide/ (21/2/2010).

[8] BECK, K. Extreme Programming Explained. 2000. (7/3/2010).

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/

Anexos

Anexo #1: Descripción de las áreas de trabajo de la Vicerrectoría Económica.

Anexo #2: Modelo SNC 3-02 Anticipo y Liquidación de Gastos de Viaje.

UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS				ANTICIPO Y LIQUIDACION DE GASTOS DE VIAJES					MODELO		
Dirección:				Código:					D	M	A
Nombre y Apellidos:											
Motivo del Viaje:				Fecha					D	M	Hora
				Salida Estimada							
				Salida							
				Regreso							
				Días de viaje							
AUTORIZADO				Estimado							
Entrega		Liquidación		D	M	A	Real				
							Hospedado				
Recibido	D	M	A	Concepto	Total	Alimentación	Hospedaje		Desayuno	Transporte	
				Entregado							
Liquidado	D	M	A	Utilizado							
				Devuelto							
Custodio	D	M	A	A Entregar							
				Solicitud No.		Anotado			Número		

Anexo #3: Modelo de Autorizo de la Bonificación.
Nota: Todos los modelos de Bonificación que se emiten llevan un número consecutivo.

UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS				LIQUIDACION DE GASTOS DE VIAJES BONIFICACION 50% DEL PASAJE				MODELO		
Dirección:				Código:				D	M	A
Nombre y Apellidos:										
Motivo del Viaje:				Fecha				D	M	Hora
Bonificación 50%				Fecha de Salida						
				Fecha de Regreso						
AUTORIZADO										
Liquidación				D	M	A	# Modelo B:			
C.I:				Concepto	Total	Alimentación	Hospedaje	Desayuno	Transporte	
Liquidado	D	M	A	Utilizado						
Custodio	D	M	A	A Entregar						
				Solicitud No.		Anotado		Número		

Anexo #4: Modelo de Solicitud de Compras y Pagos.

UCI Universidad de las Ciencias Informáticas Modelo solicitud de pago		SOLICITUD Compras: ____ Pagos: ____		Área solicitante. VR Económica	D	M	A
PAGO	MONEDA	FORMA DE PAGO					
Compra: ____	Divisa: ____	Cheque: __	Tranferencia: ___	Cuenta: Sucursal:			
Servicio: ___	MN: ___	Efectivo: ___	Cartas Créditos___	Otras formas de pago:_____			
Área que administra la partida:		VR Económica		Dirección de Planificación.			
Epigrafe del presupuesto:		Transportación Masiva					
Plan Aprobado: _____							
Plan Disponible: _____							
Importe: _____							
Disponibilidad: _____							
				No. Consecutivo: _____			
PAGO A FAVOR:							
				IMPORTE:			
FUNDAMENTACION DE LA SOLICITUD:							
FIRMAS AUTORIZADAS							
Por el área que administra la partida:							
Nombre:		Firma: _____					
Por la VR Económica							
Nombre:		Firma: _____					
APROBADO POR:							
Rector							
Nombre		Firma: _____					
				Fecha: _____			

Glosario de Términos

1. **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.
2. **CGI** (Common Gateway Interface): El programa o script del lado del servidor utilizado para tratar los datos introducidos en un formulario para llenar.
3. **CSS**: Hojas en estilo de cascada aplicables a documentos HTML que contiene diferentes estilos y son fáciles de cambiar y diseñar.
4. **DOM** (Document Object Model): Una especificación W3C para interfaces de programa de aplicación para el acceso al contenido de los documentos HTML y XML.
5. **HTTP** (Hypertext Transfer Protocol): Protocolo de transmisión del hipertexto.
6. **Internet**: Red de computadoras alrededor de todo el mundo que comparten información unas con otras por medio de páginas o sitios.
7. **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.
8. **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.
9. **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.
10. **XML** (Extensible Markup Language): un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium.