Instituto Superior
Minero Metalúrgico
de Moa

INGENIERÍA INFORMÁTICA
FACULTAD: GEOLOGÍA Y MINAS

TRABAJO DE DIPLOMA

PARA OPTAR POR EL TÍTULO DE

INGENIERO INFORMÁTICO

TÍTULO: SISTEMA DE GESTIÓN DE LA INFORMACIÓN DE DIETAS Y PAGOS MENORES OFICINA MUNICIPAL DE ESTADÍSTICA E INFORMACIÓN DE MOA.

AUTOR: RAFAEL WILLIAM RODRÍGUEZ RIGÑACK
TUTORES: ING. JOSÉ ROLANDO PÉREZ SANDÓ
ING. RAFAEL DACAL DÍAZ

MOA, 2014
"AÑO 56 DE LA REVOLUCIÓN"

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al Instituto Superior Minero Metalúrgico de Moa "Dr. Antonio Núñez Jiménez" y al Departamento de Informática para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los __ días del mes de _____ de 2014.

Rafael William Rodriguez Rigñack _____

Firma Autor

Ing. José Rolando Pérez Sandó _____

Firma Tutor

Ing. Rafael Dacal Díaz _____

Firma Tutor

Pensamiento

Cuando crezcas, descubrirás que ya defendiste mentiras, te engañaste a ti mismo o sufriste por tonterías. Si eres un buen guerrero, no te culparás por ello, pero tampoco dejarás que tus errores se repitan.

Pablo Neruda

Dedicatoria

En el camino de la vida nos enfrentamos a situaciones de todo tipo sean momentos felices o desagradables, hoy es una mezcla de ambos tengo la satisfacción de convertirme en un profesional pero a la vez me falta la persona que supo darme su apoyo incondicional en todos estos años, a quien va dedicado este trabajo íntegramente Maricel Rigñack Garbey a quien llevo en el corazón como una madre ……..

Agradecimientos

Primeramente quisiera agradecer a mi Mamá o mima como le digo cariñosamente.

A mi abuela y abuelo que tanto quiero…

A mi tío Ángel Luis por sacarme de tantos aprietos….

A mis primos los cuales considero como hermanos…..

A mi familia en general que ha sabido brindarme su apoyo incondicionalmente….

A mi grupo de amigos de info los cuales son los mejores……

A mi tutor Pikiri por ayudarme cuando más lo necesitaba…..

A los profesores de la carrera por formarme como un profesional…

RESUMEN

La Oficina Nacional de Estadística e Información del Municipio de Moa (ONEI) tiene como principal misión garantizar la producción de estadísticas de calidad a través del Sistema Estadístico Nacional ejerciendo una adecuada dirección, ejecución y control de la captación de las cifras económicas y sociales. Para ello se le asigna un presupuesto, siendo dos de sus componentes el pago de servicios y los viáticos. Actualmente existen dificultades con el control y procesamiento de la información generada en la gestión de estos componentes del presupuesto.

Esta investigación tiene como objetivo desarrollar una aplicación informática para gestionar el presupuesto asignado para dietas y pagos menores en la ONEI.

La implantación de esta herramienta informática permitirá realizar el seguimiento a la ejecución del presupuesto destinado a pagos menores y viáticos, facilitando la obtención de la información relevante, contribuyendo significativamente a las funciones desempeñadas por el Departamento de Economía de la ONEI en el municipio.

ABSTRACT

The National Office of Statistic and Information of the Municipality of Moa (ONEI) has as main mission to guarantee the production of quality statistical values through the National Statistical System; exercising an appropriate management reception, execution and control of the economic and social figures. In order to accomplish that goal a budget is assigned, being two of their components the payment of services and trips expenses. At the moment difficulties exist with the control and prosecution of the information generated in the administration of these budget components.

The main goal of this investigation is the development of an informatics application to manage the budget assigned for trips expenses and smaller payments in the ONEI.

The deployment of this computer tool will allow carrying out the pursuit of the execution of the budget dedicated to smaller payments and trips expenses, facilitating the obtaining of the outstanding information. It will also contribute significantly to the functions developed by the Department of Economy of the ONEI in the municipality.

Índice

INTRODUCCIÓN

Las Tecnologías de la Informática y las Comunicaciones (TIC) se ven envueltas en un constante y acelerado desarrollo. Uno de los problemas más importantes que se discute en la sociedad cubana se relaciona con el control de los recursos y materiales, como medida preventiva al despilfarro, el delito económico, la malversación y las ilegalidades, entre otros elementos que contribuyen a fomentar las pérdidas de valores y la corrupción de la sociedad. Nuestro país no ha dudado en la utilización de las TIC en busca de soluciones que contribuyan al ahorro de recursos de empresas e instituciones.

La Oficina Nacional de Estadística e Información del Municipio de Moa no queda fuera de este desarrollo informático, ya que pretende informatizar todos los procesos que sean posibles. Esta entidad se encarga de realizar los censos de población y viviendas en el país, encuestas económicas, sociales, demográficas y de salud en el país. Su principal misión es garantizar la producción de estadísticas de calidad a través del Sistema Estadístico Nacional ejerciendo una adecuada dirección, ejecución y control de la captación de las cifras económicas y sociales, para enfrentar las metas del desarrollo económico y social del país y su adecuado reflejo internacional, conscientes de ser útiles y necesarios a la sociedad así como su adecuada difusión de acuerdo con los requerimientos de la economía y las demás necesidades del país en información estadística. Para el cumplimiento de sus objetivos a la ONEI se le asigna un presupuesto, el cual es desglosado para cada uno de los meses del año, acorde a sus niveles de actividad para el período. Dentro de dicho presupuesto se contemplan los pagos menores (pagos por servicios recibidos por parte de terceros) y la entrega de viáticos al personal que labora en la entidad. Debido a la alta movilidad del personal en tareas fuera de la entidad el control de la entregas de viáticos es un aspecto significativo dentro de la ejecución del presupuesto. Por su parte, los pagos de servicios involucran una lista de servicios aprobada anualmente, aunque pueden existir variaciones dentro del año.

El Departamento de Economía tiene la necesidad de informatizar estos procesos para un mejor control de la información. Actualmente dicha gestión está asociada a un conjunto de insuficiencias y deficiencias que afectan el control requerido. Durante la ejecución de estos procesos se genera una gran cantidad de información en la cual queda plasmado todo el movimiento contable referente a las actividades mencionadas, por lo que es de extrema necesidad garantizar una gestión de calidad. Siendo estos elementos auditables, los registros se realizan en un fichero de Microsoft Excel, este tipo de almacenamiento de datos es muy utilizado en el mundo empresarial pero no garantiza totalmente la gestión de toda la información generada, pudiendo introducirse además errores en el momento de la introducción de datos y cifras por parte del personal encargado.

Es importante destacar que existe otra forma de registro de datos, realizada mediante un expediente en formato duro. La necesidad de revisión y consulta constante de estos historiales se torna complicada debido a la gran cantidad de información que generan los procesos de control del presupuesto asociado a los viáticos y pagos menores. Debido a la constante manipulación existe gran riesgo de deterioro y pérdida de documentos, lo cual afectaría de forma directa en operaciones de carácter auditable a la entidad.

También es necesario destacar el tiempo empleado para realizar alguna consulta o búsqueda de información, debido a la alta movilidad de información y procesos que se generan la realización de estos se torna tediosa, provocando pérdidas de tiempo que influyen en el flujo de las actividades realizadas.

Acorde a lo analizando anteriormente se plantea como **problema científico** ¿Cómo mejorar la gestión del presupuesto asignado para dietas y pagos menores en la Oficina Nacional de Estadística e Información de Moa?

Considerándose como o**bjeto de estudio** el proceso de gestión del presupuesto, siendo el **campo de acción** abarcado la informatización de la gestión del presupuesto

asignado a dietas y pagos menores.

La presente investigación tiene como o**bjetivo general**: Desarrollar una aplicación informática para gestionar el presupuesto asignado para dietas y pagos menores en la Oficina Nacional de Estadística e Información de Moa.

Se plantea como **idea a defender** que: El desarrollo de un sistema informático, orientado al control del presupuesto destinado a las dietas y pagos de servicios, dándole seguimiento a la ejecución del mismo y facilitando la obtención de la información relevante, contribuirá significativamente al funcionamiento del Departamento de Economía de la ONEI en el municipio.

Para dar cumplimiento al objetivo de la investigación se definen los siguientes **objetivos específicos:**

1. Establecer el marco teórico-conceptual de la investigación.
2. Desarrollar los principales elementos del ciclo de vida del software, acorde a la metodología seleccionada.
3. Realizar el estudio de factibilidad.

Para darle cumplimiento a los objetivos específicos se han planteado las siguientes **tareas de la investigación:**

- Realizar un análisis del proceso de gestión del presupuesto asignado para dietas y pagos menores en la ONEI.
- Establecer el estado del arte.
- Determinar las necesidades de funcionamiento de la aplicación a desarrollar.
- Seleccionar las herramientas y tecnologías más adecuadas para implementar la propuesta de solución.
- Elaborar la documentación correspondiente a la metodología de desarrollo seleccionada.
- Implementar la aplicación.
- Realizar las pruebas funcionales al sistema construido

- Determinar la factibilidad del sistema.

Para el desarrollo de este trabajo de diploma se utilizaron los métodos que se describen a continuación:

Métodos teóricos

- Histórico – Lógico: en la búsqueda de antecedentes del software, las herramientas utilizadas y la forma en que se realizaba el proceso de gestión de la información de las dietas y pagos menores de la ONEI.
- Análisis y Síntesis: permite alcanzar una profundización en el conocimiento del problema en su totalidad, descomponiéndolo en partes para sintetizar su estudio y precisar sus múltiples relaciones y comportamientos.

Métodos empíricos

- Entrevistas: para determinar los requisitos funcionales de la aplicación que se quiere desarrollar. Se llevaron a cabo varias audiencias con el cliente.
- La observación: es útil para entender el comportamiento de la aplicación y sus especificaciones.
- Análisis de documento: para elaborar los fundamentos teóricos que se relacionan con el campo de acción.

El presente Trabajo de Diploma consta de introducción, cuatro capítulos, conclusiones, recomendaciones y anexos:

Capítulo 1. Fundamentación Teórica: Este capítulo aborda el estado del arte de la investigación, se analizan las tendencias de las herramientas y tecnologías para escoger cuales se utilizarán en el desarrollo de la aplicación.

Capítulo 2. Análisis y diseño de la aplicación: Este capítulo se aplica la metodología escogida para el desarrollo del proyecto abordando sus dos primeras fases.

Capítulo 3. Implementación y realización de pruebas: En este capítulo se muestra la implementación de las tarjetas de ingeniería así como las pruebas realizadas con sus resultados.

Capítulo 4. Estudio de Factibilidad: En este capítulo se realiza un estudio de factibilidad del producto. Además de un estudio de los esfuerzos requeridos para la realización de la aplicación.

Capítulo 1. Marco Teórico- Contextual de la Investigación

1.1 Introducción

En este capítulo se exteriorizarán los antecedentes de la investigación para el establecimiento de marco teórico-contextual, a partir de estudios realizados del tema; se realiza un proceso investigativo de los aspectos teóricos necesarios, se plasma un estudio general de los lenguajes de programación y de los sistemas de bases de datos, la profundización de las diferentes metodologías existentes para el desarrollo de aplicaciones, y la selección de alguna de ellas a utilizar en este trabajo. También, se realiza una descripción de las herramientas y tecnologías utilizadas para el análisis y diseño del software.

1.2 Antecedentes de la investigación

Se realizó un estudio de sistemas semejantes que se relacionen al que se desea obtener.

Sistema web para la gestión de Viajes y Viáticos

Implementación de una solución web que permite la optimización de los procesos de Solicitud, Reservaciones, Relación de Gastos, Liquidación y Contabilidad de todos los gastos asociados a los viajes (Viáticos, Boletos, Incidentales). [1]

Entre las principales características del Sistema Web para la gestión de Viajes y Viáticos se pueden señalar las siguientes:

- Ofrece control del cumplimiento de las políticas de viaje de la empresa.
- Optimiza el proceso de elaboración y aprobación de solicitudes reduciendo el tiempo total de emisión de anticipos y especies.
- Acelera el ciclo de reembolso de gastos.
- Control y seguimiento a la morosidad de empleados.
- Mayor control de pago de anticipos en moneda local, minimizando el riesgo de fraudes.
- Reduce los costos indirectos si se incorpora al viajero y al aprobador al proceso automatizado.

- Permite integrarse a sistemas externos de reservaciones, lo que permite una única interfaz al viajero.

Es un sistema que gestiona la información relacionada con los gastos de un viaje (Viáticos), entre otras muchas funcionalidades que actualmente no son necesarias. Además de ser un software propietario por lo que no se puede cambiar su código para poder adaptarlo a las necesidades y condiciones de la ONEI. En el caso de que se pudiesen realizar cambios sería más trabajoso modificarlo que hacer uno nuevo ya que no existe documentación de la realización del mismo, imposibilitando la completa comprensión del código.

Sistema de Control de Viáticos

El control de viáticos es un sistema eficiente para la administración de la información referente a las cuentas contables de cada concepto este tipo de gastos de los viáticos El programa ofrece todas las operaciones significativamente de alta y folio. Permite entre otras funcionalidades que los usuarios puedan operar fácilmente ya que tiene una captura de los comprobantes de viáticos, además de generar reportes de los viáticos [18]

Este sistema es tiene varias funcionalidades para la gestión de viáticos, pero solo abarca el contenido de los viáticos o dietas, no gestiona la información referente a los demás pagos que aborda la investigación, además de ser una aplicación propietaria, lo cual dificulta la reutilización del código en caso de necesidad de utilizar esta aplicación.

VIÁTICOS - PORTAL

Es una solución para la gestión y aprobación de los gastos de representación que automatiza la entrega de viáticos por los empleados a través de un portal web accesible desde cualquier lugar con conexión a internet. La herramienta permite la administración inteligente de los documentos de viáticos a partir del establecimiento de un flujo de trabajo que se personaliza de acuerdo al organigrama de cada departamento afectado.

Este sistema cuenta con características importantes para la gestión de viáticos:

- Implementa un modelo de gestión electrónica que elimina el uso del papel.
- Reduce los tiempos de revisión y validación de las notas de gastos.
- El empleado genera directamente sus notas de gastos. No es necesario que la captura de datos la realice otra persona, lo que aumenta la eficiencia del proceso.

Es necesario señala que esta aplicación presenta funcionalidades que podrían hacer de la gestión de viáticos un proceso ágil y organizado, pero no contiene algún modulo para la gestión de otros gastos. [19]

1.3 Lenguajes de Programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. Un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. [2]

1.4. C++

Es un lenguaje imperativo orientado a objetos derivado del C. En realidad un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. Estrictamente hablando, C no es un subconjunto de C++; de hecho es posible escribir código C que es ilegal en C++. Pero a efectos prácticos, dado el esfuerzo de compatibilidad desplegado en su diseño, puede considerarse que C++ es

una extensión del C clásico. La definición oficial del lenguaje nos dice que C++ es un lenguaje de propósito general basado en el C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería (en realidad la librería Estándar C es un subconjunto de la librería C++) [3]

1.4.1. Java

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc. con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, AC' estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. La sintaxis del lenguaje heredó características de C y C++, adoptando una muy similar a la del C++. Actualmente, dentro de los lenguajes populares es uno de los mejores en cuanto a definición, debido a que goza de total independencia del implementador del lenguaje y de sus clases auxiliares. Proporciona los tipos de datos primitivos similares a los de C++, proporciona todas las estructuras contenedoras clásicas. Tiene cuatro niveles de empaquetamiento: variables y funciones, al igual que los lenguajes anteriores y otros dos propios de él, denominados: clases y paquetes. Este lenguaje cuenta con una interfaz orientada a objetos para acceder de un modo portable a cualquier base de datos, promoviendo la portabilidad. Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es). [4]

1.4.2 C#

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la norma International 36 e ISO. C# combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, Visual Basic o Delphi. De hecho, su creador Anders Heljsberg fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++. La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, además la migración a este lenguaje por los programadores de C/C++/Java sea lo más inmediata posible. [5]

1.5 Sistemas Gestores de Base de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Algunos ejemplos de SGBD son Oracle, PostgreSQL, MySQL, MS SQL Server, etc.
Un SGBD debe permitir:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes [6].

1.5.1 Oracle

Es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle

Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

Oracle es básicamente una herramienta cliente/servidor para la gestión de bases de datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio provocan que sólo se vea en empresas muy grandes y multinacionales, por norma general. Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux. [7]

1.5.1.1 MySQL

Es un sistema de gestión de bases de datos relacional, fue creada por la empresa sueca MySQL AB, la cual tiene el copyright del código fuente del servidor SQL, así como también de la marca. MySQL es un software de código abierto, licenciado bajo la GPL de la GNU, aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL. [8]

El lenguaje de programación que utiliza MySQL es lenguaje de consulta estructurada (Structured Query Language) que fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales. Inicialmente, MySQL carecía de algunos elementos esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de esto, atrajo a los

desarrolladores de páginas web con contenido dinámico, debido a su simplicidad, de tal manera que los elementos faltantes fueron complementados por la vía de las aplicaciones que la utilizan. Poco a poco estos elementos faltantes, están siendo incorporados tanto por desarrolladores internos, como por desarrolladores de software libre. En las últimas versiones se pueden destacar las siguientes características principales:

El principal objetivo de MySQL es velocidad y robustez.

- Soporta gran cantidad de tipos de datos para las columnas.
- Gran portabilidad entre sistemas, puede trabajar en distintas plataformas y sistemas operativos.
- Cada base de datos cuenta con 3 archivos: Uno de estructura, uno de datos y uno de índice y soporta hasta 32 índices por tabla.
- Aprovecha la potencia de sistemas multiproceso, gracias a su implementación multihilo.
- Flexible sistema de contraseñas (passwords) y gestión de usuarios, con un muy buen nivel de seguridad en los datos.
- El servidor soporta mensajes de error en distintas lenguas.

1.5.1.2 SQL Server

SQL Server es un conjunto de objetos eficientemente almacenados. Los objetos donde se almacena la información se denominan tablas, y estas a su vez están compuestas de filas y columnas. En el centro de SQL Server está el motor de SQL Server, el cual procesa los comandos de la base de datos. Los procesos se ejecutan dentro del sistema operativo y entienden únicamente de conexiones y de sentencias SQL. Incluye herramientas para la administración de los recursos que el

ordenador proporciona y los gestiona para un mejor rendimiento de la base de datos.

Transact-SQL es el lenguaje que utiliza SQL Server para poder enviar peticiones tanto de consultas, inserciones, modificaciones, y de borrado a las tablas, así como otras peticiones que el usuario necesite sobre los datos. El lenguaje estándar SQL se emplea para los sistemas de bases de datos relacionales RDBMS. [9]

Características de Microsoft SQL Server:

Soporte de transacciones. Escalabilidad, estabilidad y seguridad. Soporta procedimientos almacenados. Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red solo acceden a la información. Además permite administrar información de otros servidores de datos. Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en sus versiones 2005 y 2008 pasa a ser el SQL Express Edition, que se distribuye en forma gratuita.

1.5.1.3 PostgreSQl

Es un servidor de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Principales características:

Alta concurrencia: Mediante un sistema denominado MVCC (Acceso Concurrente Multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso

escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos. [10]

- Amplia variedad de tipos nativos. PostgreSQL provee nativamente soporte para:
- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR
- Direcciones MAC.
- Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GIST de PostgreSQL.

Otras características:

- Claves ajenas también denominadas Llaves Ajenas o Claves Foráneas (foreignkeys).
- Disparadores (triggers)
- Vistas.
- Integridad transaccional.
- Herencia de tablas.

- Tipos de datos y operaciones geométricas.

PostgreSQL está considerado como la base de datos de código abierto más avanzada del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características, a partir de PostgreSQL 7.1.x.

DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

Altamente Extensible: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

Soporte SQL Comprensivo: PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial: PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible: La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

Lenguajes Procedurales: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

Cliente/Servidor: PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL, gracias a su licencia BSD, se permite la utilización del código para ser comercializado. Uno de los casos ejemplo es la de Enterprise DB (Postgresql Plus), la cual incluye varios agregados y una interfaz de desarrollo basada en Java.

1.6 Metodologías de Desarrollo de Software

Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales. [11]

1.6.1 Metodologías Tradicionales

Están basadas en las técnicas de toda la vida, aplicadas desde hace mucho tiempo a los problemas de ingeniería. Persiguen la consecución de un proyecto cuya planificación está bien definida. Contando con los medios adecuados para controlar su ejecución, corregir posibles desviaciones y documentar lo realizado.

1.6.1.1 Rational Unified Process (RUP)

Es una metodología tradicional entre sus principales características encontramos que:

- Entrega una forma disciplinada de asignar tareas y responsabilidades.
- Su meta es asegurar la producción de software de alta calidad, que cumpla las necesidades de los usuarios, dentro de las restricciones.
- Utiliza el desarrollo iterativo para enfrentar el riesgo.
- Utilizado por varias empresas a nivel mundial, en grandes y pequeños proyectos.

Las Fases de RUP

- Concepción: especificación de la visión del producto final y su caso de negocio, definiendo el alcance del proyecto.
- Elaboración: planificación de las actividades y recursos necesarios, especificación de las características y el diseño de la arquitectura.
- Construcción: del producto y la evolución de la visión, la arquitectura y los planos, hasta que el producto esté listo para la entrega a la comunidad de usuarios.
- Transición: traspasar el producto a los usuarios, lo que incluye manufacturar, entregar, entrenar, dar soporte y mantener el producto hasta que los usuarios estén satisfechos.[12]

1.6.2 Metodologías Ágiles

Los procesos ágiles de desarrollo de software, conocidos anteriormente como metodologías livianas, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye:

- Planificación
- Análisis de requerimientos
- Diseño
- Codificación
- Revisión y documentación

Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener un demo (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

Ejemplos de Metodologías Ágiles

- Extreme Programming (XP)
- Scrum
- Agile Modeling Adaptive Software Development (ASD)
- Crystal Clear
- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)
- Lean Software Development (LSD)

1.6.2.1 Extreme Programming (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. [13]

Roles XP

- Programador
- Cliente
- Encargado de Pruebas(Tester)
- Encargado de seguimiento (Tracker)

- Entrenador (Coach)
- Consultor
- Gestor (Big Boss)

Características de XP

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- **Refabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.
- **Propiedad del código compartida:** en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- **Simplicidad en el código:** es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Fases de la Metodología (XP):

Fase 1: **Planificación**

- Se escriben historias de usuarios, cuya idea principal es describir un caso de uso en dos o tres líneas con terminología del cliente (*de hecho, se supone que deben ser escritos por el mismo*) de tal forma que creen un test de aceptación para historia de usuarios (*HU*) y permitan hacer una estimación de tiempo de desarrollo del mismo.
- Se crea un plan de lanzamiento que debe servir para crear un calendario que todos puedan cumplir y en cuyo desarrollo hayan participado todas las personas involucradas en el proyecto.
- Se usa como base las historias usuarios, participando el cliente en la elección de las que se desarrollarán, y según las estimaciones de tiempo de los mismos que se desarrollarán, y según las estimaciones de tiempo de los mismos se crearán las iteraciones del proyecto.
- El desarrollo se divide en iteraciones, cada una de las cuales comienzan con un plan de iteración, para el que se eligen las historias de usuarios a desarrollar y las tareas de desarrollo.
- Se cambia el proceso cuanto sea necesario, para adaptarlo al proyecto.

Fase 2: **Diseño.**

- Se eligen los diseños funcionales más simples.
- Se elige una metáfora del sistema para que el nombrado de clases, siga una misma línea, facilitando la reutilización y la comprensión de código.
- Se escriben tarjetas de clases – responsabilidades - colaboración (CRC) para cada objeto, que permita abstraerse al pensamiento estructurado y que el equipo de desarrollo completo participe en el diseño.

Fase 3: **Codificación.**

- El cliente está siempre disponible, de ser posible cara a cara. La idea es que forme parte del equipo de desarrollo y esté presente en todas las fases de XP.

- Es usar el tiempo del cliente para estas tareas en lugar de crear una detallada especificación de requisitos, y evitar la entrega de un producto insuficiente.
- El código se ajustará a unos estándares de codificación, asegurando la consistencia y facilidad de comprensión y refactorización del mismo.
- Las pruebas unitarias se codifican antes que el código en sí, haciendo que la codificación de este último sea más rápida y que cuando se afronte esta misma se tenga más claro qué objetivos tiene que cumplir lo que se va a codificar.
- La programación del código se realiza en parejas, para aumentar la calidad del mismo. En cada momento solo habrá una pareja de programadores que integre el código.
- Se integra código y se lanza dicha integración de manera frecuente, evitando divergencias en el desarrollo. De esta manera se evitará pasar grandes períodos de tiempo integrando el código al final del desarrollo, ya que las incompatibilidades serán detectadas enseguida.
- Se usa la propiedad colectiva del código, lo que se traduce en que cualquier programador puede cambiar cualquier parte del código. El objetivo es fomentar la contribución de ideas por parte de todo el equipo de desarrollo.
- No se hacen horas extras de trabajo.

Fase 4: **Pruebas.**

- Todo el código debe tener pruebas unitarias, y debe pasarlas antes de ser lanzado.
- Cuando se encuentra un error de codificación o bug, se desarrollan pruebas para evitar volver a caer en el mismo error.
- Se realizan pruebas de aceptación frecuentemente, publicando los resultados de las mismas. Estas pruebas son generalmente a partir de las

historias de usuarios elegidas para la iteración y son pruebas de caja negra, en las que el cliente verifica el correcto funcionamiento de lo que se está probando. Cuando se pasa la prueba de aceptación, se considera que la correspondiente historia de usuario se ha completado.

Tabla 1.1 Comparación entre Metodologías Ágiles y Tradicionales

Metodologías Ágiles		Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.		Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.		Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).		Impuestas externamente.
Proceso menos controlado, con pocos principios.		Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.		Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.		El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.		Grupos grandes y posiblemente distribuidos.
Pocos artefactos.		Más artefactos.
Pocos roles.		Más roles.
Menos énfasis en la arquitectura del software.		La arquitectura del software es esencial y se expresa mediante modelos.

1.7. Patrones Arquitectónicos

Los patrones arquitectónicos, o patrones de arquitectura, son patrones de diseño de

software que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen una escalera más grande.

1.7.1 Patrón arquitectónico Modelo Vista Controlador

El patrón arquitectónico Modelo Vista Controlador separa la lógica de negocio (el modelo) y la presentación (la vista) logrando un mantenimiento más rápido y sencillo de las aplicaciones. Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Elementos del patrón:

- **Modelo**: Datos y reglas de negocio.
- **Vista**: Muestra la información del modelo al usuario.
- **Controlador**: Gestiona las entradas del usuario.

1.7.1.1 Arquitectura en Capas.

Arquitectura en capas: es donde se define como organizar el modelo de diseño a través de capas, que pueden estar físicamente distribuidas, lo que quiere decir que los componentes de una capa solo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son consistentes de ningún detalle o interfaz de las superiores.

Figura 1.1: Arquitectura en Capa.

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios, de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario. [14]

1.8 Herramientas

1.8.1 Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. [15]

A partir de la versión 2005 Microsoft ofrece gratuitamente las ediciones para pequeños proyectos (Express Editions), que son varias ediciones básicas separadas por lenguajes de programación o plataforma enfocadas para novatos y entusiastas. Estas ediciones son iguales al entorno de desarrollo comercial pero sin características avanzadas. Dichas ediciones son:

- Visual Basic Express Edition

- Visual C# Express Edition
- Visual C++ Express Edition
- Visual J# Express Edition (Desapareció en Visual Studio 2008)
- Visual Web Developer Express Edition (para programar en ASP.NET)
- Visual F# (Apareció en Visual Studio 2010, es parecido al J#)*

Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de MS SQL Server llamada SQL Server Express Edition cuyas principales limitaciones son que no soporta bases de datos superiores a 4 GB de tamaño, únicamente se ejecuta en un procesador y emplea 1 GB de RAM como máximo, y no cuenta con el Agente de SQL Server.

El paquete Visual Studio Tools for Office (VSTO) añade al **framework .NET** soporte para programar utilizando Word, Excel, Outlook, e InfoPath en Visual Studio. VSTO convierte los documentos de Word o Excel en programables utilizando clases del framework, repletas de soporte para enlace de datos y controles que pueden ser usados en las aplicaciones desarrolladas. Permite integrar de forma sencilla código .NET con Outlook. Permite a los desarrolladores poner código de .NET detrás de las formas de InfoPath. Los desarrolladores pueden incluso programar contra los objetos dominantes de Office sin tener que atravesar el modelo del objeto de Office.

Language Integrated Query (LINQ) con C# y Mono

Language Integrated Query o LINQ es una tecnología integrada en .NET que proporciona la capacidad para consultar o manipular diversas fuentes de datos, independientes del proveedor utilizando de forma nativa la sintaxis de cualquier lenguaje de programación soportado por .NET, lo cual nos proporciona el soporte del compilador y nos permite concentrarnos únicamente en las búsquedas en lugar de como hacer la rutina para cada búsqueda, además la sintaxis de LINQ es similar a SQL lo que nos proporciona un estándar ya que es la misma sintaxis para todas las fuentes

de datos diferentes o similares. Dependiendo de la fuente de datos a trabajar, es el componente LINQ a utilizar, los componentes se agrupan en:

LINQ to SQL: Es el conjunto de clases, estructuras, interfaces y enumeraciones utilizadas para escribir consultas a bases de datos relacionales como PostgreSQL, SQL Server o MySQL.

LINQ to Objects: Es la API predeterminada de LINQ y permite escribir consultas para arreglos, estructuras y colecciones de objetos en memoria.

LINQ to XML: Proporciona la habilidad de escribir consultas para procesar fuentes de datos XML.

LINQ to DataSet: Es la API dedicada a trabajar con clases DataSets y DataTables, ya que aún existen aplicaciones y desarrolladores que utilizan esta solución.

LINQ es una más de las muchas tecnologías que ofrece .NET y Mono para facilitar el trabajo a los desarrolladores en cuanto al trabajo con fuentes de datos, LINQ es un enfoque total de llevar la manipulación de datos en los lenguajes orientados a objetos, librando los inconvenientes de trabajar con los lenguajes de manipulación de datos que cada proveedor posee.

1.8.1.1Embarcadero ER/Studio 8.0

Embarcadero ER/Studio: Es una herramienta de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejos.

ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes [16].

ER/Studio ofrece las siguientes funcionalidades:

- Capacidad fuerte en el diseño lógico.
- Sincronización bidireccional de los diseños lógico y físico.
- Construcción automática de Base de Datos.
- Reingeniería inversa de Base de Datos.
- Documentación basada en HTML.
- Un repositorio para el modelado.

1.9 Conclusiones del capítulo

Luego de un estudio realizado, abordando los principales conceptos, profundización en las herramientas, lenguajes de programación y metodologías de desarrollo, quedan sentadas las bases para el desarrollo del sistema de gestión de dietas y pagos menores de la ONEI, se ha decidido utilizar como metodología de desarrollo Extreme Programming (XP) debido a que promueve un constante intercambio con el cliente para la construcción del sistema. Como herramienta CASE, se selecciona para la construcción del modelo de datos al Embarcadero ER/Studio 8.0, debido a las facilidades que provee para el desarrollo de esta actividad. El lenguaje de programación será el C# 4.0, desarrollándose bajo el IDE Visual C# Express Edition 2008 dada la rapidez de construcción de las aplicaciones en este entorno. Para facilitar el trabajo con el paradigma orientado a objetos en la aplicación se utilizará el ORM LINQ, como mediador entre la capa de datos y la capa de negocio. Los datos serán manejados por el sistema gestor de base de datos PostgreSQL, considerado por la comunidad como el SGBD libre más potente en la actualidad.

Capítulo 2. Planeación y diseño

2.1 Introducción

En este capítulo, se establece la fase de planificación y diseño. En él se especifican las necesidades del cliente. También se figuran los requisitos funcionales y no funcionales de la aplicación propuesta que serán objeto de automatización mediante el empleo de las historias de usuarios (HU). Además se realiza una estimación del esfuerzo necesario para las mismas y se establece un plan de iteraciones necesarias sobre el sistema, para su posterior culminación, la creación de las tarjetas Clases-Responsabilidades-Colaboración conocidas por sus siglas: (CRC) y el modelo de datos.

2.2 Personal relacionado con la aplicación

Tabla 2.1 Personal relacionado con la aplicación

Personas relacionadas con la aplicación	**Justificación**
Especialista (contador)	Esta es la persona que tiene mayor conocimiento en la materia de gestión contable y está encargada de la gestión de la misma
Informático	Es la persona encargada de administrar la aplicación.

2.3 Funcionalidades del sistema

Después de conocer el personal relacionado, se procede a realizar el análisis de las funcionalidades que debe cumplir la aplicación. Para ello se enumerarán las funcionalidades que el sistema debe ser capaz de cumplir. Es el primer artefacto generado en la etapa de captura de requisitos, está conformada por una lista priorizada

que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para una iteración.

Tabla 2.2: Funcionalidades del sistema.

Código	Descripción del requisito funcional	Prioridad
RF1	Insertar un usuario	Media
RF2	Modificar un usuario	Baja
RF3	Eliminar un usuario	Baja
RF4	Mostrar listado de usuario	Baja
RF5	Realizar autenticación de usuario	Alta
RF6	Insertar un trabajador	Media
RF7	Modificar un trabajador	Baja
RF8	Eliminar un trabajador	Baja
RF9	Mostrar listado de trabajadores	Baja
RF10	Insertar un cargo	Media
RF11	Modificar un cargo	Baja
RF12	Eliminar un cargo	Baja
RF13	Mostrar listado de cargos	Baja
RF14	Insertar una empresa prestadora de servicios	Media
RF15	Modificar una empresa prestadora de servicios	Baja
RF16	Eliminar una empresa prestadora de servicios	Baja
RF17	Mostrar listado de una empresa prestadora de	Baja

	servicios	
RF18	Insertar un cuentapropista	Media
RF19	Modificar un cuentapropista	Baja
RF20	Eliminar un cuentapropista	Baja
RF21	Mostrar listado de cuentapropistas	Baja
RF22	Insertar un servicio	Alta
RF23	Modificar un servicio	Baja
RF24	Eliminar un servicio	Baja
RF25	Mostrar listado de servicios	Alta
RF26	Registrar una asignación de presupuesto	Alta
RF27	Modificar una asignación de presupuesto	Alta
RF28	Eliminar una asignación de presupuesto	Alta
RF29	Mostrar listado de asignaciones de presupuesto	Alta
RF30	Registrar una dieta solicitada	Alta
RF31	Modificar una dieta	Alta
RF32	Eliminar una dieta	Alta
RF33	Cerrar una dieta	Alta
RF34	Mostrar listado de dietas abiertas	Alta
RF35	Mostrar listado de dietas cerradas	Alta
RF36	Ver detalles de una dieta	Alta
RF37	Realizar un pago de servicio a un cuentapropista	Alta
RF38	Realizar una pago de servicio a una empresa	Alta

RF39	Mostrar listado de pagos realizados a un cuentapropista	Alta
RF40	Mostrar listado de pagos realizados a una empresa	Alta
RF41	Ver detalles de un pago de servicio realizado	Alta
RF42	Mostrar listado de dietas del presupuesto activo	Alta
RF43	Mostrar pagos del presupuesto activo pagos realizados a una empresa	Alta
RF44	Mostrar pagos del presupuesto activo pagos realizados a un cuentapropista	Alta
RF45	Exportar historial de dietas solicitadas	Alta
RF46	Exportar historial de pagos realizados a una empresa	Alta
RF47	Exportar historial de pagos realizados a un cuentapropista	Alta

2.3.1 Características del sistema.

Tabla 2.3: Características del sistema.

	Características del sistema
	Usabilidad
RNF1	**Facilidad de uso por parte de los usuarios: La aplicación debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.**

RNF2	**Especificación de la terminología utilizada:** La aplicación debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo**.**
RNF3	Emplear perfiles de usuario: Diferenciar las interfaces y opciones para los usuarios que accedan al sistema según los diferentes roles que estos tengan dentro del sistema.
	Fiabilidad
RNF4	**Seguridad de las bases de datos:** La seguridad de la base de datos está a nivel de usuarios autenticados, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos, trayendo consigo además la protección de la información.
RNF5	**Políticas de seguridad por usuario y rol:** El sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.
	Requerimientos de Software
RNF6	Servidor de base de datos con PostgreSQL
	Documentación
RNF7	**Manual de usuario:** La aplicación debe contar con un manual de usuario
RNF8	**Documentación actualizada del grupo de desarrollo:** Se precisa que la documentación del sistema este actualizada en todos los aspectos, fases de trabajo y ciclos de vida del proyecto propiciando así un respaldo legal.

2.4 Historias de usuarios (HU)

Las HU, son la técnica utilizada en XP para detallar los requisitos del software. Son el

resultado directo del intercambio entre los usuarios y desarrolladores a través de reuniones donde las conocidas tormenta de ideas arrojan no solo los requerimientos, sino también las posibles soluciones; representan una forma rápida de administrar las necesidades de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para gestionarlos, debido a que un requerimiento de software es descrito de forma concreta y sencilla utilizando el lenguaje común del usuario. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Tabla 2.4. HU Gestionar presupuesto.

Historia de Usuario	
Número: 5	**Usuario:** Especialista.
Nombre de la Historia: Gestionar presupuesto.	
Referencia: Ítems RF26, RF27, RF28, RF29	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista registrar una nueva asignación de presupuesto, además de modificarla, eliminarla y obtener un lista.	
Observaciones: Confirmado con el cliente.	

Tabla 2.5. HU Gestionar peticiones de dietas

Historia de Usuario	
Número: 6	**Usuario:** Especialista.
Nombre de la Historia: Gestionar peticiones de dietas.	

Referencia: Ítems RF30, RF31, RF32, RF34	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista registrar una nueva dieta solicitada, además de modificarla, eliminarla y obtener un lista.	
Observaciones: Confirmado con el cliente.	

Tabla 2.6 HU Gestionar pagos realizados

Historia de Usuario	
Número: 11	**Usuario:** Especialista.
Nombre de la Historia: Gestionar pagos realizados.	
Referencia: Ítems RF38, RF39, RF40, RF41, RF42, RF43	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** media
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista insertar un pago de servicio a un cuentapropista, insertar un pago de servicio a una empresa, mostrar listados de pago a un cuentapropista, mostrar listados de pago a una empresa, ver detalles de un pago de servicio realizado y anular un pago de servicio realizado.	
Observaciones: Confirmado con el cliente.	

Para ver las Historias de Usuarios ver: Anexo

2.5 Planificación de entregas

En esta parte se establece la prioridad de cada HU así como una estimación del esfuerzo necesario de cada una de ellas con el fin de determinar un cronograma de entregas. Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen utilizando como medida, el punto. Un punto, equivale a una semana ideal de programación (5 días). Las historias generalmente valen de 1 a 3 puntos. Por otra parte, se mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

2.6 Estimación de esfuerzo por HU

Tabla 2.7 Estimación de esfuerzo por HU

Historias de Usuario	Puntos de Estimación (semanas)
Gestionar usuario	1
Realizar la autenticación	0.5
Gestionar trabajador	0.5
Gestionar cargos	1
Gestionar empresas	0.5
Gestionar cuentapropistas	1
Gestionar servicios	0.5

Gestionar presupuesto	1
Gestionar peticiones de dietas	0.5
Gestionar dietas cerradas	1
Gestionar pagos realizados	1
Gestionar reportes del presupuesto activo	1
Exportar historiales	0.5

2.6.1 Planificación de Iteraciones

A partir de las HU antes expuestas y la estimación del esfuerzo propuesto para la realización de las mismas, se procede a realizar la planificación de la etapa de implementación del sistema, apoyándose en el tiempo e intentando concentrar las funcionalidades relacionadas en una misma iteración. En este plan se establece cuántas iteraciones serán necesarias realizar sobre el sistema para su terminación, explican de forma detallada a continuación:

2.6.2 Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto guiado por la metodología de desarrollo de software XP, se crea el plan de duración de cada una de las iteraciones que se llevarán a cabo durante el desarrollo del mismo. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas.

Tabla 2.8 Tabla de duración de las iteraciones

Iteración	**Descripción de la Iteración**	**Historias de Usuarios**	**Duración total semanas**

1ra Iteración	Esta iteración tiene como objetivo la entrega de una primera parte, en la cual se encuentran funcionalidades importantes para el sistema	Gestionar usuario. Realizar la autenticación. Gestionar trabajador. Gestionar cargos. Gestionar presupuesto. Gestionar peticiones de dieta. Gestionar dietas cerradas.	5.5
2da Iteración	En esta iteración se entregara lo que resta del sistema.	Gestionar empresas. Gestionar cuentapropistas. Gestionar servicios. Gestionar pagos realizados. Exportar Historiales Gestionar reportes del presupuesto activo	5

2.7 Clases, Responsabilidades y Colaboración

Las tarjetas clases, responsabilidades y colaboración (CRC), se realizan para facilitar la comunicación y documentar los resultados. Permiten una total participación y contribución del equipo de desarrollo en el diseño. Cada tarjeta CRC representa clases, donde el nombre de cada clase se ubica en forma de título en la parte superior de la tarjeta; sus atributos y responsabilidades más significativas se colocan a la izquierda y las clases implicadas con cada responsabilidad a la derecha, en la misma línea de su requerimiento correspondiente.

El uso de las tarjetas C.R.C permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Las tarjetas C.R.C representan objetos; responsabilidad. Esta nueva técnica de diseño es adoptada como alternativa a los diagramas UML de las clases.

A continuación se muestra una de las tarjetas CRC teniendo en cuenta las

funcionalidades principales a integrar en la aplicación.

Tabla 2.9 Tarjeta CRC: Control_Tipo_Dieta

Nombre de la clase: Control_Tipo_Dieta	
Descripción: Esta clase es la encargada de realizar las operaciones de gestión de tipos de dietas.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Devuelve un tipo de dieta por su identificador	data_context
Devuelve un tipo de dieta por su nombre	data_context
Lista de todos los tipos de dietas	data_context

Tabla 2.10 Tarjeta CRC: Control Asignación

Nombre de la clase: Control Asignación	
Descripción: Esta clase es la encargada de realizar las operaciones sobre las asignaciones.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica si existe una asignación activa	data_context
Devuelve una asignación por su identificador	data_context
Devuelve la asignación activa	data_context
Insertar una asignación	data_context
Eliminar una asignación	data_context
Listado de todas las asignaciones	data_context
Devuelve monto gastado en dietas	data_context
Devuelve el monto gastado en pagos	data_context

Ver tarjeta CRC Anexo

2.8 Modelo de Datos

A continuación se muestra el modelo lógico de la base de datos, mediante el cual se ilustra cómo está diseñada la base de datos de la aplicación, relaciones entre tablas para una mejor comprensión de la aplicación.

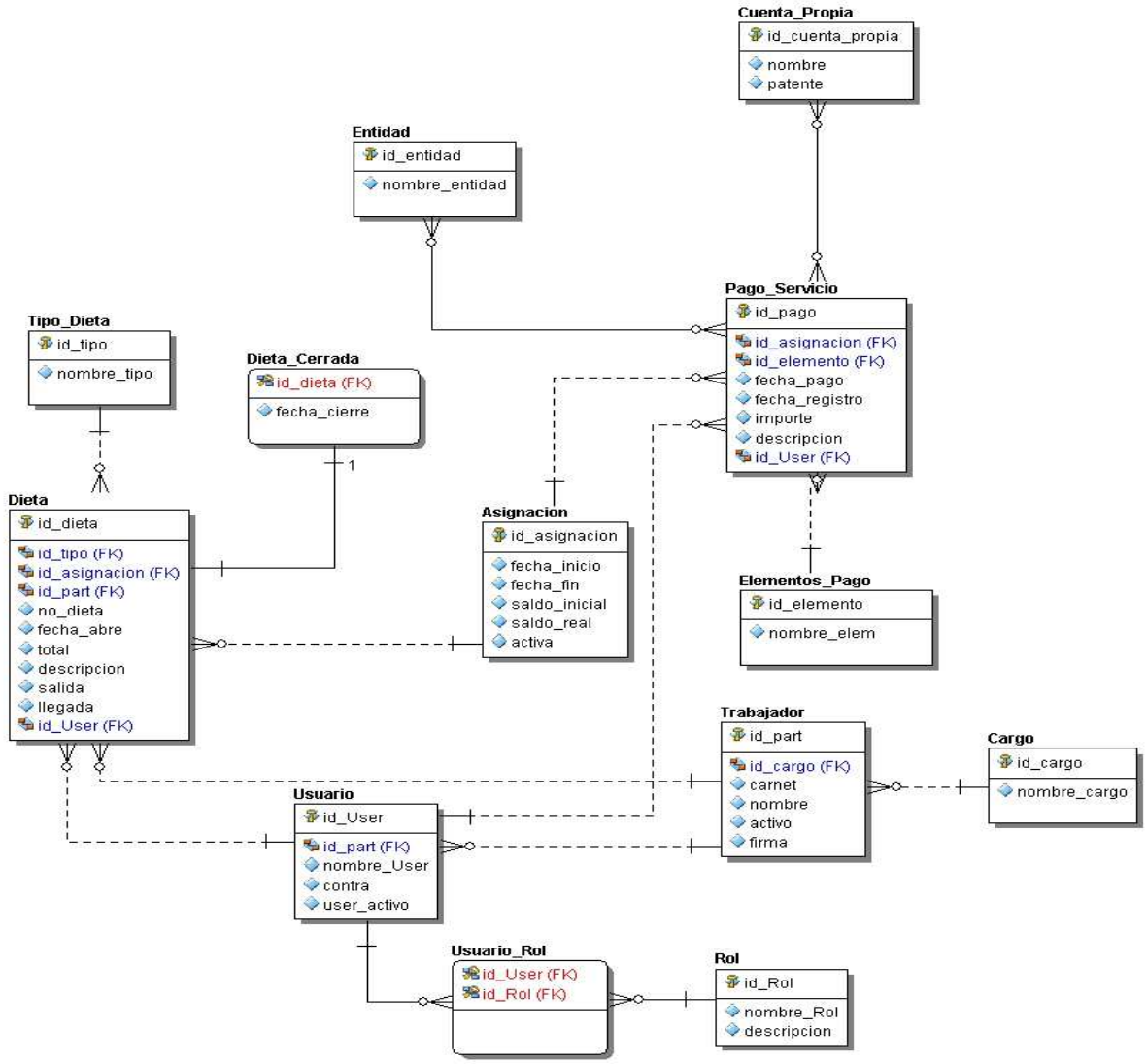


Figura 1.2 Modelos de datos

2.9 Conclusiones del capítulo

Al finalizar este capítulo se ha ejecutado la fase de planeación y diseño, se han planteado las historias de usuarios. Se realizó el plan de iteraciones para cada una de estas historias aplicando una estimación de esfuerzo de las mismas. Se presentaron las clases que se utilizarán en el desarrollo de la aplicación a través de las tarjetas CRC, además del modelo de datos de la aplicación.

Capítulo 3. Desarrollo y Pruebas

3.1 Introducción

En este capítulo se desplegará la fase de desarrollo y pruebas conforme a lo que establece la metodología XP. Se describen cada una de las tareas confeccionadas para cumplir con el desarrollo de cada una de las HU definidas. Se mostrarán las pruebas de aceptación elaboradas para comprobar que la aplicación funcione correctamente.

3.2 Desarrollo de Iteraciones

Durante la fase planificación y diseño fueron detalladas las historias de usuario correspondientes a cada una de las iteraciones a desarrollar, teniendo en cuenta las prioridades y restricciones de tiempo, previstas por el cliente.

3.2.1 Tareas de ingeniería por HU

Dentro del contenido de este plan, las HU se descomponen en tareas de programación o de ingeniería, y a su vez, estas son asignadas al equipo de desarrollo para su implementación. Las tareas no tienen que ser entendidas necesariamente por el cliente, sólo son utilizadas por los miembros del equipo de desarrollo, por lo que pueden ser escritas en lenguaje técnico.

A continuación se presentan las Tareas de Ingeniería agrupadas por las respectivas historias de usuario a las que pertenecen.

Tabla 3.1. Tarea# 14: Registrar una nueva asignación de presupuesto

Tarea de Ingeniería	
Número de la tarea: 14	**Número de Historia:** 5
Nombre de la Tarea: Registrar una nueva asignación de presupuesto	

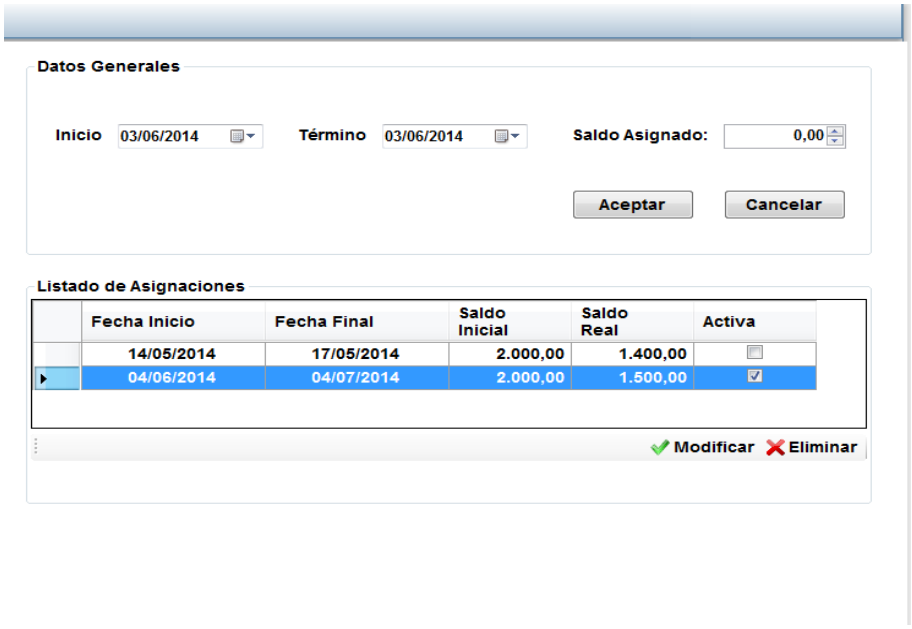
Tipo de Tarea: Desarrollo	
Fecha de Inicio: 04/marzo/2014	**Fecha Fin:** 06/marzo/2014
Programador Responsable: Rafael Rodríguez Rigñack	
Descripción: Se muestran los campos a llenar y se crea una nueva asignación de presupuesto.	
Prototipo de Interfaz 	

Tabla 3.2. Tarea# 18: Registrar una nueva dieta

Tarea de Ingeniería	
Número de la tarea: 18	**Número de Historia:** 6
Nombre de la Tarea: Registrar una nueva dieta	

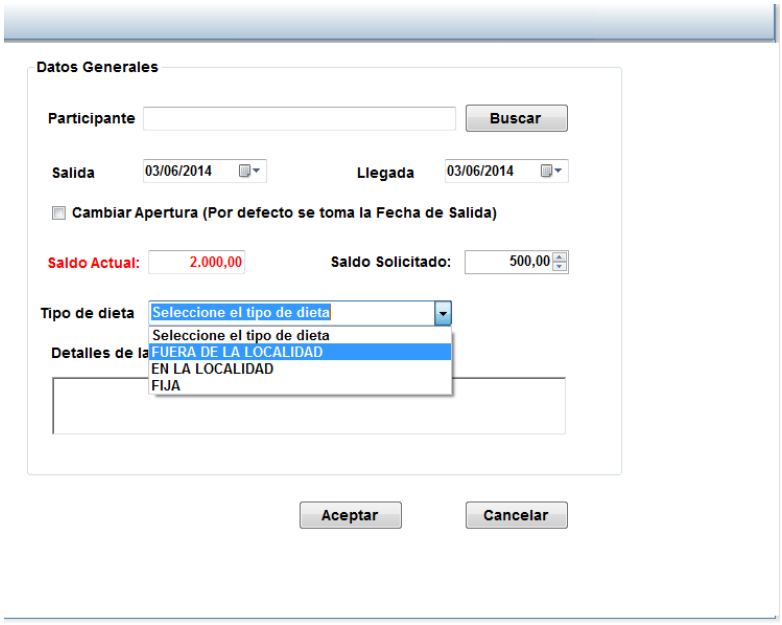
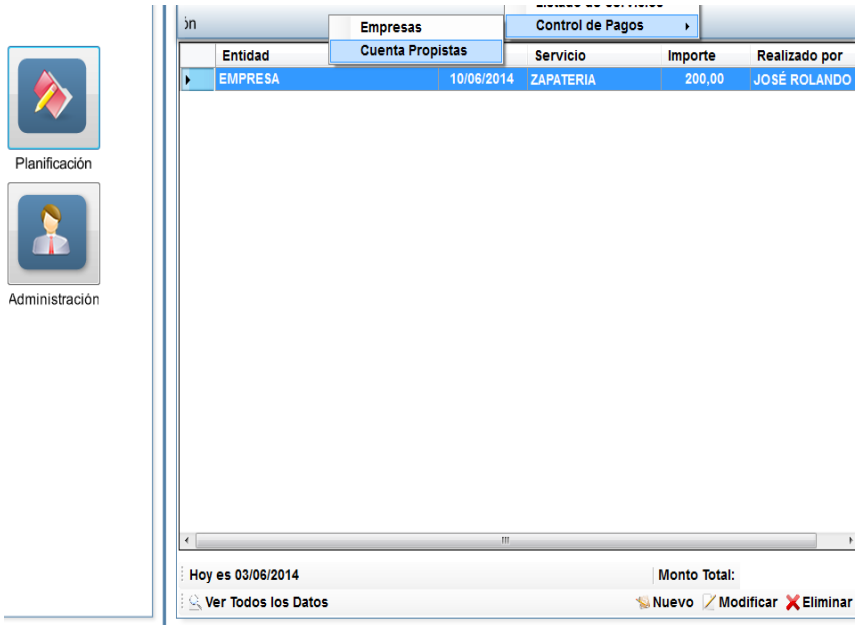
Tipo de Tarea: Desarrollo	
Fecha de Inicio: 17/marzo/2014	**Fecha Fin:** 18/marzo/2014
Programador Responsable: Rafael Rodríguez Rigñack	
Descripción: Muestra los campos para introducir los datos y crea una nueva dieta.	
Prototipo de Interfaz 	

Tabla 3.3. Tarea# 38: Insertar un pago de servicio a un cuentapropista

Tarea de Ingeniería	
Número de la tarea: 38	**Número de Historia:** 11
Nombre de la Tarea: Insertar un pago de servicio a un cuentapropista	
Tipo de Tarea: Desarrollo	
Fecha de Inicio: 28/abril/2014	**Fecha Fin:**

	30/abril/2014
Programador Responsable: Rafael Rodríguez Rigñack	
Descripción: Muestra los campos a llenar y permite insertar un pago de servicio a un cuentapropista.	
Prototipo de Interfaz	



Para ver Tareas de Ingeniería ver Anexo

Tabla 3.4 Historias de Usuario abordadas en la primera iteración.

Historias de Usuario	Tiempo de estimación (semanas)	
	Estimación inicial	Real
Gestionar usuario.	1	1
Realizar la autenticación.	0.5	0.5
Gestionar trabajador.	0.5	1

Gestionar cargos.	1	1
Gestionar presupuesto.	1	1
Gestionar peticiones de dieta.	0.5	1
Gestionar dietas cerradas.	1	1

Tabla 3.5 Historias de Usuario abordadas en la segunda iteración.

Historias de Usuario	**Tiempo de estimación (semanas)**	
	Estimación inicial	**Real**
Gestionar empresas.	0.5	0.5
Gestionar cuentapropistas.	1	1
Gestionar servicios.	0.5	1
Gestionar pagos realizados.	1	1
Gestionar reportes del presupuesto activo	1	1
Exportar historiales	0.5	0.5

3.3 Pruebas

En la Programación Extrema es esencial el desarrollo de las pruebas, permitiendo probar continuamente el código. Cada vez que se desean implementar las funcionalidades que tendrá el software, XP propone una redacción sencilla de prueba. El proceso de las pruebas permite la obtención un producto con mayor calidad ofreciendo a los programadores una mayor certeza en el trabajo que desempeñan.

En la metodología XP hay dos tipos de pruebas; las unitarias o desarrollo dirigido por pruebas, desarrolladas por los programadores verificando su código de forma automática, y las pruebas de aceptación, las cuáles son evaluadas luego de culminar

una iteración verificando así que se cumplió la funcionalidad requerida por el cliente. Con estas normas se obtiene un código simple y funcional de manera bastante rápida y eficiente.

3.3.1 Desarrollo dirigido por Pruebas

El desarrollo dirigido por pruebas, se enfoca en la implementación orientada a pruebas. El código debe ser probado paso a paso para lograr un resultado, aunque no con lógica para el negocio, pero si funcional. Algunas personas confunden este término con las llamadas “pruebas de caja blanca” las cuáles se les practican a los métodos u operaciones para medir la funcionalidad del mismo, desde el punto de vista de validez del cliente.

Sin embargo, el TDD se aplica antes de comenzar a implementar cada paso de la tarea en desarrollo, asumiendo que la prueba es insatisfactoria desde un inicio. Sólo una vez que se haya cumplido de la forma más sencilla posible la lógica del código a probar se asume como cumplida.

Luego se realiza un proceso conocido como refactorización de código perteneciente a una de las doce prácticas planteadas por la metodología XP, el cual consiste en mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. Es esencia el TDD, se enfoca en la lógica del negocio y las pruebas de caja blanca en la lógica del negocio.

3.3.2 Pruebas de aceptación (PA)

Las pruebas de aceptación en XP, se pueden asociar con las pruebas de caja negra que se aplican en la metodología RUP, sólo que se crean a partir de las historias de usuario y no por un listado de requerimientos. Durante las iteraciones, las HU se traducen a pruebas de aceptación. En ellas se especifican desde la perspectiva del cliente, los escenarios para probar que una historia de usuario ha sido implementada correctamente.

La misma puede tener todas las pruebas de aceptación que necesite para asegurar su

correcto funcionamiento. El objetivo que persiguen estas pruebas, es garantizar que las funcionalidades solicitadas por el cliente han sido realizadas satisfactoriamente.
Para realizar las pruebas de aceptación (PA) el cliente utiliza el siguiente modelo:

A continuación se muestra algunas de las pruebas de aceptación realizadas.

Tabla 3.6. Prueba de aceptación para la HU Gestionar Presupuesto.

Prueba de Aceptación: Gestionar presupuesto
HU: 5
Nombre: Prueba para gestionar presupuesto
Descripción: Se realiza una prueba para verificar que la aplicación inserte un nuevo presupuesto, modifique, elimine y permita mostrar un listado de presupuesto existente.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo presupuesto, luego presiona el botón aceptar e inserta un nuevo presupuesto, permitiendo modificarlo, eliminarlo y mostrar una lista de presupuesto.
Resultado: Se inserta correctamente el nuevo presupuesto, modifica, elimina y muestra una lista de presupuesto.
Evaluación de la prueba: Aceptada.

Tabla 3.7. Prueba de aceptación para la HU Gestionar dieta solicitada.

Prueba de Aceptación: Gestionar dieta solicitada
HU: 6
Nombre: Prueba para gestionar nueva dieta solicitada
Descripción: Se realiza una prueba para verificar que la aplicación inserte una nueva dieta, modifique, elimine y permita mostrar un listado de dietas solicitadas existentes.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar una nueva dieta, luego presiona el botón aceptar e inserta una nueva dieta, permitiendo modificarla, eliminarla y mostrar una lista de dietas.
Resultado: Se inserta correctamente el nuevo presupuesto, modifica, elimina y muestra una lista de presupuesto.
Evaluación de la prueba: Aceptada.

Pruebas de aceptación: Ver anexo

3.4 Conclusiones del capítulo

En este capítulo se llevó a cabo la fase de desarrollo y pruebas. Se realizó el desarrollo de las iteraciones a partir de la distribución de tareas por historias de usuarios y se le hicieron las pruebas de aceptación a estas para verificar las funcionalidades de la aplicación. Todas las pruebas fueron aceptadas por el cliente.

Capítulo 4. Estudio de factibilidad económica

4.1 Introducción

En este capítulo se realiza el estudio de factibilidad del producto. El cual es significativo, pues se tienen en cuenta los costos a incurrir, deduciéndose si el proyecto realizado será factible o no llevarlo a cabo. Hay muchas formas de calcular el costo, pero para nuestro caso se utilizará la Metodología Costo Efectividad. En la misma se desglosan elementos para identificar los costos y beneficios del proyecto, los efectos económicos y la ficha de costo. [17]

- El costo que involucra la implementación de la solución informática, adquisición y puesta en marcha del sistema hardware / software y los costos de operación asociados.
- La efectividad que se entiende como capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo por el cual se ideó, es decir, un proyecto será más o menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad del cumplimiento del objetivo).

4.2 Efectos Económicos

- Efectos directos
- Efectos indirectos
- Efectos externos
- Intangibles

4.2.1 Efectos directos

POSITIVOS:

Se optimizará la realización de los modelos de pagos por concepto de viáticos y pagos menores.

Disminuirá el riesgo de deterioro de los registros en formato duro, evitando pérdida de información.

Se mejorará la gestión de la información del presupuesto de la ONEI.

NEGATIVOS:

Para usar la aplicación es imprescindible el uso de un ordenador, aparejado a los gastos que este trae de consumo de corriente eléctrica y mantenimiento.

4.2.2 Efecto indirecto

Los efectos económicos observados que pudiera repercutir sobre otros mercados no son perceptibles, aunque este proyecto no está construido con la finalidad de comercializarse.

4.2.3 Intangibles

En la valoración económica siempre hay elementos perceptibles por una comunidad como perjuicio o beneficio, pero al momento de ponderar en unidades monetarias esto resulta difícil o prácticamente imposible. A fin de medir con precisión los efectos, deberán considerarse dos situaciones:

4.2.3.1 Situación sin proyecto

La gestión de información referente al presupuesto y viáticos, se registran en hojas de cálculo Excel y en formato duro, por lo que el proceso de gestión de esta información tiende a ser exhausto y complicado, además la constante revisión y manipulación promueve el deterioro de los registros y en algunos casos hasta pérdida de información.

4.2.3.2 Situación con proyecto

Con la realización de la aplicación para la gestión de estas informaciones se ha logrado disminuir el tiempo de ejecución de esta tarea, mejor organización y disponibilidad de

los registros contables, garantizando el correcto funcionamiento de las operaciones sobre ellos.

4.3 Beneficios Y Costos Intangibles en el proyecto

Costos:

Resistencia al cambio.

Beneficios:

Mayor comodidad para los usuarios.

Mejora en la calidad de la información.

Menor tiempo empleado en la introducción de los datos.

Menor usabilidad a los registros en formato duro, lo que evita su deterioro.

4.4 Ficha de costo

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar una ficha de costo de un producto informático. Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional.

4.4.1 Costos en Moneda Libremente Convertible

Costos Directos.

1. Compra de equipos de cómputo: No procede.
2. Alquiler de equipos de cómputo: No procede.
3. Compra de licencia de Software: No procede.
4. Depreciación de equipos: $ 50.00.
5. Materiales directos: No procede.

Total: $ 50.00.

Costos Indirectos.

1. Formación del personal que elabora el proyecto: No procede.
2. Gastos en llamadas telefónicas: No procede.

3. Gastos para el mantenimiento del centro: No procede.

4. Capacitación: No procede.

5. Gastos en representación: No procede.

Total: $0.00.

Gastos de distribución y venta.

1. Participación en ferias o exposiciones: No procede.

2. Gastos en transportación: No procede.

3. Compra de materiales de propagandas: No procede.

Total: $0.00.

4.4.2 Costos en Moneda Nacional

Costos Directos.

1. Salario del personal que laborará en el proyecto: $100.00.

2. El 12% del total de gastos por salarios se dedica a la seguridad social: No procede.

3. El 0.09% de salario total, por concepto de vacaciones a acumular: No procede.

4. Gasto por consumo de energía eléctrica: $ 32.50.

5. Gastos en llamadas telefónicas: No procede.

6. Gastos administrativos: No procede.

Total: $ 132.50.

Como se hizo referencia anteriormente, la técnica seleccionada para evaluar la factibilidad del proyecto es la Metodología Costo-Efectividad. El punto de equilibrio aplicable a proyectos donde los beneficios tangibles no son evidentes el análisis se basa exclusivamente en los costos. Para esta técnica es imprescindible definir una variable discreta que haga variar los costos. Teniendo en cuenta que el costo para este proyecto es despreciable, tomaremos como costo el tiempo en minutos empleado desde que se introducen los datos hasta el tiempo en que se muestran los resultados y el gráfico de comportamiento.

Valores de la variable (Solución manual):

Gestionar peticiones de dietas (20 min).

Gestionar servicios (30 min).

Gestionar pagos realizados (20 min).

Gestionar presupuesto (30 min).

Valores de la variable (Solución con el software):

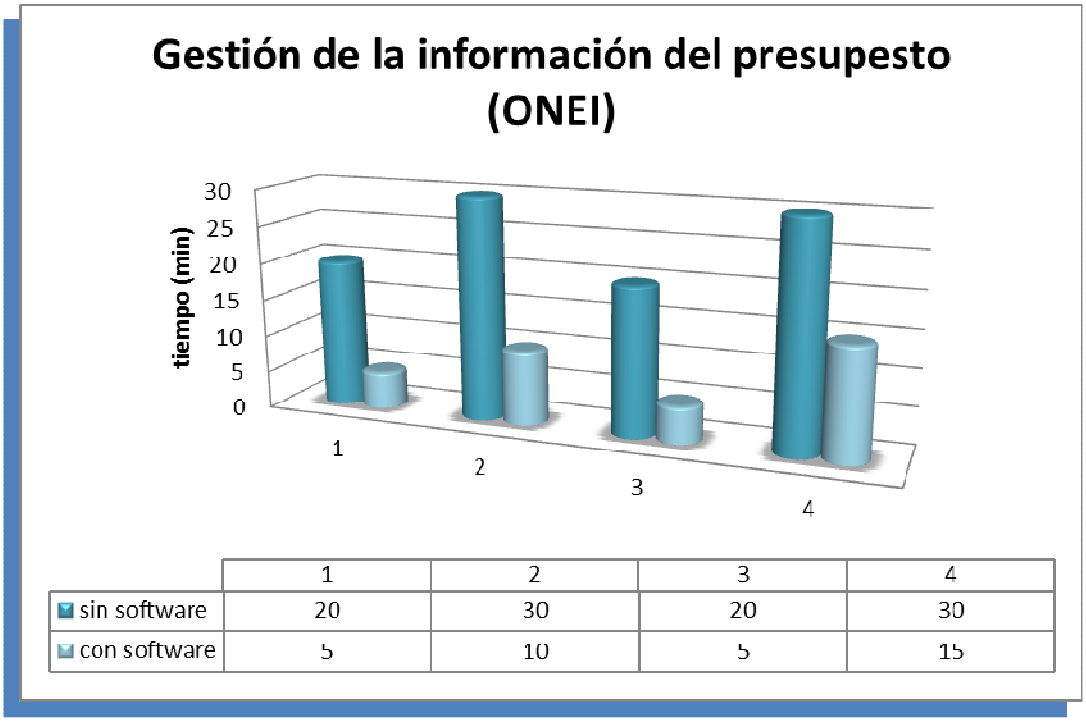
Gestionar peticiones de dietas (5 min).

Gestionar servicios (10 min).

Gestionar pagos realizados (5min).

Gestionar presupuesto (15 min).

Se muestra un gráfico a continuación que muestra la solución de las actividades sin proyecto comparado con la solución con proyecto. Tiempo de la realización de las actividades.



	1	2	3	4
sin software	20	30	20	30
con software	5	10	5	15

Figura 4.1 Grafico de tiempo de realización del proceso

4.5 Conclusiones del capítulo

En este capítulo se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad (Beneficio), se analizó los efectos económicos, los beneficios y costos intangibles, así como se calculó el costo de ejecución del proyecto mediante la ficha de costo arrojando como resultado $50.00 CUC y $135.50 CUP. Demostrándose la factibilidad económica del proyecto.

Conclusiones Generales.

Al concluir la investigación se han cumplido el objetivo y las tareas propuestas:

Se elaboró el marco teórico metodológico que fundamenta la investigación, esto permitió que quedara identificada la situación problémica existente y se establecieran las bases para el posterior diseño e implementación de la aplicación.

Se estableció el estado del arte de la investigación, resaltándose que las herramientas existentes no responden a las necesidades reflejadas por el entorno, por lo que se decide el desarrollo de una aplicación informática orientada a la situación actual.

Se realizó un levantamiento en el que se identificaron los principales elementos involucrados en la gestión del presupuesto destinado al pago de servicios y viáticos en la ONEI, determinándose las principales funcionalidades a implementar en la propuesta de solución.

Con el desarrollo de la aplicación informática para la gestión del presupuesto destinado al pago de servicios y viáticos en la ONEI se dió cumplimiento al objetivo general de la presente investigación, pues se obtuvo como resultado una herramienta informática que potencia el proceso de recogida y posterior procesamiento de información relevante para el Departamento de Economía.

Se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad Beneficio, analizándose los efectos económicos, los beneficios y costos intangibles. Se calculó además el costo de ejecución del proyecto mediante la ficha de costo, arrojando como resultado $50.00 CUC y $135.50 CUP.

Recomendaciones.

El autor de la presente investigación ha considerado pertinente ofrecer las siguientes recomendaciones:

- Realizar la implantación del software en la ONEI.
- Realizar encuestas de satisfacción a los usuarios finales que permitan realizarle nuevas versiones a la aplicación informática teniendo en cuenta las recomendaciones hechas por ellos.
- Generalizar la aplicación construida en las diversas estructuras de la ONEI, realizándose un estudio previo con las consiguientes adaptaciones de entorno.

Referencias Bibliográficas.

1. Sistema Web para la Gestión de Viajes y Viáticos. http://www.gram-asociados.com/Gestion_de_Viajes.htm?Lf2tTUIjobKy%FE (16/2/2014).
2. Trigo, Vicente. *Historia y evolución de los lenguajes de programación, Marzo 2000.*
3. C++. http://www.faqs.org/docs/artu/C++_evolution.html. (18/03/2014)
4. Sierra, Alejandro Aguilar. Programación Extrema y Software Libre. 2009.
5. González, José Antonio. *El lenguaje de programación c#.*
6. Arid, Sofía*: Desarrollo del módulo Peso específico de los suelos para el sistema de gestión de informes de ensayos.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Minero Metalúrgico "José Antonio Núñez Jiménez", Moa, Holguín, junio 2013.
7. Oracle (SGBD) http://www.oracle.com/es. (20/03/2014).
8. MySQL (SGBD) http://dev.mysql.com/doc. (23/03/2014)
9. SqlServer (SGBD) http://www.dotnet4all.com/snippets/2008/04/factsheet-for-sql-server. (15/03/2014)
10. PostgreSql (SGBD) http://www.postgresql-es.org/. (16/03/2014)
11. Canós, José H. *Metodologías Ágiles para el desarrollo de software.*
12. Martínez, Alejandro. *Guía a Rational Unified Process,2001.*
13. XP Agile universe: www.agileuniverse.com (19/04/2014).
14. Arquitectura en capas: http://bulma.net/body.phtml. (25/03/2014)
15. Microsoft Visual studio: http://www.microsoft.com/spain/visualstudio (22/03/2014).
16. Salazar, Ricardo*: Sistema de Gestión de Capacitación ECRIN.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior

Minero Metalúrgico “José Antonio Núñez Jiménez”, Moa, Holguín, junio 2013.

17. Costo-Efectividad Disponible en: www.crid.or.cr/digitalizacion/pdf/spa/doc9321-02.pdf (23/5/2014).
18. Sistema de Control de Viáticos : http://www.reset.com.mx (17/06/2014)
19. Viáticos-Portal http://www.edicomgroup.com (02/02/2014)

Bibliografía.

[GONZALEZ], Gonzales*, Antonio. El lenguaje de programación C#*.

[HERNANDEZ, 2005], Hernández*, Metodología de la Investigación ¿Cómo hacer una tesis*?, 2005.

[MENDOZA, 2004], Mendoza*, Metodología de desarrollo de software*, 2004.

[MATO, 1999], Mato*, Diseño de una Base de datos,* 1999.

[QUINONES, 2000], Quiñones*, Introducción a PostrgreSQL*, 2000.

MC#. *CSharp*: Manual de Interfaces para CSharp. Universidad de Perú.2003.

Schenone, Marcelo: *Diseño de una metodología ágil de desarrollo de software.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Universidad de Buenos Aires, abril 2004.

Méndez, Osmani*: Sistema automatizado para la gestión energética en el ISMM.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Minero Metalúrgico “José Antonio Núñez Jiménez”, Moa, Holguín, junio 2008.

Leyva, Ernesto*: Sistema de gestión de información para la evaluación del control interno.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Minero Metalúrgico “José Antonio Núñez Jiménez”, Moa, Holguín, junio 2011.

Glosario de términos.

Aplicación: Es el programa que el usuario activa para trabajar en el ordenador. Existen muchos programas de ordenador que pueden clasificarse como aplicación. Generalmente se les conoce como Software.

API: Una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Dietas de viaje: Pago por concepto de viaje que se le atribuye al personal que realiza labores de trabajo fuera de la entidad.

Gestión de la Información: (GI), es un conjunto de procesos por los cuales se controla el ciclo de vida de la información, desde su obtención - por creación o captura, hasta su disposición final - archivada o eliminada. Los procesos también comprenden la extracción, combinación, depuración y distribución de la información a los interesados. Los objetivos de la Gestión de la Información es garantizar la integridad, disponibilidad y confidencialidad de la información.

HU: Historia de Usuarios son las tablas creadas para la realización de las interfaces del sistema.

IU: Interfaces de Usuarios

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas o sistemas operativos.

Código abierto : Licencias Open Source (Código Abierto) es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

Programación Extrema(XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo.

Pagos menores: son aquellos que se pagan por servicios recibidos por parte de la entidad de terceros.

RUP: El Proceso Unificado Rational (RUP) es una metodología de desarrollo para la programación orientada a objetos. Según Rational (diseñadores de Rose Rational y el Idioma Modelado Unificado UML), RUP está como un mentor en línea que mantiene pautas, plantillas, y ejemplos de todos los aspectos y fases de desarrollo del programa.

Software Libre: Es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, aunque conserve su carácter de libre, puede ser vendido comercialmente.

Tecnologías de Informática y las comunicaciones (TIC): Se denominan tecnologías de la información y la comunicación al conjunto de tecnologías que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética. Las TIC incluyen la electrónica como tecnología base que soporta el desarrollo de las telecomunicaciones, la informática y el audiovisual.

Anexo Historias de usuario.

Tabla 1. HU Realizar autenticación.

Historia de Usuario	
Número: 2	**Usuario:** Especialista.
Nombre de la Historia: Realizar la autenticación.	
Referencia: Ítems RF5	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** media
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: El usuario introduce sus datos y procede a la autenticación.	
Observaciones: Confirmado con el cliente.	

Tabla 2 HU Gestionar usuario

Historia de Usuario	
Número: 1	**Usuario:** Especialista.
Nombre de la Historia: Gestionar usuario.	
Referencia: Ítems RF1, RF2, RF3, RF4.	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista insertar, modificar, eliminar un usuario, además de mostrar un listado de usuarios.	
Observaciones: Confirmado con el cliente.	

Tabla 3 HU Gestionar trabajador.

Historia de Usuario	
Número: 3	**Usuario:** Especialista.
Nombre de la Historia: Gestionar trabajador.	
Referencia: Ítems RF6, RF7, RF8, RF9	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista insertar, modificar, eliminar un trabajador, además mostrar un listado de trabajadores.	
Observaciones: Confirmado con el cliente.	

Tabla 4 HU Gestionar cargos.

Historia de Usuario	
Número: 4	**Usuario:** Especialista.
Nombre de la Historia: Gestionar cargos.	
Referencia: Ítems RF10, RF11, RF12, RF13	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista insertar, modificar, eliminar un cargo, además de listar los cargos.	
Observaciones: Confirmado con el cliente.	

Tabla 5 HU Gestionar dietas cerradas

Historia de Usuario	
Número: 7	**Usuario:** Especialista.
Nombre de la Historia: Gestionar dietas cerradas.	
Referencia: Ítems RF33,RF35	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista cerrar una dieta, mostrar un listado de dietas cerradas y ver detalles de una dieta cerrada.	
Observaciones: Confirmado con el cliente.	

Tabla 6 HU Gestionar empresas

Historia de Usuario	
Número: 8	**Usuario:** Especialista.
Nombre de la Historia: Gestionar empresas.	
Referencia: Ítems RF14, RF15, RF16, RF17	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista insertar, modificar, eliminar una empresa, además de mostrar un listado de las empresas.	
Observaciones: Confirmado con el cliente.	

Tabla 8. HU Gestionar cuentapropistas

Historia de Usuario	
Número: 9	**Usuario:** Especialista.
Nombre de la Historia: Gestionar cuentapropistas.	
Referencia: Ítems RF18, RF19, RF20, RF21	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista insertar, modificar, eliminar un cuentapropista, además de mostrar un listado.	
Observaciones: Confirmado con el cliente.	

Tabla 9. HU Gestionar servicios

Historia de Usuario	
Número: 10	**Usuario:** Especialista.
Nombre de la Historia: Gestionar servicios.	
Referencia: Ítems RF22, RF23, RF24, RF25	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** media
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista insertar, modificar, eliminar un servicio, además de mostrar un listado de servicios.	
Observaciones: Confirmado con el cliente.	

Tabla 10. HU Gestionar reportes del presupuesto activo

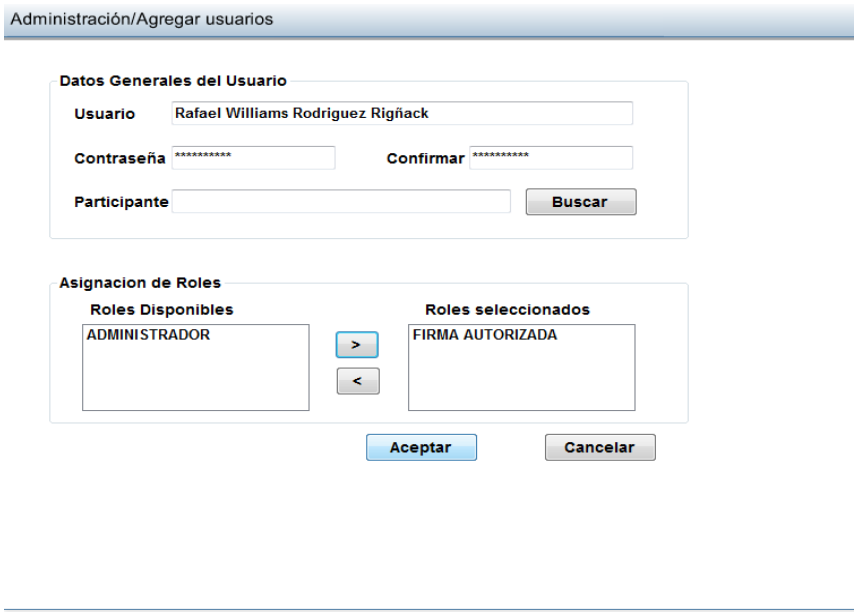
Historia de Usuario	
Número: 12	**Usuario:** Especialista.
Nombre de la Historia: Gestionar reportes del presupuesto activo	
Referencia: Ítems RF44, RF45, RF46	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** media
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista mostrar reportes de dietas del presupuesto activo, mostrar reporte de pagos de servicio a un cuentapropista del presupuesto activo y mostrar reporte de pago de servicio a una empresa del presupuesto activo.	
Observaciones: Confirmado con el cliente.	

Tabla 11. HU Exportar historiales

Historia de Usuario	
Número: 13	**Usuario:** Especialista.
Nombre de la Historia: Exportar historiales	
Referencia: Ítems RF47, RF48, RF49	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** media
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Rafael Rodríguez Rigñack.	
Descripción: Permite al especialista exportar historial de dietas solicitadas, pagos de servicios a una empresa y pagos de servicios a un cuentapropista.	
Observaciones: Confirmado con el cliente.	

Anexo Tareas de Ingeniería

Tabla 12 Tarea# 1: Insertar usuario

Tarea de Ingeniería	
Número de la tarea: 1	**Número de Historia:**1
Nombre de la Tarea: Insertar usuario	
Tipo de Tarea: Desarrollo	
Fecha de Inicio: 06/febrero/2014	**Fecha Fin: 07/febrero/2014**
Programador Responsable: Rafael Rodríguez Rigñack	
Descripción: Se muestra los campos a llenar y se inserta un usuario.	
Prototipo de Interfaz 	

Nota: las tareas de ingeniería que faltan se encuentran disponibles en el expediente del proyecto.

Anexo Pruebas de aceptación

Tabla 13. Prueba de aceptación para la HU Realizar autenticación.

Prueba de Aceptación: Realizar autenticación
HU: 2
Nombre: Prueba para realizar la autenticación.
Descripción: Se realiza una prueba para verificar la autenticación de usuarios en la aplicación.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra una ventana donde se muestran los campos que se requieren para la autenticación de usuarios.
Resultado: El usuario accede correctamente a la aplicación.
Evaluación de la prueba: Aceptada.

Tabla 14 Prueba de aceptación para la HU Gestionar usuarios.

Prueba de Aceptación: Gestionar usuarios
HU: 1
Nombre: Prueba para gestionar usuarios
Descripción: Se realiza una prueba para verificar que la aplicación inserte, modifique, elimine y permita mostrar un listado de usuarios existentes.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.

Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo usuario, luego presiona el botón aceptar e inserta un nuevo usuario, permitiendo modificarlo, eliminarlo y mostrar una lista de usuarios.
Resultado: Se inserta correctamente el usuario, modifica, elimina y muestra una lista de usuarios.
Evaluación de la prueba: Aceptada.

Tabla 15. Prueba de aceptación para la HU Gestionar Trabajadores.

Prueba de Aceptación: Gestionar trabajadores
HU: 3
Nombre: Prueba para gestionar trabajadores
Descripción: Se realiza una prueba para verificar que la aplicación inserte, modifique, elimine y permita mostrar un listado de trabajadores existentes.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo trabajador, luego presiona el botón aceptar e inserta un nuevo trabador, permitiendo modificarlo, eliminarlo y mostrar una lista de trabajadores.
Resultado: Se inserta correctamente el usuario, modifica, elimina y muestra una lista de trabajadores.

Evaluación de la prueba: Aceptada.

Tabla 16. Prueba de aceptación para la HU Gestionar Cargos.

Prueba de Aceptación: Gestionar cargos
HU: 4
Nombre: Prueba para gestionar cargos
Descripción: Se realiza una prueba para verificar que la aplicación inserte, modifique, elimine y permita mostrar un listado de cargos existentes.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo cargo, luego presiona el botón aceptar e inserta un nuevo cargo, permitiendo modificarlo, eliminarlo y mostrar una lista de cargos.
Resultado: Se inserta correctamente el nuevo cargo, modifica, elimina y muestra una lista de cargos.
Evaluación de la prueba: Aceptada.

Tabla 17. Prueba de aceptación para la HU Gestionar dietas cerradas.

Prueba de Aceptación: Gestionar dieta cerradas
HU: 7
Nombre: Prueba para gestionar dietas cerradas

Descripción: Se realiza una prueba para verificar que la aplicación cierre una dieta, ver detalles de una dieta cerrada y muestre un listado de dietas cerradas.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para cerrar una dieta, luego presiona el botón aceptar y cierra la dieta, ver detalles y mostrar una lista de dietas cerradas.
Resultado: Se cierra correctamente una dieta, muestra detalles y muestra una lista de dietas cerradas.
Evaluación de la prueba: Aceptada.

Tabla 18. Prueba de aceptación para la HU Gestionar Empresas.

Prueba de Aceptación: Gestionar empresas
HU: 8
Nombre: Prueba para gestionar empresas
Descripción: Se realiza una prueba para verificar que la aplicación inserte una nueva empresa, modifique una empresa, elimine y permita mostrar un listado de empresas existente.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar una nueva empresa, luego presiona el botón aceptar e inserta una nueva empresa, permitiendo modificarla, eliminarlo y mostrar

una lista de empresas.
Resultado: Se inserta correctamente la nueva empresa, modifica, elimina y muestra una lista de empresas.
Evaluación de la prueba: Aceptada.

Tabla 19. Prueba de aceptación para la HU Gestionar Cuentapropistas.

Prueba de Aceptación: Gestionar cuentapropistas
HU: 9
Nombre: Prueba para gestionar cuentapropistas
Descripción: Se realiza una prueba para verificar que la aplicación inserte un nuevo cuentapropista, modifique un cuentapropistas, elimine y permita mostrar un listado de cuentapropistas existentes.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo cuentapropista, luego presiona el botón aceptar e inserta un nuevo cuentapropista, permitiendo modificarlo, eliminarlo y mostrar una lista de cuentapropistas.
Resultado: Se inserta correctamente el nuevo cuentapropista, modifica, elimina y muestra una lista de cuentapropistas.
Evaluación de la prueba: Aceptada.

Tabla 20. Prueba de aceptación para la HU Gestionar servicios.

Prueba de Aceptación: Gestionar servicios
HU: 10
Nombre: Prueba para gestionar servicios
Descripción: Se realiza una prueba para verificar que la aplicación inserte un nuevo servicio, modifique un servicio, elimine y permita mostrar un listado de servicios.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo servicio, luego presiona el botón aceptar e inserta un nuevo servicio, permitiendo modificarlo, eliminarlo y mostrar una lista de servicios.
Resultado: Se inserta correctamente el nuevo servicio, modifica, elimina y muestra una lista de servicios.
Evaluación de la prueba: Aceptada.

Tabla 21. Prueba de aceptación para la HU Gestionar pago de servicios.

Prueba de Aceptación: Gestionar pago de servicio a un cuentapropista
HU: 11
Nombre: Prueba para gestionar un pago de servicio a un cuentapropista
Descripción: Se realiza una prueba para verificar que la aplicación inserte un nuevo pago de servicio a un cuentapropista y permita mostrar un listado de pago de servicio a un cuentapropista.

Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo pago de servicio a un cuentapropista, luego presiona el botón aceptar e inserta un nuevo pago de servicio y mostrar una lista de pago de servicio.
Resultado: Se inserta correctamente el nuevo pago de servicio a un cuentapropista y muestra una lista de pagos de servicio a un cuentapropista.
Evaluación de la prueba: Aceptada.

Tabla 22. Prueba de aceptación para la HU Gestionar pago de servicios.

Prueba de Aceptación: Gestionar pago de servicio a una empresa
HU: 11
Nombre: Prueba para gestionar un pago de servicio a una empresa
Descripción: Se realiza una prueba para verificar que la aplicación inserte un nuevo pago de servicio a una empresa y permita mostrar un listado de pago de servicio a una empresa.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los campos que se requieren para insertar un nuevo pago de servicio a una empresa, luego presiona el botón aceptar e inserta un nuevo pago de servicio y mostrar una lista de pago de servicio a una empresa.
Resultado: Se inserta correctamente el nuevo pago de servicio a una empresa y muestra una lista de pagos de servicio a una empresa.

Evaluación de la prueba: Aceptada.

Tabla 23. Prueba de aceptación para la HU Gestionar pago de servicios.

Prueba de Aceptación: Ver detalles de un pago de servicio.
HU: 11
Nombre: Prueba para ver detalles y anular un pago de servicio
Descripción: Se realiza una prueba para verificar que la aplicación permita ver detalles de un pago de servicio.
Condiciones de ejecución: Se deben llenar los campos que solicita la aplicación.
Entrada / Pasos de ejecución: Se muestra un formulario donde se muestran los pagos de servicios, se selecciona el que se requiere para ver detalles de un pago de servicio, luego presiona el botón aceptar y muestra los detalles del pago servicio.
Resultado: Se muestra correctamente el pago de servicio.

Nota: las pruebas de aceptación que faltan se encuentran disponibles en el expediente del proyecto.

Anexo Tarjetas CRC

Tabla 24 Tarjeta CRC: Control_Area_Admin

Nombre de la clase: Control_Area_Admin	
Descripción: Esta clase es la encargada de realizar las operaciones con las áreas.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica se existe un área por el nombre	data_context
Devuelve un área administrativa por su identificador	data_context
Insertar una nueva Área	data_context
Modificar un Área	data_context
Eliminar un área	data_context
Devuelve un listado de áreas	data_context

Tabla 25 Tarjeta CRC: Control_Cargo

Nombre de la clase: Control_Cargo	
Descripción: Esta clase es la encargada de realizar las operaciones de la gestión de cargos.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica si existe un cargo por el nombre	data_context
Devuelve un cargo por su identificador	data_context
Insertar un nuevo cargo	data_context
Modificar un cargo	data_context
Eliminar un cargo	data_context

Listado de todos los cargos	data_context

Tabla 26 Tarjeta CRC: Control_Cuenta_Propia

Nombre de la clase: Control_Cuenta_Propia	
Descripción: Esta clase es la encargada de realizar las operaciones de la gestión de cuentapropistas.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica si existe un cuentapropista por el nombre	data_context
Verifica existe un cuentapropista por la patente	data_context
Devuelve un cuentapropista por su identificador	data_context
Inserta un cuentapropista	data_context
Modifica un cuentapropista	data_context
Listado de todos los cuentapropista	data_context
Elimina un cuentapropista	data_context

Tabla 27 Tarjeta CRC: Control_Empresa

Nombre de la clase: Control_Empresa	
Descripción: Esta clase es la encargada de realizar las operaciones de la gestión de empresas.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica si existe una empresa por el nombre.	data_context
Devuelve una empresa por su identificador.	data_context

Devuelve una empresa por su nombre.	data_context
Inserta una nueva empresa.	data_context
Modifica una empresa.	data_context
Listado de todas las empresas.	data_context
Elimina una empresa.	data_context

Tabla 28 Tarjeta CRC: Control_Rol

Nombre de la clase: Control_Rol	
Descripción: Esta clase es la encargada de realizar las operaciones sobre los roles.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica si existe un rol por el nombre	data_context
Verifica si existe un rol por su identificador.	data_context
Devuelve un rol por su nombre.	data_context
Inserta un nuevo rol.	data_context
Modifica un rol.	data_context
Listado de todos los roles.	data_context

Tabla 29 Tarjeta CRC: Control_Servicio

Nombre de la clase: Control_Servicio	
Descripción: Esta clase es la encargada de realizar las operaciones de gestión de servicios.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica si existe un pago de servicio por el nombre	data_context

Devuelve un pago de servicio por el identificador.	data_context
Devuelve un pago de servicio por el nombre.	data_context
Inserta un nuevo pago de servicio.	data_context
Modifica un pago de servicio.	data_context
Elimina un pago de servicio.	data_context
Listado de pago de servicio.	data_context

Tabla 30 Tarjeta CRC: Control_Trabajador

Nombre de la clase: Control_Trabajador	
Descripción: Esta clase es la encargada de realizar las operaciones de gestión de servicios.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Verifica si existe un trabajador por el nombre	data_context
Verifica si existe un trabajador por el carnet	data_context
Inserta un nuevo trabajador.	data_context
Modifica un trabajador.	data_context
Elimina un trabajador.	data_context
Verifica si tiene dietas ejecutadas.	data_context
Verifica si tiene firma autorizada.	data_context
Dar baja a un trabajador.	data_context
Listado de trabajadores.	data_context
Obtiene dietas de un trabajador.	data_context

Tabla 31 Tarjeta CRC: Controlar_Dieta

Nombre de la clase: Controlar_Dieta

Descripción: Esta clase es la encargada de realizar las operaciones de gestión de dietas.	
Atributos:	**Colaborador**
data_context: DataContext_ORM	
Responsabilidades:	
Devuelve una dieta por su identificador.	data_context
Inserta una nueva dieta.	data_context
Modifica una dieta.	data_context
Elimina una dieta.	data_context
Listado de dietas.	data_context
Listado de dietas abiertas.	data_context
Listado de dietas cerradas.	data_context
Devuelve el monto total de dietas abiertas.	data_context
Devuelve el monto total de dietas cerradas.	data_context
Cerrar una dieta.	data_context

Nota: las tarjetas CRC que faltan se encuentran disponibles en el expediente del proyecto.