

Sistema Integral de Registro y Control de Viáticos para el Tribunal de Cuentas de la provincia de La Pampa

PROYECTO FINAL CON MODALIDAD PROYECTO DE DESARROLLO

Datos Personales:

Apellido, Nombre: Gatti, Marysol

DNI: 32.901.620

N° de legajo: 3746

Universidad Nacional de La Pampa

Facultad de Ingeniería

Carrera: Ingeniería en Sistemas

Director de Tesis: Lic. Guillermo Javier Lafuente

Agradecimientos

A mis padres por todo el esfuerzo realizado para que yo hoy pueda estar culminando mi carrera universitaria. Por todo su amor y apoyo incondicional, este logro es por y para ustedes.

A mis hermanos por ser mis amigos y alentarme siempre a seguir adelante.

A mis abuelos por regalarme tanto de sus vidas.

A mis amigas de siempre que la distancia fortaleció aún más el vínculo y me han apoyado desde lejos todos estos años.

A mis compañeros de facultad por compartir tan gratos momentos y por la hermosa posibilidad de aprender juntos.

A mis amigos que esta etapa facultativa me regaló y estando lejos de casa se vuelven familia, en especial Daniela por haber compartido y aprendido tanto juntas.

A mi tutor Guillermo Lafuente por haberme guiado durante este proyecto y compartir su conocimiento, por su gentileza y predisposición: GRACIAS.

A los docentes y no docentes de la Facultad de Ingeniería por colaborar en la formación de profesionales inculcando siempre buenos valores.

Al personal del Tribunal de Cuentas especialmente a Marcelo Rubio por su predisposición, paciencia y amabilidad durante la realización de este proyecto.

Por último gracias a Fede, mi marido, por enseñarme tanto de la Ingeniería pero mucho más de la vida y el amor.

Prefacio

El siguiente documento se presenta como trabajo final requerido para la obtención del grado Académico de Ingeniería en Sistemas de la Universidad Nacional de La Pampa. El mismo se desarrolló bajo la modalidad de proyecto de desarrollo y se basó en la realización de un Sistema Integral de Registro y Control de Viáticos para el Tribunal de Cuentas de la Provincia de La Pampa, abarcando un tiempo comprendido entre el período del 10 de diciembre de 2014 y el 18 de mayo de 2016 aproximadamente. Se trabajó bajo la tutela del Licenciado Guillermo Javier Lafuente y desde el Tribunal de Cuentas con el Analista Programador Marcelo Rubio.

Contenido

Prefacio	1
Tabla de Figuras	5
Tabla de Tablas.....	7
1. Introducción	8
1.1. Identificación del problema	8
1.2. Objetivos	9
1.3. Estructura del trabajo final.....	9
2. Marco Conceptual.....	11
2.1. Modelo Espiral.....	11
2.1.1. Regiones de tareas	12
2.1.2. Modelo Espiral Win-Win (Victoria-Victoria).....	12
2.2. Modelo Vista Controlador	13
2.3. Arquitectura Cliente – Servidor.....	14
2.3.1. Componentes de la arquitectura cliente/servidor.....	15
2.4. Apreciaciones del capítulo	15
3. Planificación del Proyecto	17
3.1. Establecer el ámbito y determinar la factibilidad del software	17
3.1.1. Requerimientos funcionales	18
3.1.1.1. Consideraciones del requerimiento funcional: confección de comisiones.....	19
3.1.2. Requerimientos no funcionales.	19
3.2. Definir los recursos requeridos	20
3.3. Estimar el costo y esfuerzo.....	21
3.4. Comunicación con el cliente	22
3.5. Generación de documentación.....	22
3.5.1. Especificación de Requerimientos de Software (ERS).....	24
3.5.2. Diagramas de casos de uso	24
3.5.3. Diagramas de actividades.....	24
4. Diseño.....	27
4.1. Diseño de los Datos	27
4.1.1. Diagrama de Entidad - Relación	28

4.1.2. Diccionario de Datos	29
4.2. Diseño de la Arquitectura	29
4.2.1. Estilos de arquitectura	29
4.2.2. Patrones arquitectónicos	30
4.3. Diseño de la Interfaz de usuario.....	31
4.3.1. Diseño de prototipos para requerimientos específicos.....	32
4.3.2. Ventajas de utilizar patrones de diseño de interfaz	33
5. Elección de tecnologías para la implementación.....	35
5.1. CRM	35
5.2. Vtiger CRM	35
5.3. Servidor Apache	36
5.4. Phpmyadmin	36
5.5. Tecnologías relacionadas al versionado de código	37
5.5.1. Git.....	37
5.5.2. Fundamentos de Git.....	37
5.5.3. BitBucket.....	38
5.5.4. Tortoise Git.....	39
5.6. Putty	39
5.7. Entorno de desarrollo Eclipse	40
5.8. Google Maps APIS	40
6. Codificación	42
6.1. Entorno de desarrollo	42
6.1.1. Versionado del código.....	44
6.2. Codificación de funcionalidades.....	46
6.2.1. Creación de módulos.....	46
6.2.2. Módulos secundarios	47
6.2.2.1. Funcionalidades relacionadas al módulo Usuario.....	48
6.2.3. Módulos primarios	50
6.2.3.1. Creación de comisiones.....	53
6.2.3.2. Codificación particular para dependencia Vialidad.....	58
6.2.3.3. Control de vehículos en la solicitud y rendición de viáticos	58
6.2.3.4. Destino de comisiones a través de Google Maps	59

6.2.3.5. Configuración de los montos de viáticos según variables.....	61
6.2.3.6. Módulo de control.....	63
6.2.3.7. Tareas añadidas al cron.....	65
6.2.3.8. Impresión de documentos, cargar itinerario y gastos	66
6.2.3.9. Restricción de acceso a jurisdicciones	67
7. Pruebas y puesta en marcha	69
7.1. Pruebas de software	69
7.1.1. Pruebas de tipo caja negra.....	69
7.1.2. Pruebas de tipo caja blanca	71
7.1.3. Pruebas unitarias.....	73
7.1.4. Pruebas de integración	74
7.1.5. Pruebas de sistema	74
7.2. Puesta en marcha del sistema.....	76
7.3. Cron, tareas automatizadas.	77
7.4. Herramientas para la toma de decisiones.	78
8. Conclusiones.....	80
Anexo I: Especificación de requerimientos	82
Anexo II: Plantilla Casos de Uso	87
Anexo III: Planilla de Solicitud de Viáticos.....	88
Anexo IV: Planilla de Rendición de Viáticos	89
Anexo V: Planilla de Carga de Itinerario y Gastos	90
Anexo VI: Documento Generado por el Sistema (Planilla de Rendición de Viáticos)	91
Bibliografía	92

Tabla de Figuras

Figura 1: Modelo de Espiral común.....	12
Figura 2: Arquitectura Modelo-Vista-Controlador.....	14
Figura 3: Diagrama de casos de uso	25
Figura 4: Diagrama de actividades relacionado de la selección de vehículos.....	26
Figura 5: Diagrama de entidad y relación.	28
Figura 6: Captura del Sistema de Control de Viáticos	31
Figura 7: Captura del Sistema de Control de Viáticos. Lista de Comisiones	32
Figura 8: Armado de comisión desde el módulo rendición	33
Figura 9: Pantalla perteneciente a la carga de destinos desde Google Maps	34
Figura 11: Primera pantalla de instalación Vtiger	43
Figura 12: Requisitos para la instalación de Vtiger	43
Figura 13: Parámetros de configuración de Vtiger	44
Figura 14 : Creación de repositorio desde Bitbucket	44
Figura 15: Clonación de repositorio desde el cliente Tortoise Git	45
Figura 16: Sincronización de la copia local con el servidor de versiones.....	45
Figura 17: Template de soporte al español del modulo Afiliado	47
Figura 18: Interfaz que permite añadir campos y bloques	48
Figura 19: Fragmento de código relacionado a la creación de módulos	49
Figura 20: Fragmento de código que relaciona a un usuario con un afiliado	49
Figura 21: Fragmento de código perteneciente a la capa lógica de solicitud y rendición	50
Figura 22: Fragmento de código perteneciente a la vista de solicitud y rendición	50
Figura 23: Módulo de control.....	51
Figura 24: Visualización de rendiciones en el módulo control.....	51
Figura 25: Fragmento de código que permite la relación entre dos módulos.....	52
Figura 26: Fragmento de código que permite la relación entre módulos (cont).....	52
Figura 27: Interfaz que muestra la relación entre dos módulos	53
Figura 28: Fragmento de código de la vista de solicitud y rendición	54
Figura 29: Interfaz para el armado de comisiones.....	54
Figura 30: Fragmento de código de la capa lógica para recuperar los datos de un afiliado.....	55
Figura 31: Fragmento de código cliente para agregar afiliados a una comisión	55
Figura 32: Fragmento de código cliente para obtener el viático por día en pesos.....	56
Figura 33: Fragmento de código perteneciente a la capa lógica de la consulta de viático por día ..	56
Figura 34: Código cliente que permite agregar un afiliado a una comisión	57
Figura 35: Interfaz de confección de comisiones y cálculos de subtotales y totales.....	57
Figura 36: Campos añadidos para vialidad en la confección de comisiones	58
Figura 37: Interfaz de consulta de disponibilidad de los vehículos.....	59
Figura 38: Validación de control de disponibilidad de vehículos en solicitudes y rendiciones	59
Figura 39: Fragmento de código para la creación del mapa	60
Figura 40: Código cliente para dar soporte a los marcadores en el mapa.....	60

Figura 41: Visualización de los marcadores en el mapa.....	61
Figura 42: Módulo de configuración de los montos de viático.....	62
Figura 43: Módulo de configuración de los montos de viático(cont)	62
Figura 44: Fragmento de código que crea un registro de control luego de la rendición.....	63
Figura 45: Primer bloque del módulo de control.....	63
Figura 46: Comparación de los vehículos de una solicitud y una rendición de un mismo viático	64
Figura 47: Comparación de la cantidad de afiliado entre una solicitud y una rendición de un mismo viático	64
Figura 48: Interfaz de lista de registros de control	64
Figura 49: Fragmento de código que pinta la lista de registros de control	65
Figura 50: Programa Flujo de Control	66
Figura 51: Fragmento de código perteneciente a la impresión de archivo PDF	67
Figura 52: Edición del archivo de configuración del sitio de producción.....	77
Figura 53: Informes de la dependencia Salud	79
Figura 54: Configuración de un filtro para el módulo solicitud.	79

Tabla de Tablas

Detalle de tareas y duración del proyecto de desarrollo.....	23
--	----

1. Introducción

El siguiente informe final, con modalidad Proyecto de Desarrollo, detalla la realización del Sistema Integral de Registro y Control de Viáticos (SIRCV) para el Tribunal de Cuentas de la provincia de La Pampa. El proyecto comienza en el análisis de requerimientos y culmina en la puesta en marcha del SIRCV, por lo que se especifican todas las etapas del ciclo de vida del software así como también la información teórica utilizada para la realización del mismo.

El sistema integral permite la realización de todos los procesos administrativos que se realizan actualmente de forma manual para el control y liquidación de viáticos de todas las entidades dependientes del Gobierno de la provincia de La Pampa.

En el informe se encuentran detalladas las problemáticas que se tuvieron que superar en las distintas etapas del ciclo de vida de software, principalmente porque no existe un sistema de base, ni un modelo de entidades y relaciones, así como tampoco, un manual de procedimientos. Es por esto que, para la realización del SIRCV, se realizaron múltiples entrevistas y encuentros con el personal del tribunal.

Para el desarrollo del sistema se utilizó el ciclo de vida “En Espiral Win Win” diseñado por el equipo de Barry Boehm (1). Se consideró que es el ciclo de vida más apropiado a los requerimientos planteados. El modelo Win-Win (todos ganan) es una adaptación del modelo espiral que se enfatiza en la participación del cliente en el proceso de desarrollo de un producto de software. El personal del tribunal de cuentas tuvo una participación activa en todas las decisiones del diseño del software.

Tal como se ha mencionado el software registra y controla los viáticos de toda la provincia de La Pampa. Por el momento sólo será utilizado por el personal del Tribunal de Cuentas pero en un futuro cercano será utilizado por todo el personal de la provincia. Es decir es un sistema de gran envergadura que debe satisfacer eficientemente factores de calidad internos y externos. Debido a estas circunstancias, en vez de codificar un sistema desde cero, se realizó una customización de una tecnología mundialmente conocida que es Vtiger (2).

Vtiger es una aplicación CRM (3) de código abierto, desarrollada principalmente en PHP (uno de los requerimientos solicitados por el tribunal), que ya plantea una estructura para la administración de usuarios, permisos y perfiles, así como también, una gestión modular con informes sumamente útiles para la toma de decisiones. Características imprescindibles para el desarrollo de este sistema.

1.1. Identificación del problema

El Tribunal de Cuentas de la provincia de La Pampa tiene diversas funcionalidades, una de ellas es controlar y realizar la liquidación de los viáticos de todas las instituciones públicas pertenecientes al gobierno provincial.

Actualmente, este proceso administrativo utiliza el papel como soporte físico de la información. El proceso de control y liquidación de viáticos se separa en dos etapas: Solicitud y Rendición. Ambas son asentadas en sendas planillas, las cuales son controladas por el Tribunal de Cuentas para

validar la información presentada, evitando casos de fraude o error humano. Se debe realizar un cotejo entre la solicitud y la rendición que pertenecen a un mismo viático para realizar una correcta liquidación.

El problema actual del Tribunal de Cuentas reside en que los controles se realizan sobre soportes físicos, es decir sobre los papeles presentados. Se pierden muchísimas horas hombre para poder detectar una falla y no es posible la toma de decisiones puesto que no se puede obtener información global, oportuna y exacta.

Además, otro de los problemas presentados por el tribunal, es la falta de cumplimiento de procedimientos por parte de los empleados. En cada viático debe existir una planilla de solicitud y una de rendición. Cada una de ellas debe pertenecer solamente a un viático. La solicitud debe de presentarse antes de realizar el viático y la rendición después. Este procedimiento es muy difícil de limitar sin un sistema computacional.

A su vez cuando la carga de datos es realizada de forma manual, muchas veces es ilegible, generando pérdida de tiempo y mayores posibilidades de errores en el control y la liquidación.

1.2. Objetivos

El objetivo de esta tesis con modalidad Proyecto de Desarrollo es desarrollar un sistema orientado a la web que permita digitalizar los procesos administrativos que se llevan a cabo en la rendición y liquidación de viáticos. Los usuarios podrán acceder al sistema desde cualquier computadora conectada a la red interna del Tribunal de Cuentas de la provincia, y en un futuro cercano, desde cualquier computadora conectada a internet.

Uno de los propósitos principales es que el sistema se asemeje lo más posible al proceso manual que actualmente lleva a cabo el tribunal, y que además, brinde información oportuna y exacta para la toma de decisiones.

Es decir, el sistema no solamente debe realizar lo que actualmente se hace manualmente sino que, aprovechando la digitalización de los datos, brinde información útil a directivos y evite situaciones de fraude.

1.3. Estructura del trabajo final

El siguiente informe final se encuentra estructurado en capítulos, y la descripción de los mismos se detalla a continuación.

Capítulo 2: Marco conceptual. Se detalla el ciclo de vida de software utilizado y se introducen conceptos relacionados a las tecnologías utilizadas para una correcta comprensión del informe.

Capítulo 3: Planificación del proyecto. En este capítulo se detalla el modelado de los requerimientos, análisis y generación de documentos funcionales.

Capítulo 4: El proceso de diseño. Se especifican las tareas de diseño realizadas donde el cliente tiene mayor participación. Se introducen conceptos sobre el diseño y luego una especificación del diseño arquitectónico y de componentes.

Capítulo 5: Elección de tecnologías para la implementación. Se detallan las tecnologías utilizadas en la implementación. Se especifican las características de estas y los motivos por los cuales fueron seleccionadas.

Capítulo 6: Codificación. Se mencionan las tareas realizadas en dicha etapa asociadas a los requerimientos.

Capítulo 7: Pruebas y puesta en marcha. Se explican las tareas de pruebas funcionales y de requerimientos realizadas. Se describen los tipos de pruebas realizadas de forma automática y manual.

Capítulo 8: Conclusiones. Se exponen las conclusiones sobre el informe según los objetivos planteados.

2. Marco Conceptual

Todo proyecto de software se desencadena por alguna necesidad de negocios a saber: la de corregir un defecto en una aplicación existente, la de adaptar un sistema a un ambiente de negocios cambiante, la de ampliar las funciones y características de una aplicación ya existente o la necesidad de crear un producto, servicio o sistema nuevo.

El siguiente informe describe al SIRCV, un software realizado desde cero, debido a que no existe un sistema de base. Es decir, todos los requerimientos planteados por el cliente anteriormente se realizaban de forma manual. Es por esto que se deben tomar decisiones en cuanto al ciclo de vida de software a utilizar, arquitectura de software y patrones arquitectónicos.

A continuación se expone un marco conceptual de las herramientas seleccionadas junto con los motivos de dicha elección.

2.1. Modelo Espiral

El desarrollo en espiral es un modelo de ciclo de vida del software definido por primera vez por Barry Boehm (4). Las actividades de este modelo se conforman en una espiral tal como lo describe Pressman en su libro Ingeniería de Software (5), en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior. Para ello, se comienza mirando las posibles alternativas de desarrollo, se opta por la de riesgo más asumible y se hace un ciclo de la espiral. Si el cliente quiere seguir haciendo mejoras en el software, se vuelven a evaluar las distintas alternativas nuevas y riesgos realizándose otra vuelta de la espiral. Es decir, con el empleo del modelo espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, se entrega un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema.



Figura 1: Modelo de espiral común (Ingeniería de Software, pp 39, Roger Pressman 2010)

El primer circuito alrededor de la espiral da como resultado el desarrollo de un modelo del producto; las vueltas sucesivas se usan para desarrollar un prototipo y, luego, versiones cada vez más sofisticadas del software. Cada paso por la región de planeación da como resultado ajustes en el plan del proyecto. El costo y la programación de actividades se ajustan con base en la retroalimentación obtenida del cliente después de la entrega. El modelo espiral es un enfoque

realista para el desarrollo de sistemas y de software a gran escala. Como el software evoluciona a medida que el proceso avanza, el desarrollador y cliente comprenden y reaccionan mejor ante los riesgos en cada nivel de evolución.

El modelo espiral, esquematizado en la Figura 1, usa los prototipos como mecanismo de reducción de riesgos, pero, más importante, permite aplicar el enfoque de realizar prototipos en cualquier etapa de la evolución del producto. Mantiene el enfoque de escalón sistemático sugerido por el ciclo de vida clásico, pero lo incorpora en una estructura iterativa que refleja al mundo real en una forma más realista. El modelo espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y, si se aplica de manera apropiada, debe reducir los riesgos antes de que se vuelvan un problema.

2.1.1. Regiones de tareas

El modelo en espiral se divide en un número de actividades de marco de trabajo, también llamadas regiones de tareas.

Generalmente, existen entre tres y seis regiones de tareas, los cuales son:

- **Comunicación con el cliente:** las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- **Planeación:** las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto. Se identifican los riesgos y se realizan estimaciones.
- **Modelado:** se analizan las diferentes alternativas frente a los riesgos planteados y se realizan las tareas de análisis y diseño requeridas para construir una o más representaciones de la aplicación.
- **Construcción:** las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica)
- **Despliegue:** las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante las regiones previas de una misma iteración.

2.1.2. Modelo Espiral Win-Win (Victoria-Victoria)

Una variante interesante del modelo espiral es el modelo espiral Win-Win. El Modelo Espiral clásico sugiere la comunicación con el cliente solamente para fijar los requisitos, pero esto es en un contexto ideal que rara vez ocurre. Normalmente cliente y desarrollador entran en una negociación, se negocia costo frente a funcionalidad, rendimiento, calidad, etc.

El modelo Win-Win define un conjunto de actividades de negociación al principio de cada paso alrededor de la espiral:

1. Identificación del sistema o subsistemas claves del cliente (saber qué quiere).

2. Determinación de "condiciones de victoria" del cliente (saber qué necesita y satisface)
3. Negociación de "condiciones de victoria" del cliente para obtener condiciones "Win & Win" (negociar para que ambos ganen).

2.2. Modelo Vista Controlador

La arquitectura Modelo - Vista - Controlador (MVC) es uno de varios modelos sugeridos para la infraestructura de aplicaciones webs que desacoplan la interfaz de usuario de sus funciones y contenido informativo (5).

El modelo contiene todo el contenido y la lógica de procesamiento específicos de la aplicación, incluso todos los objetos de contenido, acceso a fuentes de datos o información externos y todas las funciones de procesamiento que son específicas de la aplicación.

La vista contiene todas las funciones específicas de la interfaz y permite la presentación de contenido y lógica de procesamiento, incluidos todos los objetos de contenido, el acceso a fuentes de datos o información del exterior y todas las funciones de procesamiento que requiere el usuario final.

El controlador administra el acceso al modelo y la vista, y coordina el flujo de datos entre ellos. En una aplicación web, la vista es actualizada por el controlador con datos del modelo, basándose en las entradas que da el usuario. En la Figura 2 se muestra una representación de la arquitectura MVC.

En relación con dicha Figura, el controlador maneja las solicitudes o datos del usuario. El controlador también selecciona el objeto de vista que sea aplicable con base en la solicitud del usuario.

Una vez determinado el tipo de solicitud, se transmite al modelo un pedido de comportamiento, que implementa la funcionalidad o recupera el contenido requerido para dar acomodo a la solicitud.

El objeto de modelo accede a los datos almacenados en una base de datos corporativa, como parte de un almacén de datos locales o como una colección de archivos independientes.

El objeto de vista apropiado debe dar formato y organizar los datos desarrollados por el modelo para luego transmitirlos desde el servidor de la aplicación hacia el navegador del cliente para que se desplieguen en la máquina de éste.

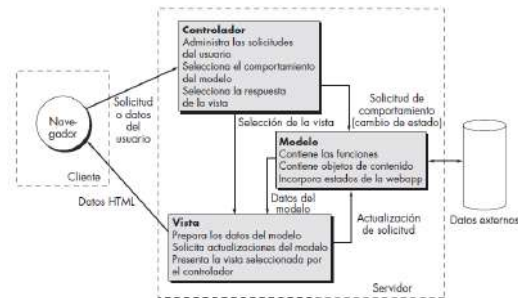


Figura 2: Arquitectura Modelo-Vista-Controlador (Ingeniería de Software, pp 329, Roger Pressman 2010)

2.3. Arquitectura Cliente – Servidor

La arquitectura cliente - servidor es el procesamiento cooperativo de información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales (6).

Desde el punto de vista funcional, se puede definir al modelo cliente-servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aún en entornos multiplataforma.

En el modelo usual cliente-servidor, un servidor se activa y espera las solicitudes de los clientes que comparten los servicios que este brinda. Estas solicitudes pueden darse de forma simultánea y es por ello que el servidor debe garantizar soporte de múltiples conexiones basándose en dos factores fundamentales: disponibilidad y confiabilidad.

Además, el cliente no conoce la lógica del servidor sólo conoce su interfaz externa (que debe estar bien definida), como tampoco depende de la ubicación física del servidor, ni del tipo de equipo físico del mismo ni de su sistema operativo. Por ejemplo, en el caso del SIRCV sólo depende del programa cliente que es el Navegador.

El esquema de funcionamiento de un sistema cliente-servidor es:

- El cliente solicita una información al servidor.
- El servidor recibe la petición del cliente.
- El servidor procesa dicha solicitud.
- El servidor envía el resultado obtenido al cliente.
- El cliente recibe el resultado y lo procesa.

2.3.1. Componentes de la arquitectura cliente/servidor

El modelo cliente - servidor es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además, esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos.

Por lo tanto, según Falgueras (6), los tres elementos fundamentales sobre los cuales se desarrollan e implantan los sistemas cliente – servidor son: el proceso cliente que es quien inicia el diálogo, el proceso servidor que pasivamente espera a que lleguen peticiones de servicio y el middleware que corresponde a la interfaz que provee la conectividad entre el cliente y el servidor para poder intercambiar mensajes.

2.4. Apreciaciones del capítulo

Esta sección introdujo los modelos utilizados para el desarrollo del SIRCV, debido a que en los próximos capítulos se hacen referencia. La inmersión pretendió que el lector comprenda estos conceptos y que los tenga presente en cada capítulo ya que cada etapa del proyecto fue realizada mediante el ciclo de vida en espiral (considerando la mejora Win-win), utilizando el patrón arquitectónico MVC e implementándose en un modelo cliente – servidor.

En las primeras reuniones con el Tribunal de Cuentas se detectó que el sistema sería complejo y que existían puntos a resolver que no se podían planificar en una primera etapa (ya que dependían de la realización de otros). Es decir se necesitaba avanzar en el proyecto, para detectar los riesgos y decidir conjuntamente con el cliente. Por ello se seleccionó un ciclo de vida en espiral junto con la mejora Win-Win. Este modelo propone que se realicen distintas entregas evolutivas donde cada una de ellas son versiones cada vez más sofisticadas que van ajustando el plan de proyecto.

Además, el cliente es una parte fundamental ya que interviene en cada vuelta del espiral, que fue lo que se realizó en este trabajo. La autora se reunió con el cliente frecuentemente durante el período de tiempo que tomó la realización del proyecto, en las instalaciones de la Facultad de Ingeniería de General Pico, para debatir requerimientos, analizar riesgos y para la presentación de prototipos.

En cuanto a la selección del patrón arquitectónico, se seleccionó el MVC ya que al dividir la lógica del negocio permite que el proyecto sea más escalable, esta ventaja es muy importante ya que el equipo de desarrollo del Tribunal de Cuentas continuará con el mantenimiento y futuras modificaciones del sistema.

Por último el sistema se encuentra implementado en una arquitectura cliente – servidor, ya que se adapta a los requerimientos que expresó el cliente. Los datos deben estar centralizados y

accesibles simultáneamente desde cualquier dependencia de la provincia de La Pampa. En la primera entrega solo será accedido en una red local para en un futuro cercano desde cualquier computadora con conexión a Internet.

En los próximos capítulos se detallan las etapas del ciclo de vida del software llevadas a cabo para la realización del SIRCV, utilizando los métodos descritos en este capítulo.

3. Planificación del Proyecto

La planificación establece un mapa de ruta para llegar a los objetivos planteados. Una correcta planificación del proyecto de software incrementa las posibilidades de éxito. Antes del comienzo del proyecto el equipo de software debe estimar el trabajo que se va a realizar, los recursos que se requerirán y el tiempo que transcurrirá.

A continuación se detallan las actividades asociadas a la planificación del proyecto, en cada tarea primero se explica el procedimiento general para cualquier proyecto de software y luego se expone lo realizado en el Sistema de Control de Viáticos.

3.1. Establecer el ámbito y determinar la factibilidad del software

El primer paso dentro de la planificación según Pressman (5) es la de establecer el ámbito del software y determinar la factibilidad del mismo.

El ámbito del software describe las funciones y características que se entregan a los usuarios finales, los datos que son de entrada y de salida, las interfaces, las restricciones, el rendimiento y la confiabilidad que se asocian al sistema.

Para definir el ámbito se pueden realizar varias actividades, entre ellas se realiza una narrativa luego de la interlocución con los participantes del proyecto y el cliente; o directamente el cliente desarrolla algunos casos de usos puntuales. En muchas ocasiones esto no es suficiente y es necesario desglosar las actividades o casos de usos para llegar a un mayor nivel de detalle y realizar una mejor estimación.

Una vez definido el ámbito se realiza una primera estimación y se analiza la factibilidad de culminación del proyecto bajo la estimación planteada.

Para definir el ámbito del SIRCV, siguiendo una metodología de ciclo de vida en espiral, se ponderó fuertemente la participación del cliente. Luego se analizaron los riesgos, se tomaron las decisiones que se consideraron adecuadas y se estimaron los tiempos y costos.

En las primeras reuniones la comunicación era dificultosa. Existía mucha información desconocida, el dominio era totalmente ajeno y además, al no existir un manual de procedimientos, el personal del Tribunal de cuentas tuvo que transmitir con un gran nivel de detalle la forma que ellos realizan su trabajo. Existían situaciones o procedimientos que pretendían modificar y no llegaban a un acuerdo antes de las reuniones, las cuales sirvieron para reglamentar y organizar muchos aspectos de sus procedimientos. Sobre todo establecer reglas para que, por ejemplo, no se presenten solicitudes y rendiciones en simultaneo o que todos los datos sean cargados en ambas etapas.

El Tribunal de Cuentas se divide en áreas, donde en cada área se realiza el control y liquidación de viáticos pertenecientes a comisiones de una dependencia de la provincia, pero como no todas las dependencias realizan el procedimiento de la misma manera, se realizaron encuentros con las áreas que presentaban particularidades. Aumentando de esta manera notablemente los requerimientos a analizar y planificar.

Entonces se desarrolló una narrativa donde se mostraba en rasgos generales que producto era el que se pretendía obtener.

Luego el cliente planteó los requerimientos funcionales obligatorios. Muchas veces un caso de uso para un usuario final no es factible o aconsejable en términos de sistemas informáticos, por esto se guió y se terminaron de definir las actividades a realizar mediante la confección de distintos documentos y diagramas en soporte papel.

Esta etapa fue una de las más complejas del proyecto, y se ha retomado en etapas posteriores, debido a que el Tribunal de Cuentas no posee un manual de procedimientos. Cada dependencia de la provincia funciona de manera diferente y por ende rinde los viáticos de distinta manera. Las diversas reuniones permitieron refinar y llegar a objetivos y requerimientos claros.

3.1.1. Requerimientos funcionales

En esta sección se detallan los requerimientos funcionales acordados entre la alumna y el cliente. Luego en el capítulo 3.5 se explicará la generación de los documentos que detallan cada uno de ellos. Los requerimientos funcionales analizados son:

- Permitir distintos tipos de usuarios y perfiles según la jerarquía actual de los empleados de la provincia.
- Dar soporte a los distintos roles dentro del Tribunal de Cuentas.
- Permitir cargar las solicitudes y rendiciones de viáticos a todas las dependencias de la provincia, cada dependencia presenta particularidades.
- Realizar tareas de control y preventivas para evitar fraudes en solicitudes y rendiciones de viáticos con respecto al dinero solicitado, los vehículo y los afiliados involucrados.
- Brindar información necesaria para la toma de decisiones.
- Migrar todos los datos aportados por el tribunal para evitar una carga manual y masiva de afiliados.
- Dar soporte a usuarios habilitantes y no habilitantes.
- Permitir la carga de destinos a través de coordenadas.
- Permitir la carga de los valores en pesos de viático por día según el tipo de afiliado y particularidades de la comisión.
- Dar soporte a la confección de comisiones (una comisión es un conjunto de afiliados, cada afiliado es un empleado de la provincia).

- Impresión PDF de los documentos generados por el sistema: solicitud de viáticos, rendición de viáticos, planilla de gastos y planilla de itinerario.
- Otros.

3.1.1.1. Consideraciones del requerimiento funcional: confección de comisiones.

Uno de los requerimientos más complejos es la confección de comisiones. Se deben considerar muchas posibilidades para evitar situaciones de fraude.

En la creación de una solicitud se debe controlar que el afiliado seleccionado no esté asociado a otra comisión en el mismo rango de tiempo. Si esto sucede se advierte. En la rendición se debe advertir pero no se debe permitir guardar.

Un afiliado se asocia a una comisión durante un rango de tiempo, se debe controlar que no se superponga la fecha con ninguna comisión rendida o solicitada.

En el caso de los afiliados que pertenecen a la dependencia Vialidad pueden pertenecer a la comisión durante distintos rangos de tiempo, por ejemplo para un mismo viático un afiliado participa dos días y otro solamente uno. Un claro ejemplo que el Tribunal de Cuentas brindó es la situación de un chofer de vialidad que lleva a un empleado a reparar una ruta y el mismo permanece cinco días en casilla mientras que el chofer vuelve y luego lo va a buscar cuando termina la reparación. El chofer solo participa un día en el viático y quien repara la ruta cinco, como el objetivo es el mismo se realiza una sola solicitud y una sola rendición.

Además en vialidad se debe considerar el horario de llegada y salida para su correcta liquidación. Para el resto de las dependencias, en el caso de que los viáticos solamente duren un día, se debe permitir establecer si el día fue “largo”, “corto” o “medio” . Para estas etiquetas se establecen distintos porcentajes de pago del valor que le corresponde al afiliado por día de viático realizado.

3.1.2. Requerimientos no funcionales.

Dentro del acuerdo sobre los requerimientos no funcionales se determinó cumplir los atributos de calidad FURPS (2):

- Funcionabilidad: asegurar que el producto funciona tal como estaba especificado.
- Usabilidad: asegurar que las personas van a poder utilizar las funcionalidades para cumplir sus objetivos.
- Confiabilidad: es la probabilidad de operación libre de fallas en un entorno determinado y durante un tiempo específico.

- Rendimiento: asegurar que las tareas se realicen correctamente pero también en el menor tiempo posible.
- Mantenibilidad: proveer que un sistema o componente software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios en el entorno fácilmente. Este requerimiento es fundamental ya que parte del equipo de desarrollo del Tribunal de Cuentas continuará con el proyecto.

Además de estos atributos se acordaron cumplir con los siguientes:

- Disponibilidad: Estar disponible 100% durante todo el día ya que se pueden solicitar, rendir o consultar viáticos fuera del horario laboral.
- Seguridad: El acceso al Sistema debe estar restringido por el uso de claves asignadas a cada uno de los usuarios. Sólo podrán ingresar al Sistema las personas que estén registradas, estos usuarios serán clasificados en varios roles y perfiles con acceso a las opciones de trabajo definidas para cada rol.
- El proyecto debe ser un sistema web, basado en una arquitectura cliente-servidor de tres capas y desarrollado bajo el lenguaje PHP.

3.2. Definir los recursos requeridos

En esta etapa se planifican los recursos que deberán estar disponibles en el proyecto.

Recursos Humanos: Se evalúa el ámbito del software y se analizan las habilidades requeridas. Cuando se trata de un proyecto chico una misma persona puede desempeñar varias habilidades.

El Sistema de Control de Viáticos no se trata de un proyecto chico pero es llevado a cabo por una sola persona. Esto no se plantea como un error de estimación de recursos, sino que la vinculación entre la universidad y el tribunal solo evaluó la participación de un solo alumno.

Recursos de software reutilizables: La ingeniería de software basada en componentes pone el énfasis en la reusabilidad, es decir en la creación y reutilización de bloques de software.

En esta etapa de planificación de recursos se analiza si es necesario contar con bloques de software ya realizados y testeados por otros desarrolladores, siempre pensando en código abierto no propietario ya que este último nunca fue considerado ni por el Tribunal ni por la Facultad de Ingeniería.

Reutilizar código abierto permite la posibilidad de obtener ventajas como mayor productividad acortando los tiempos de desarrollo, además permite brindar un software más: estable, seguro, eficiente, con menos errores, etc.

En la planificación de recursos del SIRCV se optó por reutilizar un software de código abierto en PHP: Vtiger (2) CRM. En esta decisión se contempló la experiencia con dicha tecnología, la confiabilidad en la misma y la disminución del tiempo de desarrollo puesto que Vtiger plantea, entre otros, una estructura de usuarios, perfiles y permisos. Así como también posee una interfaz amigable, fácil de modificar y plantea una estructura modular que se adapta perfectamente al ámbito presentado por el Tribunal de Cuentas. Cabe destacar que el equipo de desarrollo del Tribunal de Cuentas planteó la necesidad de que el sistema sea implementado en el lenguaje PHP.

Además, como Vtiger es un CRM posee herramientas para implementar que procesan los datos y brindar información útil para la toma de decisiones (función muy importante en cualquier CRM), como por ejemplo informes específicos, flujos de trabajo, etc. Algunos ejemplos de informes que Vtiger genera automáticamente son: “Gastos de viáticos mensuales por dependencia”, “Cantidad de solicitudes de viáticos sin ser rendidas”, “Rendiciones de viáticos con diferencia en la solicitud asociada”, etc.

Recursos ambientales: El entorno que soporta a un proyecto de software, con frecuencia llamado entorno de ingeniería de software, incorpora al software y hardware.

En esta etapa se planifica los equipamientos de hardware que proporcionen una plataforma que soporte el software requerido.

En la planificación de recursos ambientales del proyecto se debió considerar, tal como se plasmó en el ámbito del proyecto, que se solicitó una arquitectura cliente-servidor. Por ello se planificó la puesta en marcha de un servidor en el Tribunal de Cuentas de la provincia de La Pampa bajo el sistema operativo **Debian 7**.

Además se debió planificar y realizar un minucioso análisis de riesgo, junto con el Tribunal de Cuentas, debido a que el servidor debe poseer características que permitan un óptimo rendimiento. Actualmente el proyecto está planificado para que sea utilizado solamente por personal del Tribunal de Cuentas pero en un futuro cercano será utilizado por todos los organismos de la provincia. Es decir, múltiples personas accediendo al mismo tiempo desde distintos lugares al sistema. Entonces la información debe estar segura pero además el servidor debe brindar un servicio óptimo en cuanto a velocidad y estabilidad, evitando que el proceso de carga de datos o de obtención de información sea lento. Estas características permiten aumentar la posibilidad de que el usuario confíe y utilice el sistema.

3.3. Estimar el costo y esfuerzo

La estimación del costo y el esfuerzo del software contiene demasiadas variables (humanas, ambientales, técnicas, etc.) que pueden afectar al costo y al esfuerzo dedicados al proyecto de software.

Para realizar una correcta estimación de costo y esfuerzo una de las alternativas es utilizar técnicas de descomposición simples. Al descomponer un proyecto en funciones principales y actividades de

ingeniería de software relacionadas, la estimación de costo y esfuerzo puede realizarse en forma escalonada. Además realizar una estimación de un proyecto en su totalidad resulta muy complejo y posiblemente no se llegue a un resultado “real”. Por esta razón se debe descomponer el problema y volver a caracterizarlo como un conjunto de problemas más pequeños.

La estimación del costo y esfuerzo del SIRCV es realizado bajo esta técnica, separando al proyecto en etapas, las cuales fueron mencionadas en el capítulo uno (y plasmadas en capítulos diferentes de este documento). En la Tabla 1 se muestran las tareas que forman parte del proyecto junto con las horas dedicadas a cada una de ellas. La duración total fue estimada en 200 horas.

3.4. Comunicación con el cliente

Una correcta y fluida comunicación con el cliente genera mayores posibilidades de que los resultados cumplan con los requerimientos. La interacción con el cliente sirve para definir puntos importantes y confusos.

Tal como se describió en el Capítulo 1, para la realización del SIRCV la comunicación con el cliente se produce en todas las etapas del proyecto ya que se trata de un modelo en espiral win-win.

Para el desarrollo del proyecto se realizaron encuentros por Skype y reuniones en persona con distintos sectores del Tribunal de Cuentas, para lograr una mayor comprensión sobre los casos particulares de cada sector. Como ya se mencionó, no es la misma administración de viáticos para la dependencia de Salud que de Vialidad por dar un ejemplo. Estas particularidades, debieron ser tenidas en cuenta en la planificación y sobre todo en la estimación.

3.5. Generación de documentación

En muchos proyectos, no solamente los informáticos, la culminación de una fase o tarea trae como resultado la generación de uno o varios documentos. Estos documentos son útiles para aumentar la usabilidad, la vida del software y mejorar el mantenimiento del producto.

En cada etapa del proyecto para el Tribunal de Cuentas de la provincia se realizaron varios documentos, en la etapa de planificación se generó la tabla correspondiente a las estimaciones de horas dedicadas a cada etapa, el documento de especificación de requerimientos, los diagramas de casos de uso y el diagrama de actividades.

Fases/Tareas	Duración hs.
Etapas 1 - Planificación del Proyecto	24
- Charla con el cliente.	6
- Establecer el ámbito del proyecto.	4
- Identificar problemas y requerimientos.	4
- Definir recursos requeridos.	1
- Plantear casos de uso.	3
- Realizar documentación.	6
Etapas 2 - Diseño	26
- Charla con el cliente.	6
- Realizar diseño de los datos.	10
- Diseño arquitectónico y de la interfaz.	5
- Realizar documentación.	5
Etapas 3 - Codificación	100
- Investigación sobre las tecnologías a utilizar.	10
- Instalación del entorno de desarrollo y producción. Servidor apache y phpmyadmin. Instalación de Vtiger.	3
- Codificación de la vista	20
- Codificación del modelo	20
- Codificación del controlador	20
- Codificaciones de funciones especiales añadidas en una capa superior de Vtiger para requerimientos específicos.	20
- Realizar Documentación	5
- Charla con el cliente	2
Etapas 4 – Testing y puesta en marcha	50
- Pruebas de requerimientos específicos	10
- Pruebas globales	10
- Implementación en servidor de producción	5
- Pruebas en el sitio de producción	10
- Documentación operativa del sistema	5
- Charla, capacitación con el cliente y entrega del producto.	10
TOTAL	200 horas

Tabla 1: Detalle de tareas y duración del proyecto de desarrollo.

3.5.1. Especificación de Requerimientos de Software (ERS)

La especificación de requerimientos de software (ERS) constituye un documento que se crea cuando debe especificarse una descripción detallada de todos los aspectos del software que se va a elaborar, antes de que comience su desarrollo. Este documento es el resultado del establecimiento del ámbito y la factibilidad del software abordado en este capítulo.

Para la realización del ERS se utilizó la plantilla propuesta por el sitio www.academia.edu (7) (ver Anexo I) y fue completando conjuntamente con parte del personal del Tribunal de Cuentas.

3.5.2. Diagramas de casos de uso

Los diagramas de casos de uso sirven para mostrar las funciones de un sistema de software desde el punto de vista de sus interacciones con el exterior y sin entrar en la descripción detallada ni en la implementación de estas funciones (6).

Este diagrama también es el resultado del establecimiento del ámbito y la factibilidad del software, anteriormente en la sección 3.1 se nombraron cuáles fueron los casos de usos acordados con el personal del Tribunal de Cuentas, los diagramas especifican cada uno de ellos.

En la Figura 3 se muestra uno de los diagramas de casos de uso obtenidos para un refinamiento de los requerimientos funcionales, para su realización se utilizó la aplicación Start UML (8). Además, como complemento del diagrama y para una especificación detallada de algunos casos (los más complejos) se empleó la plantilla propuesta por el sitio www.academia.edu (9) (ver Anexo II).

3.5.3. Diagramas de actividades

El diagrama de actividades enriquece el caso de uso al proporcionar una representación gráfica del flujo de interacción dentro de un escenario específico. Un diagrama de actividades utiliza rectángulos redondeados para denotar una función específica del sistema, flechas para representar flujo a través de éste, rombos de decisión para ilustrar una ramificación de las decisiones y líneas continuas para indicar que están ocurriendo actividades en paralelo. Debe observarse que el diagrama de actividades agrega detalles adicionales que no se mencionan directamente (pero que están implícitos) en el caso de uso.

En este caso se continuó con la notación basada en UML (10), pues es también conocido por el Tribunal y además es un lenguaje que facilita la comunicación a la hora de comunicar ideas, propuestas y conceptos al cliente. Desde las primeras reuniones con el Tribunal se pudo comprobar que la utilización de diagramas realizados en UML facilitó la tarea de establecer

conceptos y modelar el sistema entre ambas partes. A su vez, la utilización de UML permite brindar un mejor soporte para actualizaciones futuras del software, ya que será realizado por otro equipo de desarrolladores que conocerán el lenguaje y podrán tomar decisiones a partir de los diagramas presentados.

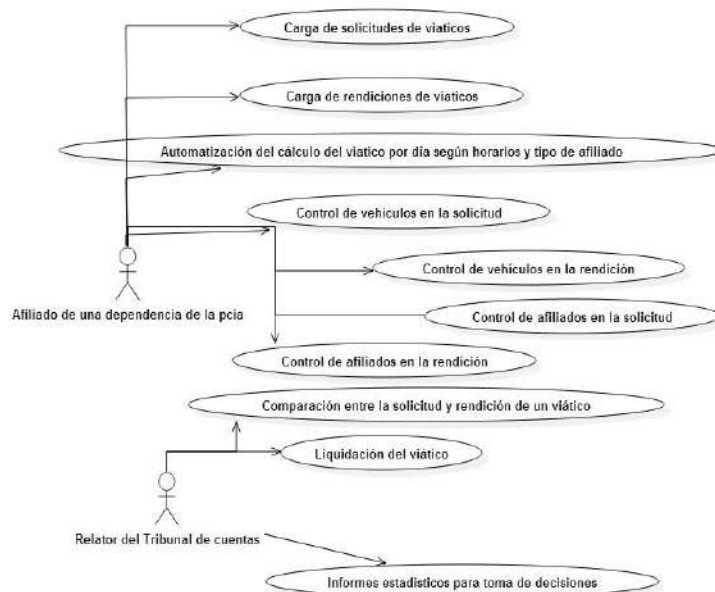


Figura 3: Diagrama de casos de uso

En la planificación del proyecto para el Tribunal de Cuentas se realizaron diagramas solo para algunas actividades, se seleccionaron aquellas que presentaban mayor complejidad para poder lograr una idea más clara y un intercambio más enriquecedor con el cliente. En la Figura 4 se muestra uno de los diagramas correspondientes a la función “Selección de los vehículos en la rendición de un viático”.

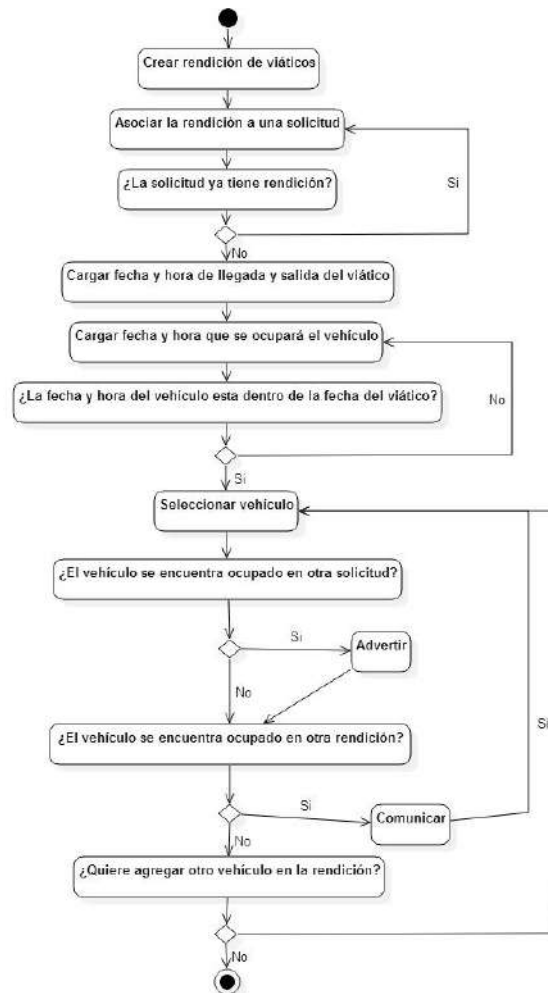


Figura 4: Diagrama de actividades relacionado de la selección de vehículos

4. Diseño

El diseño de software se ubica en el área técnica de la ingeniería de software. El diseño del software comienza una vez que se han analizado y modelado los requerimientos, es la última acción de la ingeniería de software dentro de la actividad de modelado y prepara la etapa de construcción (generación y prueba de código).

El proceso de diseño tiene como principal meta cumplir con lo detallado en el documento de especificación de requerimientos (abordado en el capítulo anterior). Utiliza los documentos generados en el planeamiento del proyecto, pero en esta etapa se refinan y elaboran con más detalles específicos para la implantación.

McGlaughlin (11) define tres características que funcionan como guía a la hora de evaluar un buen diseño:

- Debe implementar todos los requerimientos explícitos contenidos en el modelo de requerimientos y dar cabida a todos los requerimientos implícitos que desean los participantes.
- Debe ser una guía legible y comprensible para quienes generan el código y para los que lo prueban y dan el apoyo posterior.
- Debe proporcionar el panorama completo del software, y abordar los dominios de los datos, las funciones y el comportamiento desde el punto de vista de la implementación

En el proceso del diseño se realiza el diseño de datos, el diseño arquitectónico y el diseño de la interfaz. A continuación se detallan cada uno de estos diseños junto con las elecciones y documentos generados para el proyecto del SIRCV.

4.1. Diseño de los Datos

El diseño de datos crea un modelo de datos y/o información que se representa con un nivel de abstracción (la visión de datos del cliente/usuario). Este modelo de datos, es entonces refinado en progresivas representaciones específicas de la implementación, que pueden ser procesadas por un sistema basado en computadora. En muchas aplicaciones de software, según Pressman (5), la arquitectura de datos tendrá una gran influencia sobre la arquitectura del software que debe procesarlo.

Los objetos de datos definidos durante el planeamiento del proyecto son modelados utilizando diagramas de entidad-relación y acompañados con el diccionario de datos.

Un diseño de datos bien estructurado ayuda a simplificar el flujo del programa, hace más fácil el diseño e implementación de componentes de software y más eficiente el procesamiento conjunto.

En la realización del diseño de los datos del proyecto del Tribunal de cuentas se presentaron muchas complicaciones. Más allá de los requerimientos y las especificaciones concretas, el

Tribunal entregó un conjunto de tablas con datos extraídas de otro sistema interno del propio Tribunal de Cuentas.

El cliente solicitó que los datos sean migrados para evitar la carga manual (el SIRCV administrará a todos los empleados de la provincia). Las tablas entregadas contenían muchísima información útil para este proyecto pero la ingeniería inversa no fue una tarea fácil. No existía cohesión en los datos y además eran redundantes.

Por ejemplo, se pretendía obtener todos los afiliados que existían en la provincia junto con la dependencia a la que pertenecían y el cargo jerárquico. Al no existir una clave única, se intentó identificar a cada afiliado por el número de afiliado o por el número de documento, pero estos se encontraban de distintas maneras por ejemplo: Afiliado 1, DNI: 3290160; Afiliado 1, DNI: 32.901.620. Para esta problemática se crearon scripts específicos para automatizar la depuración de los datos.

Otro problema era que si un afiliado trabajó en x número de dependencias por ejemplo: Salud, Justicia y Educación, el mismo se encontraba 3 veces repetido, pero el cliente había especificado que solo podía realizar viáticos para una dependencia de la provincia. Para esta problemática también se realizaron distintos scripts para la limpieza de datos.

4.1.1. Diagrama de Entidad - Relación

El diagrama de entidad - relación (DER) representa gráficamente los objetos de datos y las relaciones que forman parte del sistema. Los objetos de datos se representan con un rectángulo etiquetado. Las relaciones se indican con una línea etiquetada que conecta objetos.

Este documento, como los anteriores, ha sido muy útil para la interacción con el cliente y para el correcto diseño de los datos. En la Figura 5 se muestra el DER correspondiente al Sistema de Control de Viáticos.

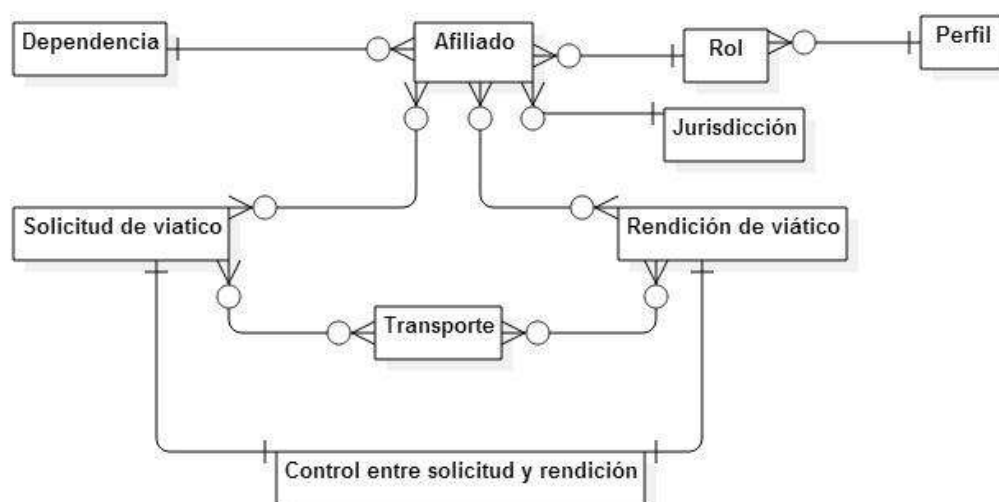


Figura 5: Diagrama de entidad - relación.

4.1.2. Diccionario de Datos

Un diccionario de datos es un conjunto de metadatos que contiene las características lógicas de los datos que se van a utilizar en el sistema. El objetivo de un diccionario de datos es dar precisión sobre los datos que se manejan en un sistema, evitando así malas interpretaciones o ambigüedades.

La realización del diccionario de datos en este proyecto forma parte de la arquitectura de datos y se ha realizado posteriormente a la confección del DER. Se describe por cada entidad los atributos que le corresponde estableciendo las claves primarias, foráneas según las relaciones establecidas en el diagrama.

4.2. Diseño de la Arquitectura

El diseño arquitectónico se centra en la representación de la estructura de los componentes del software, sus propiedades e interacciones. En el contexto del diseño de la arquitectura, un componente software puede ser algo tan simple como un módulo de programa o una clase en el paradigma orientado a objetos, pero también puede ampliarse para que incluya bases de datos y “middleware” que permitan la configuración de una red de clientes y servidores.

Las propiedades de los componentes son aquellas características necesarias para entender cómo interactúan unos componentes con otros. En el nivel arquitectónico, no se especifican las propiedades internas (por ejemplo, detalles de un algoritmo). Las relaciones entre los componentes pueden ser tan simples como una invocación de procedimiento de un módulo a otro o tan complejo como un protocolo de acceso a una base de datos.

4.2.1. Estilos de arquitectura

Existen distintos estilos arquitectónicos entre los cuales un diseñador de software puede optar, según los requerimientos, a saber:

- Arquitectura centrada en datos
- Arquitectura de flujo de datos
- Arquitectura de llamar y regresar
- Arquitectura orientada a objetos
- Arquitectura en capas

Según los requerimientos, analizados en los primeros capítulos, se optó por seleccionar para el proyecto una arquitectura de llamar y regresar. Este estilo arquitectónico permite obtener una

estructura fácil de escalar y modificar. Dentro de este estilo se encuentra un sub-estilo denominado: Arquitectura de llamada a procedimiento remoto.

En las arquitecturas de llamada de procedimiento remoto los componentes de una arquitectura de programa principal/subprograma están distribuidos a través de computadoras múltiples en una red.

Una arquitectura cliente-servidor es del tipo de llamada-regreso. Como se introdujo en el Capítulo 1 es la seleccionada para este proyecto. Las cualidades que este estilo fueron las determinantes para seleccionarlo ya que uno de los requerimientos principales planteados por el Tribunal, es que cada dependencia pueda conectarse a través de Internet al SIRCV para poder cargar en tiempo real las solicitudes y rendiciones. Además otro requerimiento solicitado fue que los datos estén centralizados y no distribuidos.

Además se estableció que por el momento será necesario sólo un servidor y como la aplicación es orientada a la web, los clientes solo deben contar con un navegador web para interactuar con el mismo.

4.2.2. Patrones arquitectónicos

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos.

Dentro de los patrones arquitectónicos, según Pressman (5), podemos encontrar:

- Modelo vista controlador
- Inyección de dependencias
- Arquitectura dirigida por eventos
- Arquitectura orientada a servicios

Como se planteó en el Capítulo 1, el patrón de diseño seleccionado para este proyecto es el de MVC. Debido a que como indica el autor del libro “Implantación de aplicaciones web en entornos internet, intranet y extranet” (12) una de las ventajas de este patrón es que convierte la aplicación en un modelo modular fácil de entender a simple vista y de poder actualizarlo también.

El patrón de diseño MVC desacopla la interfaz de usuario de sus funciones y contenidos informativos. Esto permite que se puedan actualizar partes del sistema sin afectar al resto. Por lo tanto, se consideró la mejor opción para este proyecto debido a que el sector de informática del Tribunal de Cuentas continuará el proyecto con futuros requerimientos.

Además, la selección de este patrón hace beneficiosa la codificación, la implementación y el testing. Se pueden realizar módulos y ser probados de forma independiente. Con respecto a la vista muchas veces se puede reutilizar código, ya que directamente no depende de ningún modelo. Por ejemplo, la misma vista que se codifica para la solicitud de viáticos puede ser la de

rendición de viáticos ya que ambos módulos contienen los mismos campos para funciones diferentes.

4.3. Diseño de la Interfaz de usuario

La interfaz de usuario es la manifestación visible del software y el diseño de la misma describe la forma en el que el software se comunica con los usuarios.

Esta etapa incorpora elementos estéticos (como distribución, color, gráficos, mecanismos de interacción, etc.), elementos ergonómicos (por ejemplo, distribución y colocación de la información, navegación por la interfaz, etc.) y elementos técnicos (como patrones reutilizables).

Como se expuso en los capítulos anteriores la implementación del SIRCV es realizada mediante una adaptación de la tecnología Vtiger. Esta decisión se debió a la confiabilidad que presenta esta herramienta y la posibilidad de reutilización de múltiples de funciones. Uno de los componentes a reutilizar es la interfaz. Vtiger, como se puede observar en las Figuras 6 y 7, posee una interfaz muy amigable, fácil de modificar y de agregar componentes. Moderna, adaptable al usuario final (según experiencias de trabajos previos al SIRCV) y tiene predefinido cuatro estilos que pueden ser heredados para que las nuevas funcionalidades tengan la misma “vista” que el sistema.

Avanzada

Información de la Rendición de Servicio

* Descripción	<input type="text"/>	* Corresponde a la Comisión de Servicios N°	<input type="text"/> Escribe aquí para buscar <input type="button" value="Q"/> <input type="button" value="+"/>
Año	<input type="text"/>	Dependencia	<input type="text"/> Escribe aquí para buscar <input type="button" value="Q"/> <input type="button" value="+"/>
Fecha	<input type="text"/>	Motivo de la Comisión	<input type="text"/>
Lugar	<input type="text"/> Destino en Google Maps <input type="button" value="Destino en Google Maps"/>	Fecha Iniciación	<input type="text"/>
Hora Inicio	<input type="text"/>	Fecha de Finalización	<input type="text"/>
Hora Finalización	<input type="text"/>	Coordenadas	<input type="text"/>

Detalle Comisión - Vehículo 1

Tipo de Medio de Transporte a Utilizar	<input type="text"/> Selecciona una Opción	N° de Expediente	<input type="text"/>
Legajo	<input type="text"/> --Selecione--	Número de Póliza + Vigencia	<input type="text"/>

Figura 6: Captura del SIRCV. Rendición de Servicio

N°	Dependencia	Lugar	Fecha de iniciación
86	SANTA ROSA - CENTRO LABORAL Nº 57		05-05-2015
85	TELEN - ESCUELA Nº 9	SANTA ISABEL	05-05-2015
678320	DIRECCION DE PROGRAMAS DE SALUD	SANTA ROSA	04-05-2015
713300	DIRECCION DE PROGRAMAS DE SALUD	SANTA ROSA	04-05-2015

Figura 7: Captura del SIRC. Lista de Comisiones

4.3.1. Diseño de prototipos para requerimientos específicos

Vtiger provee herramientas de interfaz para dar soporte a la mayoría de las funcionalidades que abarca el desarrollo del SIRC. A partir de un script se puede crear un nuevo módulo y este ya contiene toda la interfaz para: listar todos los registros asociados, acceder o editar un registro en particular, crear uno nuevo o relacionar dos módulos distintos. Además tiene predeterminada la interfaz para los distintos popups que se necesiten y contiene las pantallas pertenecientes a la administración de usuarios, permisos y perfiles.

Existen requerimientos expuestos por el Tribunal de Cuentas que no pudieron ser cubiertos con las herramientas que brinda Vtiger, por esto fue necesario hacer prototipos para requerimientos específicos. Como es el caso del armado de la comisión, esto se debió maquetar en una pantalla aparte para que a través de jQuery (13) vaya armando dinámicamente los afiliados que forman parte de la comisión.

jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Es decir, esta tecnología permite interaccionar con el servidor sin necesidad de recargar la página ni enviar un formulario, genera un dinamismo que se quiso sostener en todo el SIRC.

Tal como se muestra en la Figura 8, el cuadro de mando superior permite elegir un nuevo afiliado y en el cuadro inferior se muestran los detalles de toda la comisión que forman parte de la solicitud o rendición del viático.

Otro requerimiento muy importante, es que la dependencia de Vialidad pueda elegir destinos para sus viáticos que no sean ciudades como el resto de las dependencias, sino que sean coordenadas exactas (puntos en las rutas provinciales y nacionales). Para ello se diseñó una interfaz específica que, a través de Google Maps, permita señalar los puntos destinos para viáticos confeccionados.

En total son dos pantallas uno de carga de destinos (Figura 9) y otra que permite la visualización de los puntos una vez que estos son creados y relacionados a la comisión.

Nº Afiliado	Nombre	Conv.	Días	Viático-Día	Viático-Total	Gastos Bs.	Gastos Serv.	Total	Menos Anticipo	Líquido a Cobrar	Líquido a Devolver
65990	Marcela Elizabeth Abraham	Personal Ley 643	1	404	404	0	0	404	0	0	0

Afiliados: 1 Total de Viáticos: 404 Total de Gastos Bs.: 0 Total de Gastos Serv.: 0 Total: 404 Menos Anticipo: 0 Líquido a Cobrar: 404 Líquido a Devolver: 0

Figura 8: Armado de comisión desde el módulo rendición

4.3.2. Ventajas de utilizar patrones de diseño de interfaz

Los patrones son conceptos ya probados, incluyen soluciones ya probadas en otros ambientes no teorías o especulaciones. No solo describen módulos, sino estructuras y mecanismos más profundos del sistema.

Contribuyen a reutilizar diseño gráfico, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. La importancia de la reutilización del diseño no es despreciable, ya que ésta provee de numerosas ventajas: reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad informática, la eficiencia y consistencia de diseños, y además proporciona un considerable ahorro en la intervención del desarrollador.

A su vez los patrones de diseño de interfaz proveen al proyecto de: confiabilidad (muchos componentes ya están probados), reusabilidad, mantenibilidad, interoperabilidad y usabilidad entre otros. La correcta medición de estas métricas genera satisfacción en el desarrollador y en el cliente, alargando notablemente la vida del software.

Los patrones de diseño que se utilizan en Vtiger garantizan una interfaz amigable, intuitiva, así como también aumentan la seguridad del sistema a través de la validación de los campos que el mismo patrón provee. Existen distintas vistas definidas: de edición, detalle, lista, popup; así como también se definen todos los tipos de campos que componen los formularios comunes para que el desarrollador utilice.

Es decir, a través de la utilización de los patrones de diseño en el sistema no se debieron crear desde cero cada una de las vistas, ni de los campos ni validaciones, esto redujo notablemente el tiempo de desarrollo y la concentración de la autora del proyecto en el correcto funcionamiento ya que el diseño estaba prácticamente solucionado.

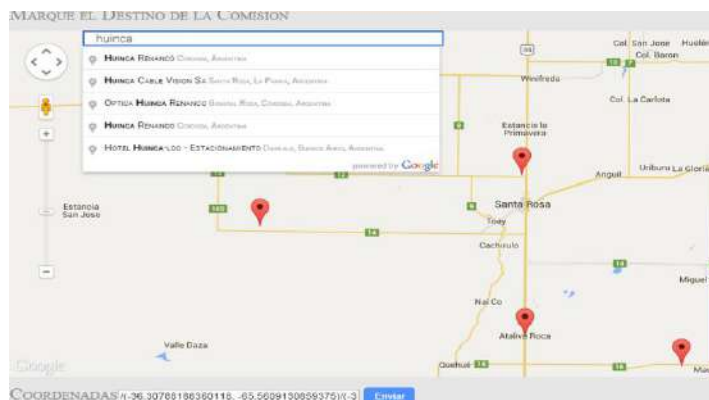


Figura 9: Pantalla perteneciente a la carga de destinos desde Google Maps

5. Elección de tecnologías para la implementación

En los capítulos anteriores se abordaron cuestiones teóricas y prácticas relacionadas al análisis y diseño de este proyecto. A continuación se detallarán las tecnologías y modelos seleccionados para la implementación del proyecto, sus funciones y las razones por las cuales fueron elegidas. Cabe aclarar que en el diseño ya se adelantó que la implementación fué realizada sobre la herramienta de código abierto Vtiger, pero el propósito de este capítulo es abordar con mayor profundidad las características de la misma y de su paradigma CRM.

Luego en el Capítulo 6 se profundizará sobre la implementación del proyecto.

5.1. CRM

Una de las tecnologías utilizadas para este proyecto es Vtiger CRM, antes de abordar sobre Vtiger es bueno definir a que se refiere la sigla CRM. En inglés “*Customer Relationship Management*” es un modelo de gestión orientado a la organización basada en la satisfacción del cliente. Según el sitio de la revista Gerencia (14) CRM promueve utilizar la tecnología para organizar, automatizar y sincronizar los procesos de negocio.

5.2. Vtiger CRM

Vtiger CRM es una aplicación CRM orientada a la web de código abierto. Esta herramienta ofrece flexibilidad y control de los procesos relacionados al cliente.

La versión base posee la administración de usuarios, perfiles, permisos y accesos para dar soporte a los distintos niveles jerárquicos en una organización. En esta versión también se encuentran módulos nativos de la herramienta: clientes, productos, presupuestos, etc. Además brinda herramientas para la creación de módulos, módulos relacionados, campos dentro de los módulos, informes, filtros, etc.

La estructura de Vtiger es modular y en las versiones más recientes está desarrollado sobre un patrón arquitectónico MVC. Tal como lo establece este patrón el código de cada sección se encuentra separado.

Al ser una tecnología de código abierto, permite la modificación de la misma para lograr adaptarla a las necesidades del proyecto. Esta ventaja forma parte de las características son parte de la decisión de utilizar Vtiger para este proyecto.

Otro de los beneficios que posee Vtiger, por los cuales el SIRCV es desarrollado sobre esta plataforma, es que integra muchas funcionalidades que fueron solicitadas en los requerimientos del proyecto. Esto permitió reutilizar código, sabiendo además que son módulos y funciones ya probadas en otros sistemas, dando garantías de integridad, seguridad y funcionalidad. Esto se puede garantizar, no solamente por la experiencia previa con la herramienta, sino también porque Vtiger es utilizado mundialmente por millones de empresas grandes y pequeñas.

Desde el sitio de Vtiger se puede acceder a las empresas más reconocidas que utilizan este CRM (15), así como también a todas las empresas colaborativas que participan en su desarrollo (16) .

Además, se puede garantizar la utilización mundial de esta herramienta si se inspeccionan las solicitudes de trabajos “freelance” en portales de trabajo como Núbelo (17) donde es muy común que se requieran programadores con experiencia en Vtiger o en la existencia de distintas comunidades donde se agrupan programadores Vtiger. La comunidad Vtiger de programadores españoles (18), con miles de suscriptos, presenta soluciones a problemas comunes o guía a los desarrolladores en las tareas de adaptación del sistema. A su vez existen en la web otras comunidades que agrupan a estos desarrolladores que no son españoles o distintos foros de consulta.

En la customización o adaptación de la herramienta para el sistema de control de viáticos se reutilizaron las funcionalidades relacionadas a la administración de usuarios y se desactivaron las otras relacionadas a la facturación por ejemplo. Pero Vtiger permite crear nuevos módulos de forma muy fácil, como por ejemplo se crearon los módulos Solicitud y Rendición. Igualmente, en el Capítulo 6 se profundizará en detalles de la codificación.

Vtiger está desarrollada por tecnologías del entorno web: PHP (19), HTML (20), Smarty (21), Javascript (22) y jQuery (13). Esta característica cumple otro de los requerimientos expuestos por el Tribunal de Cuentas de la Provincia.

Para la implantación de este proyecto se utilizó la versión 6.0, para poder ejecutar la misma se necesita tener instalado: Apache 2.0.40 como mínimo, PHP versión igual o mayor a 5.2, base de datos MYSQL versión igual o mayor a 5.1. El sistema operativo es independiente y se puede ejecutar tanto en Linux como en Windows.

5.3. Servidor Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Es un servidor modular y mucha funcionalidad está implementada por módulos externos que el programa principal carga durante su inicialización.

5.4. Phpmyadmin

Es una herramienta (23) escrita en PHP con la intención de manejar la administración de MySQL (24) a través de páginas web, utilizando Internet. Actualmente, puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves primarias y foráneas en tablas; administrar privilegios, exportar datos en varios formatos.

Una de las ventajas de utilizar Phpmyadmin, además de que es gratuito, es que provee de una interfaz muy intuitiva para manipular bases de datos. La misma facilita la ejecución de queries y la visualización de datos, de otra manera se deberían realizar mediante consola utilizando MYSQL.

5.5. Tecnologías relacionadas al versionado de código

Un controlador de versiones es un sistema que registra los cambios en un archivo o conjunto de archivos a través del tiempo para que se puedan recuperar versiones específicas de los mismos más adelante (25).

Existen diversas herramientas que facilitan la gestión de versiones llamados sistemas de control de versiones. Ejemplos de este tipo de herramientas son: Subversion (26), Git (27), entre otros.

El versionado de este proyecto fué realizado por Git, a continuación se detallan características del mismo y las ventajas de este versionado que se tuvieron en cuenta para su selección.

5.5.1. Git

Git es un controlador de versiones distribuido. Se dice distribuido porque cuando los usuarios descargan la última versión de los archivos no solo están descargando los archivos; sino también realizando una copia fiel y exacta (copia de seguridad) del repositorio (carpeta donde se alojan estos archivos) de esta manera si dicho repositorio muere, cualquiera de los usuarios que descargaron estos archivos pueden restaurar el estado del mismo haciendo uso de su copia de seguridad.

Por otra parte, esto permite que se puedan tener varios repositorios remotos para poder trabajar y colaborar con diferentes grupos de personas al mismo tiempo en un mismo proyecto, cosa que no es posible con los sistemas centralizados.

5.5.2. Fundamentos de Git

Instantáneas, no diferencias: Git modela los datos como un conjunto de estados de un mini sistema de archivos. Cada vez que se confirma un cambio, o se guarda el estado de un proyecto en Git, básicamente la herramienta hace una instantánea del aspecto de todos los archivos en ese momento, y almacena una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, Git no almacena el archivo nuevamente, sólo un enlace al archivo anterior idéntico que ya tiene almacenado.

Casi todas las operaciones son locales: Sí ya se descargó un repositorio no es necesario estar conectado a una red o conexión a internet para trabajar sobre los archivos, todo lo necesario se encuentra en la computadora local.

Integridad de los datos: Cada componente en Git es verificado mediante un checksum (suma de comprobación) antes de ser almacenado, y es identificado a partir de ese momento mediante dicha suma. De esta manera es imposible cambiar el contenido de cualquier archivo sin que Git lo note.

Git sólo añade información: Cuando se realizan operaciones en Git, generalmente sólo se añade información a la base de datos de Git. Esto asegura que el sistema no haga cambios que no se puedan deshacer, o que de algún modo borre información.

Trabaja con tres estados: Git tiene tres estados principales en los que se pueden encontrar los archivos: committed (confirmado), modified (modificado) y staged (preparado). Confirmado significa que los datos están almacenados de manera segura en la base de datos local. Modificado significa que se han realizado modificaciones en el archivo pero todavía no se han confirmado en la base de datos. Preparado significa que se ha marcado un archivo modificado en su versión actual para que evolucione a su próxima confirmación.

5.5.3. BitBucket

Para mantener un sistema de versiones, como se indicó anteriormente, se necesita un repositorio que funcione como servidor. Este servicio puede ser instalado en un servidor propio o se puede utilizar alguno, de los tantos, que son ofrecidos en la web de forma gratuita. El seleccionado para este proyecto es BitBucket.

BitBucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de revisiones Mercurial y Git. Posee muchas herramientas que facilitan las tareas al programador. Permite crear grupos de trabajo con distintos permisos de usuarios: solo lectura, escritura y lectura o administrador. Da soporte para la creación de incidencias y un chat interno. BitBucket permite visualizar fácilmente los cambios realizados en cada commit, crear branches, agregar comentarios y compartir el proyecto con otros usuarios.

Estas características facilitaron el trabajo a la hora de entregar el código fuente al Tribunal de Cuentas, como así también fue de mucha importancia ya que sirvió para ir discutiendo con el equipo de desarrollo del Tribunal.

A continuación se describe la dinámica de trabajo utilizando Git y BitBucket:

1. Se posee una copia del repositorio que gestiona BitBucket con un control de versiones Git y se trabaja de forma local utilizando Eclipse (28) como IDE de desarrollo.
2. Se crea un commit por cada sub-tarea y se realiza la operación mediante el cliente Git Tortoise Git. Al terminar un conjunto de commits asociados a un requerimiento específico se realiza un push al servidor Bitbucket (utilizando nuevamente el cliente Tortoise) .En la Figura 10 se muestran los comandos asociados a dicho paso que internamente ejecuta el cliente.
3. Se actualiza el servidor de producción del Tribunal de Cuentas que contiene SIRCV. Para realizar dicha tarea se realiza una conexión SSH con el servidor utilizando el cliente SSH Putty.

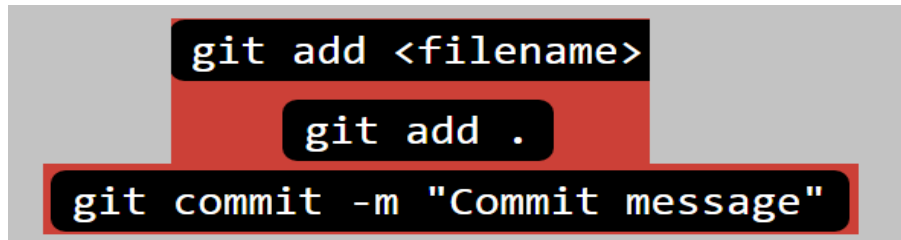


Figura 10: Comandos asociados a la gestión del repositorio de versiones

5.5.4. Tortoise Git

Es un cliente de código abierto para el sistema de control de versiones Git. Es decir, TortoiseGit gestiona los archivos a través del tiempo. Los archivos se almacenan en un repositorio local. El repositorio es como un servidor de archivos ordinario, salvo que recuerda todos los cambios hechos a sus archivos y directorios. Esto le permite recuperar versiones antiguas de sus ficheros y examinar la historia de cómo y cuándo cambiaron sus datos y quién lo cambió.

TortoiseGit se integra perfectamente en el shell de Windows (explorador), facilitando enormemente su uso, ya que navegando comúnmente a través de las carpetas se puede visualizar el estado de cada archivo y carpeta.

El estado de cada carpeta y archivo versionado se indica por pequeños iconos diferenciados según su estado: modificado, añadido, borrado o sin cambios. De esa manera se puede ver fácilmente el estado de su copia de trabajo.

Esta tecnología fue seleccionada por su facilidad de uso y porque permite gestionar las versiones desde Windows (sistema operativo del entorno de desarrollo), su uso es muy intuitivo y permite fácilmente comparar las copias de trabajo, realizar commits y push al servidor a través de su interfaz.

5.6. Putty

Es un cliente de red que soporta los protocolos SSH, Telnet y Rlogin y sirve principalmente para iniciar una sesión remota con otra máquina o servidor. Es de licencia libre y a pesar de su sencillez es muy funcional y configurable.

Esta tecnología fue seleccionada por ser una de las más populares para Windows que permiten el acceso remoto a otra máquina, además su interfaz es muy intuitiva y permite guardar datos de sesiones.

5.7. Entorno de desarrollo Eclipse

Eclipse es un IDE (Entorno de Desarrollo Integrado) tan potente como popular que incorpora un sin fin de utilidades para simplificar la labor de los programadores. Aparte de ser un entorno de desarrollo súper completo, una de las particularidades más interesantes para la comunidad es que es de código libre y gratuito.

Da soporte a varios lenguajes y tecnologías (JAVA, C++, PHP, Python, Android, Javascript) y está disponible para Linux, Mac y Windows. Eclipse dispone de un editor de texto con un analizador sintáctico.

Este IDE fue seleccionado por la experiencia de la autora del proyecto con este entorno frente a otros con las mismas prestaciones como es Netbeans (29). Utilizar un IDE facilita la codificación, ya que advierte errores de sintaxis, utiliza predicción para la utilización de métodos y clases, entre otros.

5.8. Google Maps APIS

Como se estableció en el capítulo anterior para la realización del requerimiento relacionado a la carga de destinos de comisiones en coordenadas específicas en un mapa se utilizó Google Maps conjuntamente con la API de Google (30).

En el Capítulo 4 se relacionó esta tecnología con el diseño de la interfaz pero también contiene una parte lógica que será plasmada en el próximo capítulo.

Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Google. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones.

Como en las aplicaciones web de Google, se usa un gran número de archivos Javascript para crear Google Maps. Como el usuario puede mover el mapa, la visualización del mismo se baja desde el servidor. Cuando un usuario busca un negocio, la ubicación es marcada por un indicador en forma de pin, el cual es una imagen PNG transparente sobre el mapa. Para lograr la conectividad sin sincronía con el servidor, Google aplicó el uso de AJAX dentro de esta aplicación para el desarrollo de mapas.

Google ofrece de forma gratuita una API con la que poder desarrollar aplicaciones a medida basadas en los mapas de Google, integrar los mapas en otras aplicaciones e incluso hacer *mash-up* (31) o mezclas de Google Maps y otras aplicaciones web que también disponen de una API pública.

Antes de utilizar la API de los mapas de Google, es necesario obtener una clave personal y única para cada sitio web donde se quiere utilizar. El uso de la API es gratuito para cualquier aplicación que pueda ser accedida libremente por los usuarios.

Las claves de la API de Google Maps consisten en una cadena de texto muy larga con un aspecto similar al siguiente:

“ABQIAAAA30JtKUU8se-7KKPRGSfCMBT2yXp_ZAY8_ufC3CFXhHIE1NvwkxRZNdns2BwZvEY-V68DvlyUYwi1-Q”.

Una vez obtenida la clave, cualquier página que quiera hacer uso de la API incorpora el siguiente código basado en JavaScript:

```
<script src="http://maps.google.com/maps?file=api&v=2&hl=es&key=ABQIAAAA30JtKUU8se-7KKPRGSfCMBT2yXp_ZAY8_ufC3CFXhHIE1NvwkxRZNdns2BwZvEY-V68DvlyUYwi1-Q" type="text/javascript"></script>
```

6. Codificación

En este capítulo se describen las tareas relacionadas a la codificación, realizadas luego de que los requerimientos fueron analizados y diseñados. Además a modo de introducción se especifican las tareas relacionadas para instalar el entorno de desarrollo utilizando las tecnologías mencionadas en el capítulo anterior. La puesta en marcha del entorno de producción será detallada en el Capítulo 7.

6.1. Entorno de desarrollo

El entorno de desarrollo es desarrollado en una computadora personal que actúa como cliente y servidor. Es decir, salvo por la funcionalidad de Google Maps, el entorno no necesita conexión de internet.

Vtiger puede ser ejecutado en sistemas operativos Linux o Windows, el entorno de desarrollo es ejecutado bajo Windows Seven.

Las descripciones del hardware del entorno de desarrollo son:

- Procesador: Inter® Core™ i5-3230M CPU @ 2.60GHz
- Memoria RAM : 6;00 GB DDR3
- Almacenamiento HD 500 GB + SSD 32 GB SATA 3 / On Board
- LAN 10/100Mb Base-TX Ethernet

La instalación del entorno de desarrollo consta de dos partes: el servidor y el cliente.

El cliente solo necesita un navegador web, este puede ser: Chrome (32), Firefox (33), Internet Explorer (34) u Opera (35).

Para la implementación del servidor se optó por instalar Xampp (36) v 5.6.11 ya que las características de Apache y Phpmyadmin que contiene esta versión cumplen con los requerimientos necesarios para instalar Vtiger.

Luego, se descargó la versión 6.2. Vtiger desde el sitio web (2) y se copió a la carpeta `htdocs` del servidor para su posterior ejecución.

Se nombra la carpeta descargada como “tribunal” y desde el navegador Chrome (que actúa como cliente) se ingresa a: <http://localhost/tribunal> .

Inmediatamente el navegador carga en la pantalla una imagen similar a la de la Figura 11 y se presiona la opción “Install”.

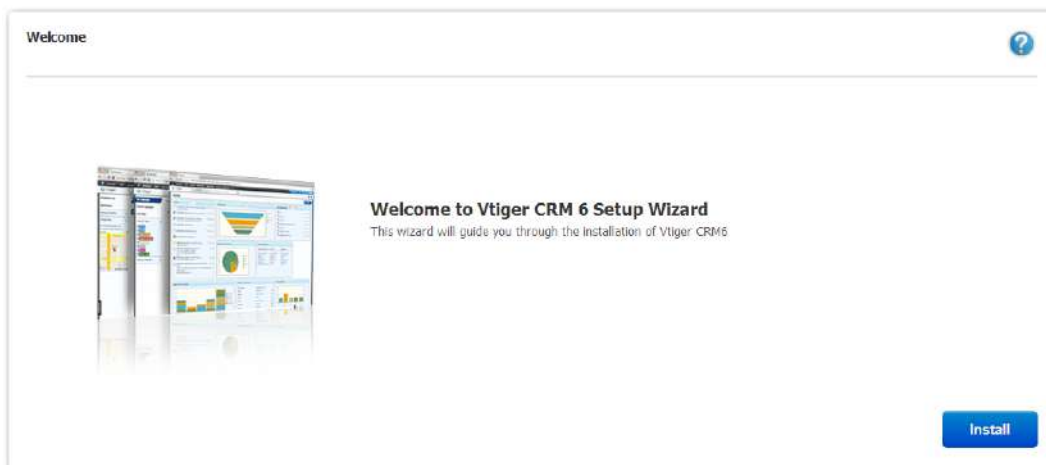


Figura 11: Primera pantalla de instalación Vtiger

Luego, como muestra la Figura 12 se chequean los requisitos para la instalación. Se advertirá con rojo aquellas características que no cumplan con los requerimientos.

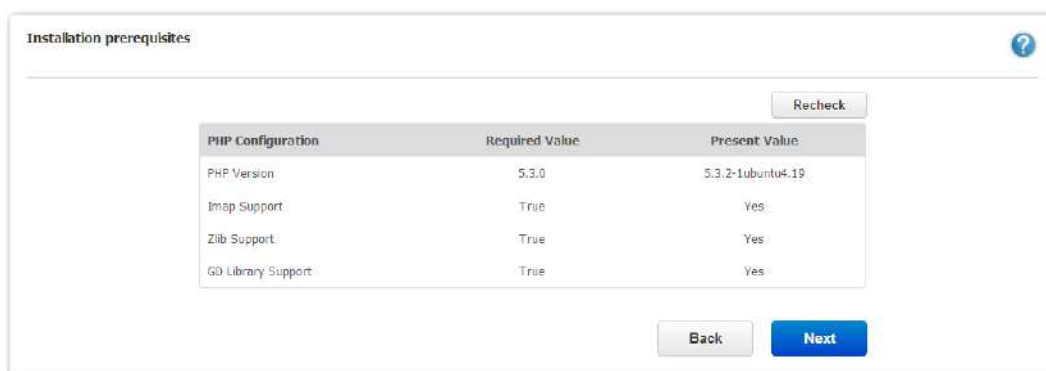


Figura 12: Requisitos para la instalación de Vtiger

Se establecen los parámetros de configuración como muestra en la Figura 13. En el panel izquierdo se deben ingresar los datos de la base de datos (que ya contiene el servidor XAMPP).

Esta configuración permite crear una nueva base de datos (se debe indicar el nombre) o seleccionar una existente (para ello se debe ingresar los datos del usuario root).

En el panel derecho se deben rellenar las opciones de configuración relacionadas al CRM. Como la contraseña del usuario admin, el nombre, la zona horaria y el formato de las fechas.

Figura 13: Parámetros de configuración de Vtiger

Luego de esta última pantalla, se deben esperar unos segundos y la instalación está completa. En realidad lo que genera esta instalación automática son todas las tablas, campos y datos en la base de datos; como así también toda la estructura de carpetas y archivos para tener el sistema de base. Este procedimiento es necesario hacerlo sólo una vez (para obtener la estructura inicial), luego cuando se migra a producción sólo se transfieren los archivos, se crea y copia la estructura de la base de datos y se modifica un archivo de configuración.

Además, antes de comenzar la codificación se necesita tener instalado un cliente Git, para este proyecto tal como se mencionara en el capítulo anterior, se utilizó Tortoise Git.

6.1.1. Versionado del código

Para comenzar con el versionado del código lo primero que se hace es crear un nuevo proyecto en BitBucket. Para ello se crea una cuenta a través de una cuenta de correo electrónico válida. Luego se accede a la opción Crear → Crear repositorio tal como muestra la Figura 14. Posteriormente se inserta el nombre del repositorio en este caso será “Sistema de Control de Viáticos”. Además se debe señalar la tecnología utilizada Git/Mercurial y la descripción del proyecto. Con estas operaciones se obtiene el servidor de versionado funcionando y accesible desde cualquier cliente Git (siempre y cuando se tengan las credenciales).



Figura 14: Creación de repositorio desde BitBucket

Luego desde el ambiente de desarrollo, en el servidor XAMPP mediante Tortoise Git se hace click derecho sobre la carpeta “tribunal” y se clona el repositorio que se creó en BitBucket pero está vacío. Para clonar el repositorio se necesita la url y las credenciales que provee BitBucket, en la Figura 15 se puede observar dicha acción.

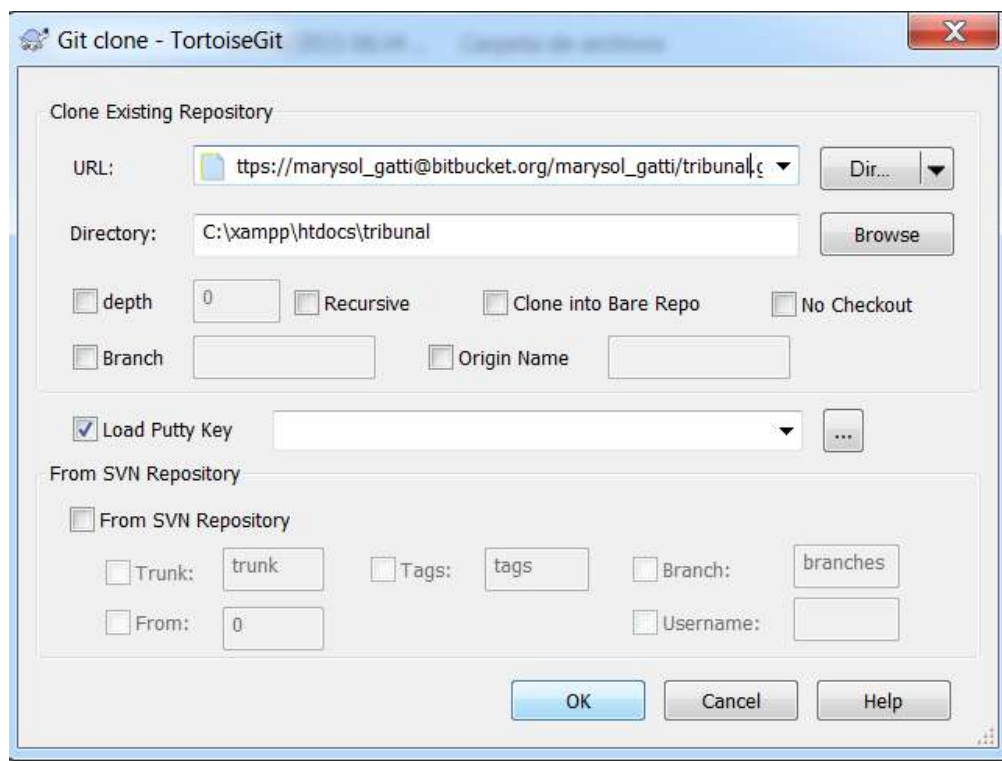


Figura 15: Clonación de repositorio desde el cliente Tortoise Git

Con este procedimiento la carpeta “tribunal” ya es un cliente versionado del proyecto Sistema de Control de Viáticos creado en BitBucket, para comprobar esto tiene que aparecer el ícono verde sobre la carpeta que demuestra que no hay ningún cambio tal como muestra la Figura 16.

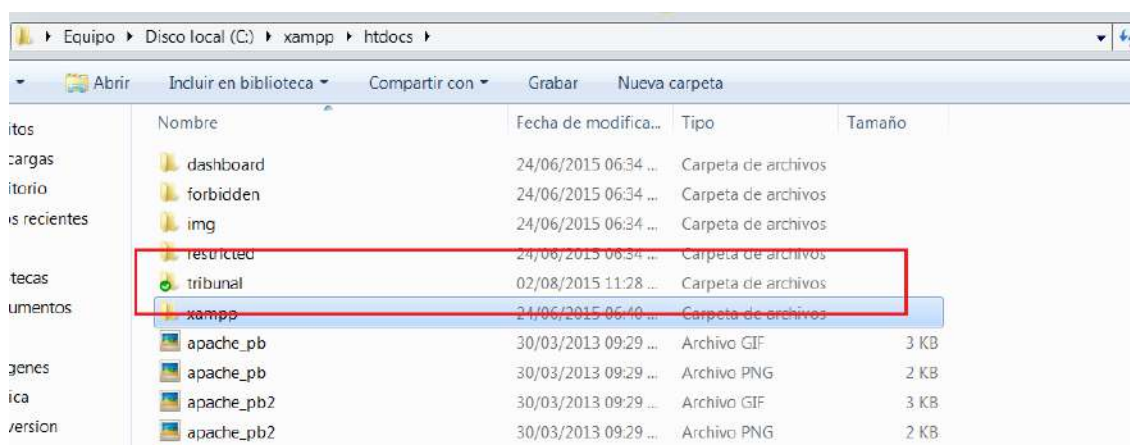


Figura 16: Sincronización de la copia local con el servidor de versiones

Luego se debe agregar el código base correspondiente al código Vtiger (aquél que se obtuvo con la realización de los pasos descritos en el apartado de la sección 6.1.1.) que se encuentra en la carpeta htdocs del servidor XAMPP del ambiente de desarrollo al proyecto Sistema de Control de Viáticos creado en BitBucket. Para ello se debe hacer click sobre la carpeta tribunal y se selecciona la opción Add.

Se agregan todos los archivos nuevos y se realiza el primer commit y push al servidor mediante el cliente Tortoise Git (la URL ya se encuentra grabada solo se necesitan las credenciales).

Después de todos estos pasos el cliente ya se encuentra sincronizado con el servidor de versiones, cada cambio realizado en cualquier archivo será advertido por el cliente mediante un ícono rojo que advierte que existen diferencias de código.

6.2. Codificación de funcionalidades

En la siguiente sección se explicarán las tareas de codificación realizadas sobre las principales funcionalidades. En su descripción se respeta el patrón arquitectónico seleccionado MVC.

6.2.1. Creación de módulos.

Los distintos módulos a realizar son aquellas entidades establecidas en el diagrama de entidad y relación diseñado y comentado en el Capítulo 4. Por cuestiones de funcionalidad se establecen como módulos primarios o principales a: Solicitud de Viáticos, Rendición de Viáticos y Control entre Solicitud y Rendición de Viáticos. Los módulos secundarios serán: Dependencia, Afiliado, Jurisdicción y Transporte. Perfil y Rol son módulos que ya vienen en la versión base de Vtiger.

Para la creación de cada módulo, Vtiger dispone de una herramienta que automatiza su creación. Es decir a través de un script ingresando el nombre del módulo y un campo crea los archivos necesarios, las tablas y datos para su estructura inicial.

Por ejemplo para la creación del módulo **Afiliado** primero se debe acceder al script que se encuentra en la ruta: `/htdocs/tribunal/vtlib/Tools/console.php`. Se selecciona la opción uno “Crear módulo” y se indica el nombre del módulo “Afiliado” y el campo principal “Número de afiliado”.

Dicho script inserta dos tablas en la base de datos cuyos nombres serán: `vtiger_afiliado` y `vtiger_afiliadocf`. Además insertará un registro en la tabla `vtiger_tab` dando de alta al módulo y asignándole un id, otro registro en la tabla `vtiger_profile2tab` dándole permisos al usuario admin para este módulo e inserta los campos de este módulo en la tabla `vtiger_field`. En este caso solo existe un nuevo campo.

Además el script creará una carpeta de nombre Afiliado en /htdocs/tribunal/modules y un archivo dentro de esta carpeta con el nombre Afiliado.php.

Por último, para que el módulo sea accedido se inserta en /htdocs/tribunal/parent_tabdata.php el número de módulo. Este archivo clasifica los elementos que contiene cada menú, para los nuevos módulos se creó un nuevo menú: "Tribunal". Entonces, el script agrega el número del módulo Afiliado en el menú Tribunal.

Las vistas de los módulos creados por el desarrollador, tienen soporte del idioma Ingles, para agregar el soporte a español se debe crear el archivo Afiliado.tpl y añadirlo en /htdocs/tribunal/languages/es_es. En dicho archivo se deben poner los valores contantes que aparecerán en las vistas desarrolladas con Smarty y su traducción en español. En la Figura 17 se observa el código de Afiliado.tpl, a medida que se necesite según las vistas se agregarán los elementos en dicho archivo.

```

*****/
$languageStrings = array(
    'Afiliado'                => 'Afiliado',
    'SINGLE_Afiliado'          => 'Afiliado',
    'LBL_ADD_RECORD'          => 'Añadir Afiliado',
    'LBL_RECORDS_LIST'        => 'Vista de Lista',
    'LBL_AFILIADOS'           => 'Afiliados',
    'LBL_AFILIADO_INFORMATION' => 'Información del Afiliado',
);
```

Figura 17: Template de soporte al español del módulo Afiliado

Vtiger define distintas vistas, las principales son: Edición, Detalle, Lista y Guardar. Si un módulo tiene una vista particular dichos archivos se encuentran en /htdocs/layout/vlayout/modules/nombre_del_modulo. Si el nombre del módulo no se encuentra en la lista de directorios Vtiger interpreta que utiliza la vista por defecto es decir los archivos que se encuentran en /htdocs/layout/vlayout/modules/Vtiger.

Es decir si se quiere definir una vista que se comporte de una forma particular, por ejemplo un EditView.tpl distinto para afiliado se debe crear el correspondiente Smarty en /htdocs/layout/vlayout/modules/Afiliado/EditView.tpl.

6.2.2. Módulos secundarios

Por cada módulo secundario se crearon los campos que se establecen en el diccionario de datos que se describió en el Capítulo 4. Para dicha acción se accede al creador de campos que dispone la herramienta y se añaden según el tipo. Hay tipos de campos que no están disponibles, por

ejemplo aquellos que relacionan un módulo con otro, pero el resto están disponibles y se separan por bloques tal como muestra la Figura 18. Además, se pueden crear nuevos bloques de campos.

Al igual que el script de creación de módulos el creador de campos inserta datos y en algunos casos tablas en la base de datos. Por ejemplo se agrega un nuevo campo que es “Apellido” de tipo Texto. Vtiger selecciona un nombre para este campo por ejemplo `cf_700` e inserta una nueva columna en la tabla `vtiger_afiliadocf` con este nombre y de tipo texto. Además agrega un registro en la tabla `vtiger_field` con el nombre de este campo, la relación al módulo y a la etiqueta “Apellido”. A su vez inserta un nuevo registro en la tabla `vtiger_profile2field` para agregar todos los permisos para el usuario **admin**.

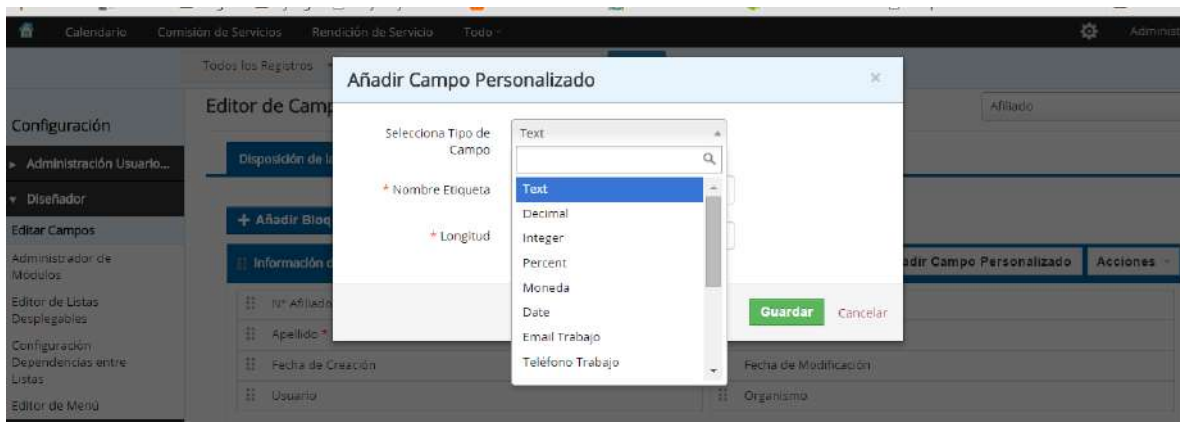


Figura 18: Interfaz que permite añadir campos y bloques

En los casos de tipo de campo Picklist (lista desplegable), crea una tabla con el nombre del campo por ejemplo `vtiger_cf_701` donde estarán todos los valores que se deben visualizar en la lista, junto con atributos de visibilidad y permisos.

Con respecto a la vista, es decir en los archivos creados con Smarty, no inserta nada ya que Vtiger tiene vistas generales que dan soporte a todos los tipos de campos.

6.2.2.1. Funcionalidades relacionadas al módulo Usuario

El módulo usuario, como ya se estableció, forma parte de los módulos secundarios pero a su vez es un módulo nativo de Vtiger. Es por esto que no se puede crear nuevos campos a través de la interfaz pero si se puede realizar por código.

Una de las funcionalidades que se pactó fue que al crear un usuario, si este es habilitante se ingrese el número de afiliado y al crearse el usuario genere automáticamente un nuevo afiliado

relacionado al usuario con la jurisdicción indicada en el primer módulo. Gracias a esto se ahorrará el tiempo en crear afiliados. No todos los usuarios son habilitantes y no todos los afiliados tendrán un usuario. Pero, se añadió también la posibilidad que teniendo un afiliado se lo pueda relacionar con un usuario que no tenga afiliado aún.

La codificación de esta tarea comienza creando un nuevo campo de tipo ckeck. Para ello se ejecuta el script de la Figura 19.

```

1  <?php
2  //se incluyen las clases necesarias
3  include_once('vtlib/Vtiger/Module.php');
4  include_once('vtlib/Vtiger/Menu.php');
5  include_once('vtlib/Vtiger/Block.php');
6  //Se trae una instancia de usuarios
7  $user = vtiger_module::getinstance('Users');
8  //Se trae un bloque de campos de usuarios
9  // donde se agregará el nuevo campo
10 $block = vtiger_block::getinstance('LBL_USER_INFORMATION', $custom);
11 // Se agrega el campo
12 $field = new Vtiger_Field();
13 $field->name = 'es_habilitante';
14 $field->label = 'Es Habilitante';
15 $field->table = 'vtiger_afiliadocf';
16 $field->column = 'es_habilitante';
17 $field->columnname = 'int(2)';
18 $field->uitype = 10;
19 $field->typeofdata = 'V~O';
20 $block->addField($field);
21 ?>

```

Figura 19: Fragmento de código relacionado a la creación de módulos

Luego se agrega en la capa lógica, el código necesario para que inserte un nuevo afiliado siempre y cuando el usuario sea habilitante. El código que se muestra en la Figura 20 se agrega en la función `save()` del archivo `/htdocs/tribunal/modules/Users/User.php`.

```

if ($this->column_fields['numero_afiliado'] != '') {
    $es_afiliado = "select * from vtiger_afiliadocf where cf_692=?";
    $result = $this->db->pquery($es_afiliado, array($this->column_fields['numero_afiliado']));
    $row = $this->db->fetchByAssoc($result);
    if ($row == NULL)
    {
        // Obtengo el id para insertar
        $query_insert = "select afiliadoid from vtiger_afiliado order by afiliadoid DESC";
        $result1 = $this->db->pquery($query_insert, array());
        $row1 = $this->db->fetchByAssoc($result1);
        $afiliadoid = $row1['afiliadoid'] + 1;
        //Inserto en afiliado, afiliadocf y crmentity
        $now = date("Y-m-d H:i:s");
        $user_id = $this->id;
        $query_afiliado = "insert into vtiger_afiliado values(?,?)";
        $result = $this->db->pquery($query_afiliado, array($afiliadoid, $this->column_fields['first_name']));
        $query_afiliadocf = "insert into vtiger_afiliadocf values('.$afiliadoid.', '.$this->column_fields['dni'].', '.$this->column_fields['numero_afiliado'].')";
        $this->db->pquery($query_afiliadocf, array());
        $query_crmentity = "insert into vtiger_crmentity (crmid, smcreatorid, smownerid, modifiedby, setype, createdtime, modifiedtime, presence, del)";
        $this->db->pquery($query_crmentity, array());
    }
    else{
        $query = "update vtiger_afiliadocf set usuario_asoc='".$this->id."' where cf_692=?";
        $result = $this->db->pquery($query, array($this->column_fields['numero_afiliado']));
    }
}

```

Figura 20: Fragmento de código que relaciona a un usuario con un afiliado

Con respecto a la codificación de la vista se agrega una funcionalidad en el Javascript que es invocado. Cuando el check “es habilitante” se activa el nuevo campo “Número de Afiliado” se desbloquea para permitir la carga del número del nuevo afiliado y de lo contrario estará inhabilitado para su edición. El código, que se visualiza en la figura, es insertado en /htdocs/tribunal/layouts/vlayout/modules/Users/resources/Edit.js.

Otra de las funcionalidades del módulo usuario es la posibilidad de que este pueda o no tener acceso a Google Maps. Solo si el usuario tiene activo el campo podrá cargar los destinos a través de Google Maps en la solicitud y rendición de viáticos. Entonces, se agrega el campo Google Maps en el módulo usuarios (con código similar a la figura 19) y luego en la vista de solicitud y rendición se filtra por este campo. Para realizar dicho filtro se realiza una parte en el código fuente del controlador que recibe los datos de los campos del modelo (Figura 21) y otro en la vista que recibe los datos del controlador (Figura 22).

```
if($moduleName == 'Solicitud' || $moduleName == 'Rendicion'){
    $data = Users_Record_Model::getCurrentUserModel();
    $google_maps = $data->get('google_maps');
    $viewer->assign('GOOGLE_MAPS', $google_maps);
}
```

Figura 21: Fragmento de código perteneciente a la capa lógica de solicitud y rendición

Dicha codificación debe realizarse para la vista y el controlador de edición y detalle.

```
{vtranslate($FIELD_MODEL->get('label'), $MODULE)}
{if $MODULE == 'Solicitud' && $FIELD_NAME == 'cf_728' && $GOOGLE_MAPS eq 1}
    <input style='margin-right:30px; margin-top:10px;' type='button' class="btn-lugar"
    value="Destino en Google Maps" onclick="return window.open('/tribunal/actions/destino_comision.html',
    'Comision', 'width=640,height=660,resizable=0,scrollbars=1');">
{/if}
```

Figura 22: Fragmento de código perteneciente a la vista de solicitud y rendición

6.2.3. Módulos primarios

La creación de módulos primarios: Solicitud de Viáticos, Rendición de Viáticos y Control de Viáticos se realiza mediante el script `console.php` como la creación de los módulos secundarios. Pero estos módulos llevan mayor codificación ya que están relacionados a las funciones principales del sistema.

Para la creación de los campos de Solicitud y Rendición se tuvieron en cuenta las planillas presentadas por el Tribunal de Cuentas de la Provincia (Anexo III y IV). Es decir, por ejemplo, que el módulo solicitud sea similar y contenga los mismos campos que la planilla presentada.

En los tres módulos primarios se realizaron relaciones entre el módulo y otro primario o secundario.

Figura 23: Módulo de control

Por ejemplo en el módulo Control se crearon relaciones con Solicitud y Rendición (la que se plasma en el modelo de entidad y relación). Entonces desde Control se puede acceder mediante un campo de tipo “popup” a todas las solicitudes y rendiciones pendientes de control. En la Figura 23 se observan los dos campos y en la Figura 24 el popup de solicitudes.

Figura 24: Visualización de rendiciones en el módulo control

Para la realización de dicha funcionalidad se implementaron dos scripts para que generen los campos y establezcan la relación entre los módulos. En la Figura 25 se observa el primer script que corresponde a la creación del campo Solicitud en el módulo control y en la Figura 26 se observa el script correspondiente a la relación entre Control y Solicitud.

```

7  <?php
8  /*
9   relacion 1:m (1 solicitud, x controls )
10  */
11  // define instances
12  $solicitud = vtiger_module::getinstance('Solicitud');
13  $control = vtiger_module::getinstance('Control');
14  $block = vtiger_block::getinstance('LBL_SOLICITUD_INFORMATION', $control);
15  // add field
16  $field = new Vtiger_Field();
17  $field->name = 'solicitud_id';
18  $field->label = 'Solicitud';
19  $field->table = 'vtiger_control';
20  $field->column = 'solicitud_id';
21  $field->columnname = 'varchar(255)';
22  $field->uitype = 10;
23  $field->typeofdata = 'V-O';
24  $block->addField($field);
25  $field->setrelatedmodules(array('Solicitud'));
26  ?>

```

Figura 25: Fragmento de código que permite la relación entre dos módulos

En la relación del ejemplo cada registro de control solo contiene una solicitud, una rendición y una dependencia. En cada uno de estos módulos existe en el panel derecho una etiqueta “Control Relacionado” que al acceder se puede visualizar el registro de control relacionado (Figura 27).

```

<?php

include_once('vtlib/Vtiger/Module.php');
$moduleinstance = vtiger_module::getinstance('Solicitud');
$festnetzmodule = vtiger_module::getinstance('Control');
$relationlabel = 'Control Relacionado';
$moduleinstance->setrelatedlist($festnetzmodule, $relationlabel, null, get_dependents_list);

?>

```

Figura 26: Fragmento de código que permite la relación entre módulos (cont)

Esta funcionalidad es aún más útil cuando la relación es de uno a muchos, en este caso solo se listará un registro.

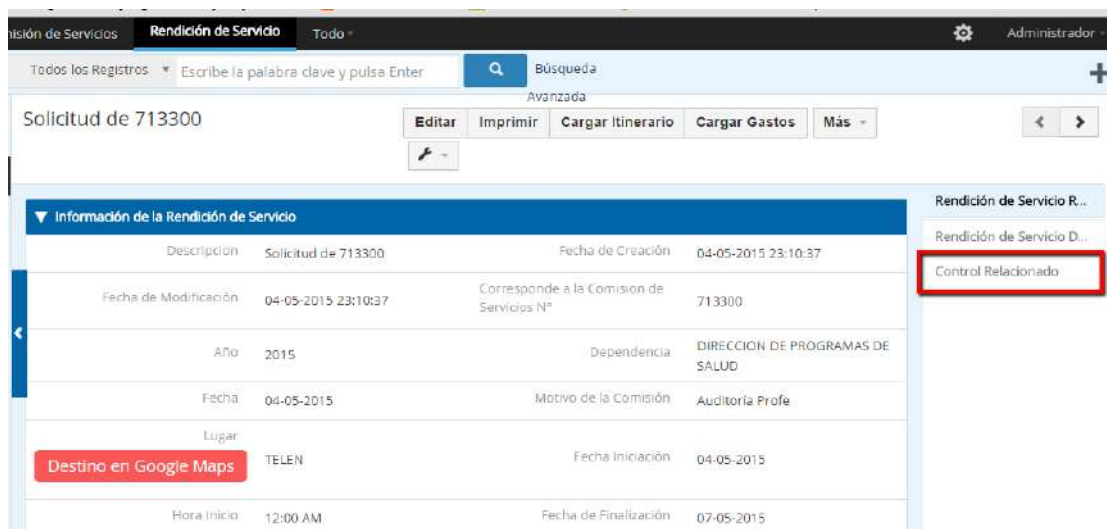


Figura 27: Interfaz que muestra la relación entre dos módulos

A continuación se detallan las tareas de codificación de algunas funcionalidades seleccionadas.

6.2.3.1. Creación de comisiones.

Como se estableció en la etapa de análisis cada solicitud y rendición tiene una comisión (conjunto de afiliados) relacionada. En muchas ocasiones puede suceder que cuando se solicita el viático se programa que viajan tres personas y luego viajan solamente dos y estos afiliados son los que se presentan en la rendición por ejemplo.

En el análisis se dio énfasis a todas las situaciones que se debían contemplar en el armado de comisiones y estas fueron plasmadas en el código.

Para permitir la carga de las comisiones en solicitudes y rendiciones se agregó un botón en el archivo `Edit.tpl` bajo la leyenda "Ingresar Afiliados", en la Figura 28 se observa el código. Además, en el archivo `/htdocs/tribunal/layout/vlayout/modulos/Solicitud/resources/Edit.js` se agregó una validación que este botón solo se podrá presionar cuando este cargada la fecha de inicio del viático y la duración.


```

<input style='margin-right:30px; margin-top:10px;' type='button' class='btn-lugar' value='Consultar Disponibilidad' onclick='jav
{/if}
{if $MODULE == 'Rendicion' && $FIELD_NAME == 'cf_1004' }
<input style='margin-right:30px; margin-top:10px;' type='button' class='btn-lugar' value='Consultar Disponibilidad' onclick='jav
{/if}
{if $MODULE == 'Solicitud' && $FIELD_NAME == 'cf_758' && $CON_RENDICION != 'true'}
<input style='margin-right:30px; margin-top:10px;' type='button' class='btn-lugar'
value='Ingresar Afiliados' onclick='return window.open('actions/index.php?usuario={$USER_ID}&clave={$CLAVE}', 'Comision');">
{/if}




```

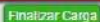

Figura 28: Fragmento de código de la vista de solicitud y rendición

Este botón, que abre un popup, invoca a un código externo a Vtiger. El mismo contiene una parte lógica, una relacionada al modelo y otra a la vista.

Para el desarrollo de la visualización se creó, utilizando HTML, los campos que se muestran en la Figura 29. Estos son los que se relacionan a la carga de afiliados.

DETALLE DEL PERSONAL QUE REALIZARÁ LA COMISIÓN DE SERVICIOS Y PLANILLA DE ANTICIPOS PARA VIÁTICOS Y GASTOS

Nº de Afiliado	65990  Cargar Datos de Afiliado
Apellido y Nombre	Mariela Elizabeth Abraham
Convenio	Personal Ley 643
Dentro/Fuera Provincia	Dentro de la Provincia
Recibe	--
Días	1  Cargar Datos Viáticos (en el caso de vialidad no contar el día de regreso, el sistema lo añade solo según los datos ingresados)
VIÁTICOS	
\$por día	505 %100
Total \$	505
GASTOS	
Bs. Ctes.	0  Sumar Gastos
Servicios	0
TOTAL ANTICIPO	505

 Finalizar Carga  Agregar

Nº Afiliado	Nombre	Conv.	Días	\$Viático-Día	Viático-Total\$	Gastos Bs.	Gastos Serv.	Total Anticipo
65990	Mariela Elizabeth Abraham	Personal Ley 643	1	505	505	0	0	505

Figura 29: Interfaz para el armado de comisiones

Para cumplir con los requerimientos y disminuir el error humano cuando el usuario carga el número de afiliado a través de jQuery se cargan todos los datos del mismo. Es decir al seleccionar la opción “Cargar Datos de Afiliado” se invoca a un archivo a través de jQuery que accede al servidor y trae la información del mismo de manera asíncrona sin dejar que el operador interactúe en el procedimiento. En las Figuras 30 y 31 se muestra el código de ambos.


```

header('Content-type: application/json');
include_once('myDBC.php');
$db = new myDBC();
$afiliadoid=$_REQUEST['afiliadoid'];
$sql = "SELECT name, cf_856, cf_694 from vtiger_afiliado inner join
vtiger_afiliadocf on vtiger_afiliado.afiliadoid=vtiger_afiliadocf.afiliadoid where cf_692=".$afiliadoid;
$res = $db->runQuery($sql);
$afiliado = $db->mostrar_datos($res);
print_r(json_encode($afiliado));
?>

```

Figura 30: Fragmento de código de la capa lógica para recuperar los datos de un afiliado

```

$('#agregarAfiliado').live("click",function() {
    afiliadoid=$('#valor_ide').val();
    $.getJSON('getAfiliados.php?afiliadoid='+afiliadoid,function(data) {
        if (data[0] != undefined){
            document.getElementById('valor_uno').value=data[0].name+ " " + data[0].cf_694;
            document.getElementById('valor_dos').value=data[0].cf_856;
            if(data[0].cf_856=='Personal Vial'){
                $("#tr_salida").show();
                $("#tr_regreso").show();
                $("#tr_un_dia").show();
                $("#datosV").show();
            }
            else{
                $("#tr_salida").hide();
                $("#tr_regreso").hide();
                $("#tr_un_dia").hide();
                $("#datosV").hide();
            }
            $("#agregar").show();
            $("#finalizar").show();
        }
        else{
            alert('El Número de Afiliado no Existe!!');
            $("#agregar").hide();
            $("#finalizar").hide();
        }
    });
});

```

Figura 31: Fragmento de código cliente para agregar afiliados a una comisión

Cuando el usuario que arma la comisión selecciona un afiliado válido debe seleccionar el tipo de viaje: Fuera o Dentro de la provincia, si recibe: alojamiento, alojamiento y comida, solamente comida o nada y por último ingresa la cantidad de días que dura el viático.

Esta acción llama a otro jQuery el cual invoca a un archivo PHP para que consulte al servidor cuanto dinero le corresponde por día al afiliado. Este importe dependerá de las opciones cargadas por el usuario, no será un campo editable y serán consultadas según lo registrado en el módulo de **Configuraciones de Viáticos**.

Se observa en la Figura 32 el código jQuery y en la Figura 33 parte del código de consulta al servidor. Se expone solo una parte de este último ya que las opciones son múltiples y se pueden originar muchísimas posibilidades según el tipo de afiliado, si el viático es fuera o dentro de la provincia, lo que recibe en el viático y a la dependencia que pertenece. Para cada combinación hay un importe de viático por día. Excluyendo a la dependencia vialidad que se tratará posteriormente.

```

function obtenerViaicoConvenio(convenio,provincia,recibio){
var xmlhttp = new XMLHttpRequest();
var valViat = null;
var url = 'obtenerViaicoConvenio.php?convenio='+convenio+'&provincia='+provincia+'&recibio='+recibio;
xmlhttp.onreadystatechange=function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
var arr = JSON.parse(xmlhttp.responseText);
$('#por_dia').val(arr.valor);
dinero_dia=$('#por_dia').val();
if (dias == 1){
$('#porcentaje').attr('disabled',false);
$('#por_dia').val((dinero_dia*$('#porcentaje').val()/100));
}
else{
$('#porcentaje').val(100);
$('#porcentaje').attr('disabled',true);
$('#por_dia').val(arr.valor);
}
$('#viatico_total').val(arr.valor*dias);
$('#total_anticipo').val(arr.valor*dias + parseInt($('#be').val())+ parseInt($('#servicios').val()));
}
}
xmlhttp.open("GET", url, true);
xmlhttp.send();
}

```

Figura 32: Fragmento de código cliente para obtener el viático por día en pesos

La forma de resolución de este requerimiento tiene muchas ventajas con respecto a la seguridad y la prevención de los casos de fraude. El usuario no asigna cuanto dinero le corresponde por día, el sistema según las opciones lo provee y no permite su edición.

Luego de este procedimiento ya es posible agregar el afiliado a la comisión, el sistema automáticamente calcula el total que solicitará o rendirá el afiliado. Al apretar el botón “Agregar” el sistema controla que el afiliado no pertenezca a otra comisión en el mismo periodo de tiempo. Si se trata de una solicitud advierte y si se trata de una rendición no permite agregar el afiliado.

```

if($convenio=='Funcionarios - Ley 643' and $provincia == 0 ){
    $sql = "SELECT cf_808 from vtiger_configuraciones_viatiscoscf";
    $res = $db->runQuery($sql);
    $row = $db->mostrar_datos($res);
    $viatico = $row[0]['cf_808'];
}
elseif($convenio=='Funcionarios - Ley 643' and $provincia == 1 ){
    $sql = "SELECT cf_822 from vtiger_configuraciones_viatiscoscf";
    $res = $db->runQuery($sql);
    $row = $db->mostrar_datos($res);
    $viatico = $row[0]['cf_822'];
}
elseif($convenio=='Justicia' and $provincia == 0 ){
    $sql = "SELECT cf_850 from vtiger_configuraciones_viatiscoscf";
    $res = $db->runQuery($sql);
    $row = $db->mostrar_datos($res);
    $viatico = $row[0]['cf_850'];
}
elseif($convenio=='Justicia' and $provincia == 1 ){
    $sql = "SELECT cf_854 from vtiger_configuraciones_viatiscoscf";
    $res = $db->runQuery($sql);
    $row = $db->mostrar_datos($res);
    $viatico = $row[0]['cf_854'];
}

```

Figura 33: Fragmento de código perteneciente a la capa lógica de la consulta de viático por día

Esta codificación también se realizó mediante jQuery y una consulta a un archivo PHP para que compruebe en el servidor la disponibilidad del afiliado. Se muestra en la Figura 34 fragmento de código correspondiente al código cliente codificado mediante jQuery.

```
function fn_agregar(){
    afiliadoid = $("#valor_ide").val();
    dia = $("#dias").val();
    fecha=window.opener.document.getElementById('Solicitud_editView_fieldName_cf_724').value;
    hora_inicio=window.opener.document.getElementById('Solicitud_editView_fieldName_cf_860').value;
    hora_fin=window.opener.document.getElementById('Solicitud_editView_fieldName_cf_862').value;
    viatico_dia=$("#por_dia").val();
    bs = $("#bs").val();
    serv = $("#servicios").val();
    clave = $("#clave").val();
    recibio = $("#total_anticipo").val();
    var viatico_regreso = 0;
    $.post('chequearAfiliado.php?afiliadoid='+afiliadoid+'&dia='+dia+'&fecha='+fecha+'&hora_inicio='+hora_inicio+'&hora_fin='+hora_fin,function(data){
        if(data>0){
            if(!confirm("Este Afiliado ya tiene otra comisión N° "+ data + " en el periodo seleccionado"))
                return false;
        }
    })
}
```

Figura 34: Código cliente que permite agregar un afiliado a una comisión

Cabe aclarar que se debe comprobar que el afiliado no esté en ninguna comisión relacionada a solicitudes y rendiciones, además de probar todas las posibilidades de que en el rango de fechas seleccionado no esté en otra comisión.

Por último, el sistema va agregando cada afiliado disponible a la comisión y calculando todos los subtotales. Cuando la carga es finalizada se confirma la comisión que realizará el viático.

En la Figura 35 se observa la pantalla de armado de comisiones. El cuadro superior permite ingresar nuevos afiliados y el cuadro inferior muestra los afiliados que ya forman parte de la comisión con los subtotales calculados.

Se realizó toda la codificación del proyecto, ya que fue programada en una capa superior a Vtiger, la mayor complejidad que presentó el desarrollo de esta tarea fue la obtención del valor de viático por día y la disponibilidad del mismo en el período establecido por el usuario, ya que ambas tareas tienen muchas variables y el código lo debe resolver rápidamente.

N° Afiliado	Nombre	Conv.	Días	Viático-Día	Viático-Total	Gastos Bs.	Gastos Serv.	Total Anticipos
09990	Mariela Elizabeth Abraham	Personal Ley 643	1	505	505	0	0	505
56629	DELGADO CARLOS ALBERTO DELGADO CARLOS ALBERTO	Funcionarios - Ley 643	2	750.00	1500	20	0	1510

AFILIADOS: 2 Totales Viáticos: 2005 Totales Gastos Bs: 20 Totales Gastos Serv.: 0 Totales Anticipos: 2015

Figura 35: Interfaz de confección de comisiones y cálculos de subtotales y totales

Como se ha especificado en este capítulo, se optó por usar funciones de jQuery: `$.getJSON(url)` y `$.post(url)` que permiten invocar desde el código cliente de una forma muy dinámica a archivos alojados en el servidor y así responder con información oportuna y actualizada. El usuario no tiene que enviar formularios ni recargar la página, todo el código fue diseñado para que el sistema sea lo más dinámico posible.

6.2.3.2. Codificación particular para dependencia Vialidad.

Como se advirtió en la planificación del proyecto la administración de viáticos para los afiliados que pertenecen a vialidad posee particularidades que deben ser resueltas por código. A continuación, se listan alguna de estas y la solución codificada.

- Existen más opciones a ser consideradas para el cálculo del viático por día: si pernocta o no en casilla, o si pernocta o no en casa. La solución fue realizada utilizando el lenguaje jQuery, donde desde la aplicación cliente se identifica cuando el campo convenio es “Personal Vial” y agrega estas opciones con campos de tipo radio(HTML). Además se agregan estas posibilidades en el jQuery que invoca al PHP para que contemple estas variables. En la Figura 36 se visualizan los nuevos campos.
- En vialidad no todos los afiliados de una misma comisión permanecen la misma cantidad de días realizando el viático. Por esto por cada afiliado se debe cargar la fecha de salida y la fecha de llegada. La solución planteada es una situación similar que el problema anterior por lo que se resolvió también con jQuery y añadiendo dichas opciones en la invocación del archivo PHP y en el propio archivo.

Figura 36: Campos añadidos para vialidad en la confección de comisiones

6.2.3.3. Control de vehículos en la solicitud y rendición de viáticos

Como se estableció en la planificación y se consideró en el diseño que una comisión puede utilizar dos vehículos. Se debe comprobar la disponibilidad de ambos. Por esto se agregó en la vista `Edit.tpl` de Solicitud y Rendición un botón bajo la leyenda “Consultar Disponibilidad” como muestra la Figura 37.

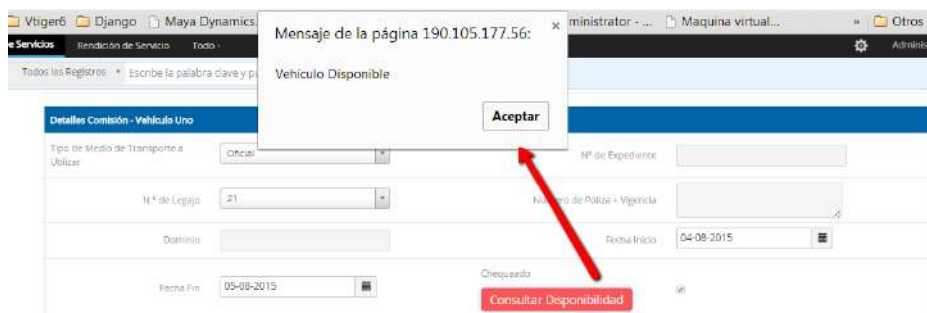


Figura 37: Interfaz de consulta de disponibilidad de los vehículos

Esta acción chequea un campo no editable que obligatoriamente debe estar activado para poder generar una solicitud o una rendición. Esto fue realizado con código cliente mediante la tecnología jQuery.

Es decir, para que el sistema te permita guardar una solicitud o una rendición se debe comprobar la disponibilidad de uno o, en el caso que se utilicen dos vehículos, de los dos. En la Figura 38 se observa el código de dicha validación añadido en la vista de solicitud y rendición. Mediante jQuery se pudo evitar que se consulte nuevamente en el servidor, simplemente comprueba que los campos estén de modo activo.

```
$( "#guardar" ).click(function() {

if(($('input[name="cf_880"]').is(':checked') && (jQuery("select[name='cf_894']").val()=="")
|| (jQuery("select[name='cf_894']").val()!=" " && $('input[name="cf_908"]').is(':checked')) ) {
return true;
}
else{
alert("El/Los Vehiculo/s no esta/n Chequeado/s");
return false;
}
})
```

Figura 38: Validación de control de disponibilidad de vehículos en solicitudes y rendiciones

6.2.3.4. Destino de comisiones a través de Google Maps

Tal como se planteó en el análisis y en el diseño existen algunos usuarios que deben cargar el/los destinos de viáticos a través de coordenadas. La propuesta que se realizó al Tribunal de Cuentas para resolver esto es mediante Google Maps, por lo pronto existen dos lógicas a desarrollar: la carga de destinos y la visualización de los mismos.

Lo primero es agregar los botones en las vistas de edición y detalle de los módulos solicitud y rendición. Esto se realiza a través de Smarty contemplando que el usuario tenga permiso (esto fue detallado en la sección 6.2.2.1).

Luego estos botones abren un popup que invoca a un código HTML externo al código Vtiger. Dicho archivo contiene código Javascript y además invoca a las librerías de Google. Para la utilización de estas librerías se utiliza la clave obtenida por una cuenta de Google registrada por el Tribunal de Cuentas, tal como se especifica en la sección 5.8. de este informe.

Mediante la librería de Google se puede obtener un mapa y situarlo en coordenadas específicas. En la Figura 39 se observa dicho código.

```
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false&libraries=places"></script>
<script type="text/javascript">
    function initialize() {
        var markers = [];
        //Creo un nuevo mapa situado en Santa Rosa, La Pampa, Argentina, con 13 de Zoom y del tipo ROADMAP
        var mapa = new google.maps.Map(document.getElementById("map_canvas"),{center: new google.maps.LatLng(-36.620436,-64.290665),
        zoom: 13,mapTypeId: google.maps.MapTypeId.ROADMAP});
```

Figura 39: Fragmento de código para la creación del mapa

Con respecto al código relacionado a la carga de puntos específicos como destinos de comisiones (Figura 40), la librería de Google ofrece métodos para la señalización y la obtención de la coordenada indicada a través de lo que ellos denominan “marcadores” (iconos de señalización con forma de globo de color rojo).

```
//Creo un evento asociado a "mapa" cuando se hace "click" sobre el
google.maps.event.addListener(mapa, "click", function(evento) {
    //Obtengo las coordenadas separadas
    var latitud = evento.latLng.lat();
    var longitud = evento.latLng.lng();

    //Puedo unir las en una unica variable si asi lo prefiero
    var coordenadas = evento.latLng.lat() + ", " + evento.latLng.lng();

    //Creo un marcador utilizando las coordenadas obtenidas y almacenadas por separado en "latitud" y "longitud"
    var coordenadas = new google.maps.LatLng(latitud, longitud); /* Debo crear un punto geografico utilizando google.maps.LatLng */
    document.getElementById('coordenadas').value=document.getElementById('coordenadas').value+"/"+coordenadas;
    var marcador = new google.maps.Marker({position: coordenadas,map: mapa, animation: google.maps.Animation.DROP, title:"Un marcador cualquiera"});
}); //Fin del evento
```

Figura 40: Código cliente para dar soporte a los marcadores en el mapa

Este código cliente guarda las coordenadas seleccionadas en un campo de tipo texto que luego la capa lógica dentro de la función `save()` invoca al modelo para su inserción en la base de datos.

Para el desarrollo del código de visualización de los puntos, es decir una vez que ya se establecieron y se quiere volver a abrir el mapa. El sistema toma del campo de tipo texto que almacena las coordenadas y las “parsea” para su correcta visualización, con las mismas tecnologías que la carga.

En la Figura 41 se visualiza el armado de los marcadores dentro del mapa según los puntos almacenados.

```
function initialize() {
    var markers = [];
    var i = 0;
    //var latitud = new Array();
    //var longitud = new Array();
    var latitud = [ <?php echo implode(' ', $lat). ' ' ?> ];
    var longitud = [ <?php echo implode(' ', $long). ' ' ?> ];
    //Creo un nuevo mapa situado en Santa Rosa, La Pampa, Argentina, con 13 de Zoom y del tipo ROADMAP
    var mapa = new google.maps.Map(document.getElementById("map_canvas"), {center: new google.maps.LatLng(latitud[0], longitud[0]), zoom: 5, mapTypeId: google.MAP_TYPE_ROADMAP});
    while(latitud[i]){
        var coordenadas = new google.maps.LatLng(latitud[i], longitud[i]); /* Debo crear un punto geografico utilizando google.maps.LatLng */
        var marcador = new google.maps.Marker({position: coordenadas, map: mapa, animation: google.maps.Animation.DROP, title:"Un marcador cualquiera"});
        i++;
    }
    //Creo un marcador utilizando las coordenadas obtenidas y almacenadas por separado en "latitud" y "longitud"
    var input = /** @type {HTMLInputElement} */(
        document.getElementById("pac-input"));
    mapa.controls[google.maps.ControlPosition.TOP_LEFT].push(input);
    var searchBox = new google.maps.places.SearchBox(
        /** @type {HTMLInputElement} */(input));
```

Figura 41: Visualización de los marcadores en el mapa

6.2.3.5. Configuración de los montos de viáticos según variables.

Como se estableció la planificación y en el diseño, el sistema necesita contener una sección para que el usuario final configure los valores de viáticos por día según las distintas combinaciones de: tipo de afiliado, pernoctación, horario de salida, horario de llegada y dependencia.

Para ello se agregó en la configuración de Vtiger una sección específica para este requerimiento. Como se observan en las Figuras 42 y 43 el usuario puede predeterminar cada valor en pesos, esta planilla es muy similar a la que actualmente maneja el personal del Tribunal de Cuentas.

▼ CONFIGURACIONES VIATICOS			
Acceder a la Configuración de Viáticos	Configuración de Viáticos	Asignado a	Administrador Administrator
Fecha de Creación	20-10-2014 23:19:45	Fecha de Modificación	21-04-2015 17:58:30
▼ Común a todos los convenios			
Comida	20.00	Alojamiento	40.00
Comida y Alojamiento	30.00		
▼ Personal Vial			
Valor Viático móvil fuera de Provincia	100	Valor Viático móvil dentro de Provincia	200
Valor Viático fijo para casilla	200	Valor Viático fijo para dependencias o casas	100
Valor Desarralgo	300	Dentro de la Pcia - Salida(Antes de las12)Viatico	497
Dentro de la Pcia - Salida(Antes de las12) Casa	298	Dentro de la Pcia - Salida(Antes de las12) Casilla	323
Dentro de la Pcia - Salida(Dspde las12) Viatico	348	Dentro de la Pcia - Salida(Dsp de las12) Casilla	226
Dentro de la Pcia - Salida(Dsp de las12) Casa	209	Dentro de la Pcia - Regreso -13 30 y 18	149
Dentro de la Pcia - Regreso - Dsp 18	298	(1 día) Dentro de la Pcia - antes de 12 - dsp 18	298

Figura 42: Módulo de configuración de los montos de viático

Esta sección solo es accedida por el perfil administración ya que contiene información muy sensible. Vale aclarar que las figuras no contienen datos verdaderos.

Con respecto a la codificación de este requerimiento se agregaron los campos en la vista del módulo configuración mediante tecnología Smarty y en la capa lógica se añadió el código necesario para guardar los cambios en la base de datos de estos campos. Además, en la base de datos se creó una nueva tabla: vtiger_configuracion_viaticos para que soporte dicha estructura.

Dentro de la Pcia - Regreso - Dsp 18	298	(1 día) Dentro de la Pcia - antes de 12 - dsp 18	298
(1 día) Dentro de la Pcia - Resto	149	Fuera de la Pcia - Día Salida - antes 12	327
Fuera de la Pcia - Día Salida - dsp 12	516	Fuera de la Pcia - Día Regreso - entre 13 30 y 18	221
Fuera de la Pcia - Día Regreso - dsp 18	442	(1 día) Fuera de la Pcia antes de 12 - dsp 18	442
(1 día) Fuera de la Pcia - Resto	221		
▼ Convenio Justicia			
Fuera de la Provincia - día	999.99	Dentro de la Provincia - día	200
▼ Convenio 543			
Fuera de la Provincia - Día Entero - Funcionario	720	Fuera de la Provincia - Día Entero - Afiliado	300
Dentro de la Provincia - Día Entero - Funcionario	420	Dentro de la Provincia - Día Entero - Afiliado	210
▼ Salud Pública			
Fuera de la Provincia - Día Entero	25	Dentro de la Provincia - Día Entero	10

Figura 43: Módulo de configuración de los montos de viático (cont)

6.2.3.6. Módulo de control

Tal como se especificó en capítulos anteriores el módulo de control tiene como propósito comparar las solicitudes y rendiciones. Es decir, controla que los campos con información más sensible de una solicitud y de una rendición de una misma comisión coincidan. Cuando esto no ocurre el sistema debe alertar de forma automática.

Para mantener la seguridad del control, estos registros no pueden ser creados ni editados por el usuario solo se crean automáticamente cuando se genera una nueva rendición (el momento oportuno para controlar). El código relacionado a dicha función se añadió en la función `save()` del módulo Rendición y una fracción del mismo es el que se observa en la Figura 44.

```
$diferencias=1;
if($totalS==$this->get('cf_1054') && $vehiculos1==true && $cantAfiliados==$cantAfiliadosR)
$diferencias=0;

$cantAfiliadosSolicitud = "SELECT COUNT( afiliadoid ) AS cantidad FROM vtiger_afiliado2solicitud WHERE solicitudid =?";
$cant = $db->pquery($cantAfiliadosSolicitud,array($this->get('solicitud_id')));
$cantAfiliados = $db->query_result($cant, 0, "cantidad");

$cantAfiliadosRendicion = "SELECT COUNT( afiliadoid ) AS cantidad FROM vtiger_afiliado2rendicion WHERE rendicionid =?";
$cantR = $db->pquery($cantAfiliadosRendicion,array($rendicion));
$cantAfiliadosR = $db->query_result($cantR, 0, "cantidad");

$sql2 = "insert into vtiger_control(name,controlid,dependencia_id, solicitud_id, rendicion_id) values(?,?,?,?)";
$result = $db->pquery($sql2,array($motivo,$controlid,$this->get('dependencia_id'),$this->get('solicitud_id'),$rendicion));
$sql3 = "insert into vtiger_controlcf(controlid,cf_1063,cf_1065,cf_1067,cf_1069,cf_1071,cf_1073,cf_1075,cf_1077,cf_1082)
values (". $controlid.", ". $vehiculo1Solicitud.", ". $vehiculo2Solicitud.", ". $vehiculo1Rendicion.", ". $vehiculo2Rendicion.",
". $cantAfiliados.", ". $cantAfiliadosR.", ". $totalS.", ". $this->get('cf_1054').", ". $diferencias." )";
$result = $db->pquery($sql3,array());
$now = date("Y-m-d H:i:s");
$query_crmentity = "insert into vtiger_crmentity (crmid,smcreatorid,smownerid,
modifiedby,setype,createdtime,modifiedtime,presence,deleted,label)
values (". $controlid.", 1,1,1, 'Control', ' ". $now.", ' ". $now.", 1,0, 'Control' )";
$db->pquery($query_crmentity, array());
$sql4 = "update vtiger_crmentity_seq set id=?";
$result = $db->pquery($sql4,array($controlid));
```

Figura 44: Fragmento de código que crea un registro de control luego de la rendición

La información que se inserta en el módulo control, según los campos establecidos en el diccionario de datos, permite alertar o no de posibles fraudes.

El primer bloque del módulo de control establece la solicitud y la rendición relacionada y la dependencia a la que pertenecen (Figura 45).

▼ Información del Control			
Motivo	Auditoría Profe	Revisado Por:	Administrador Administrator
Fecha de Creación	04-05-2015 23:10:38	Fecha de Modificación	04-05-2015 23:30:07
Dependencia	DIRECCION DE PROGRAMAS DE SALUD	Solicitud	713300
Rendicion	Solicitud de 713300		

Figura 45: Primer bloque del módulo de control

El segundo bloque compara el/los vehículos utilizados (Figura 46)

▼ Información sobre los Vehículos			
Vehículo uno - Solicitud	Legajo 2497	Vehículo dos - Solicitud	
Vehículo uno - Rendición	Legajo 2497	Vehículo dos - Rendición	

Figura 46: Comparación de los vehículos de una solicitud y una rendición de un mismo viático

El tercer bloque compara el/los afiliados que conformaron la comisión (Figura 47)

▼ Información sobre Afiliados			
Cantidad de Afiliados - Solicitud	2	Cantidad de Afiliados - Rendición	1

Figura 47: Comparación de la cantidad de afiliado entre una solicitud y una rendición de un mismo viático

Por último, el bloque final contiene información sobre el dinero, y una lógica adicional sobre si el registro de control ya fue revisado por el personal a cargo del Tribunal de Cuentas.

Visualmente al usuario se lo alerta pintando en rojo, en la vista de lista de controles, aquellos registros que tienen diferencias entre la solicitud y rendición (Figura 48).

Todos los Registros

Escribe la palabra clave y pulsa Enter

Busqueda

Avanzada

+

Acciones

Todos Control

1 to 2

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Motivo	Solicitud	Rendicion	Con Diferencias
Reunión de ministros	678320	RENDICION DE 678320	no
Auditoria Profe	713300	Solicitud de 713300	yes

Figura 48: Interfaz de lista de registros de control

Para el desarrollo de las funcionalidades de control se realizaron diversas tareas. Por un lado se creó el módulo a través de la herramienta que provee Vtiger, luego se crearon los campos mediante la interfaz de Vtiger y por último se realizaron los scripts para la creación de los campos de tipo “popup” que relacionan al módulo control con: **Dependencia, Solicitud y Rendición.**

Luego en la vista de lista de registros se añadió el código para que pinte la fila según el registro tenga o no diferencias (Figura 49).

```

{if $MODULE == 'Control' && $LISTVIEW_HEADER->getFieldDataType() == "boolean" }
{assign var="idControl" $LISTVIEW_ENTRY->getId()}
{if $LISTVIEW_ENTRY->get($LISTVIEW_HEADERNAME) == "yes"}
{literal}
<script type="text/javascript">
var newVar = "{/literal}{$LISTVIEW_ENTRY->getId()}{literal}";
$("#id"+newVar).css('background-color', '#F78181');
</script type="text/javascript">
{/literal}
{else}
{literal}
<script type="text/javascript">
var newVar = "{/literal}{$LISTVIEW_ENTRY->getId()}{literal}";
$("#id"+newVar).css('background-color', '#81F781');
</script type="text/javascript">
{/literal}
{/if}
{/if}

```

Figura 49: Fragmento de código que pinta la lista de registros de control

En la capa lógica se añadió el código necesario para que este módulo en particular no permita crear registros ni editar, solamente posibilita a añadir observaciones.

6.2.3.7. Tareas añadidas al cron

Vtiger provee de herramientas para agregar fácilmente tareas en el cron del servidor.

En sistemas operativos Unix, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes) (37).

Para agregar estas tareas que se ejecuten en segundo plano cada un intervalo de tiempo definido primero se debe crear a través de la interfaz del SIRCV, ya que en estas instancias de codificación ya no es solamente un “Vtiger” sino que se trata de un sistema integrado, un flujo de trabajo.

Los flujos de trabajo son una serie de acciones automatizadas que pueden desencadenar sobre la base de los comportamientos de una persona o la información de un registro. Permite enviar mails o realizar una tarea específica según una condición dada.

El Sistema de Control de Viáticos contiene dos flujos de trabajo:

- Notificar a quien corresponda cuando un registro de control es revisado.
- Enviar un mail por cada nueva rendición creada.

En la Figura 50 se puede observar la interfaz que permite la creación de uno de los flujos de trabajo. Este procedimiento consta de tres partes: programación del flujo de trabajo, condiciones y tarea.

Figura 50: Flujo de Trabajo

Para añadir estos flujos de trabajo en el cron del servidor se debieron realizar varias tareas de codificación y configuración.

1. Editar el archivo `/htdocs/tribunal/vtigercron.php` y cambiar la condición `php_sapi` (que anteriormente estaba como `"cli"`) al valor `php_cli` del servidor.
2. Editar el archivo `/htdocs/tribunal/cron/vtigercron.sh` y descomentar la línea para que permita la creación de logs en un archivo específico.
3. Crear el archivo de log.

Es decir, se debieron realizar cambios para que funcione la herramienta y por cuestiones de seguridad y funcionabilidad se agregó un log que registre cada ejecución de un flujo de trabajo.

En el capítulo posterior se implementará el cron en el servidor Linux, en esta etapa se explican los pasos necesarios para poder implementarlo correctamente.

6.2.3.8. Impresión de documentos, cargar itinerario y gastos

Dentro de las funcionalidades requeridas y planteadas en los primeros capítulos, el Tribunal de Cuentas de la provincia de La Pampa solicitó que el SIRCV permita en la rendición de los mismos realizar la carga del itinerario y de los gastos realizados.

En el Anexo V se observan las planillas presentadas por el Tribunal, las cuales se trataron de imitar en la codificación por cuestiones administrativas y para acercar al sistema con el usuario.

Primero se agregó un botón por cada acción en el archivo `.tpl` (realizado con Smarty) de la vista del detalle de rendición (es decir una vez que esta guardada), luego se programó la capa lógica para que realice dicha función y consulte al modelo los datos a imprimir.

La impresión de solicitudes, rendiciones, planilla de gastos e itinerario se realizó mediante Fpdf (38) una librería de PHP que genera archivos de extensión PDF. En la Figura 51 se observa un fragmento de dicho código.

La librería seleccionada para la creación de los archivos PDF permite la elección de la unidad de medida, el formado de páginas y márgenes. Además gestiona las cabeceras, pies y saltos de páginas. Todas estas características permiten que todos los documentos sean similares a los presentados por el tribunal, sobre todo porque en la mayoría se necesitaron confeccionar tablas. En el Anexo VI se observa uno de los documentos generados por el sistema.

```
$sql = "SELECT vtiger_solicitudcf.solicitudid,vtiger_dependencia.name as dependencia,cf_896,cf_898,cf_900,cf_902,cf_718,cf_720
,cf_722,cf_724,cf_726,cf_728,cf_732,cf_734,cf_736,cf_864, cf_772, cf_774, cf_776, cf_778, cf_780, cf_782,cf_784,cf_786, cf_788,
cf_790,cf_792,cf_794 from vtiger_solicitud inner join vtiger_solicitudcf on vtiger_solicitud.solicitudid=vtiger_solicitudcf.solicitudid
left join vtiger_dependencia on vtiger_dependencia.dependenciaid=vtiger_solicitudcf.cf_766 where vtiger_solicitud.solicitudid=".$solicitudid;
$res = $db->runQuery($sql);
$row = $db->mostrar_datos($res);
$dependencia = $row[0]['dependencia'];
$lugar = $row[0]['cf_718'];
$fecha = $row[0]['cf_720'];
$fecha = date("d-m-Y", strtotime($fecha));
$fecha_inicio = date("d-m-Y", strtotime($row[0]['cf_724']));
$pdf = new PDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(-5,10,'REPUBLICA ARGENTINA');
$pdf->Cell(0.001,30,'GOBIERNO DE LA PROVINCIA DE LA PAMPA');
$pdf->SetFont('Arial','',15);
$year=date("Y");
$pdf->Cell(0.001,60,'SOLICITUD AUTORIZACIÓN - COMISIÓN DE SERVICIOS N° AÑO:'. $year);
$pdf->SetFont('Arial','B',12);
$pdf->Cell(34,80,'Lugar y Fecha : ');
$pdf->SetFont('Arial','',12);
$pdf->Cell(-34,80,$lugar.' , '.$fecha);
$pdf->SetFont('Arial','B',12);
$pdf->Cell(34,100,'Dependencia : ');
$pdf->SetFont('Arial','',12);
```

Figura 51: Fragmento de código perteneciente a la impresión de archivo PDF

6.2.3.9. Restricción de acceso a jurisdicciones

Otro de los requerimientos que debieron ser codificados fue la restricción de jurisdicciones para los usuarios finales. Es decir, según el tipo de rol del usuario tenga permisos para crear y visualizar solicitudes y rendiciones de determinadas jurisdicciones.

Cada usuario tiene un rol, ese rol tiene permiso para determinadas jurisdicciones. Excepto aquellos usuarios administradores o de tipo “jefe de relatores” que pueden crear y ver documentos de todas las jurisdicciones. Cada jurisdicción tiene 1 o varias dependencias. Entonces cuando se crea una solicitud o una rendición se controla según el rol del usuario y se muestran las dependencias que correspondan.

Por ejemplo el popup que se despliega al seleccionar una dependencia desde rendición o solicitud, para un usuario que solo tiene permiso a una jurisdicción, solamente accederá a las dependencias de la misma.

Vtiger no tiene soporte para esto, la vista debió ser modificada para que solamente se visualicen las dependencias según el usuario logueado.

7. Pruebas y puesta en marcha

En el este capítulo se presentan las pruebas realizadas en el SIRCV. Las mismas fueron realizadas inicialmente en el entorno de desarrollo y cuando estas se superaron exitosamente se procedió a realizar la puesta en marcha en producción y repetir las mismas pruebas para garantizar que el sistema funcione correctamente. Como se comentó en capítulos anteriores el sitio de desarrollo y de producción no poseen las mismas tecnologías, aunque el sistema garantiza que sea independiente de la plataforma, es sumamente necesario que las pruebas se hagan en ambos entornos.

Por lo tanto este capítulo presenta las pruebas realizadas, la puesta en marcha del sitio en producción y algunas funcionalidades que posee el sistema para obtener información una vez que el sistema está terminado y con datos reales. Además se detalla la implementación de los servicios añadidos al cron del servidor de producción.

7.1. Pruebas de software

Las pruebas de software son el conjunto de actividades destinadas a validar y verificar de manera objetiva que se ha generado un software de calidad, libre de errores y que cumple con lo exigido.

No existe una clasificación oficial de los tipos de pruebas a realizar pero se pueden establecer varias clasificaciones según diversos criterios (39), a continuación se detallan las realizadas para el SIRCV:

- Según el enfoque utilizado
 - Pruebas de tipo caja negra
 - Pruebas de tipo caja blanca
- Según el ámbito de las pruebas
 - Pruebas unitarias
 - Pruebas de integración
 - Pruebas de sistema

7.1.1. Pruebas de tipo caja negra

Estas pruebas verifican un software utilizando su interfaz externa. El sistema es visto como una caja negra de las que no se conoce su funcionamiento interno ni la estructura del sistema. Únicamente alcanza con saber las entradas y las salidas esperadas.

La interfaz tiene que estar muy bien definida para llevar a cabo estas pruebas.

Para realizar estas pruebas se utilizaron dos métodos que explica Paz (39): partición de equivalencia y análisis de valores frontera.

- **Partición de equivalencia:** Se basa en dividir el dominio posible de datos de entrada de un programa en clases de equivalencia. Una clase de equivalencia representa un conjunto de estados (válidos o inválidos) según una determinada entrada. Una condición de entrada es un valor específico numérico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Por ejemplo si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen tres clases de equivalencia: por debajo, en y por encima del rango.

Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango. Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él. Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Este método se utilizó para probar casos de uso del sistema que tenían múltiples clases de equivalencia, muchos pares de valores posibles válidos e inválidos por ejemplo: asignación del valor del viático por día por dependencia, control de transportes en la solicitud y rendición de viáticos, control de fechas de comisiones en la solicitud y rendición de viáticos, etc.

Para aplicar esta técnica se realizaron los siguientes pasos:

1. Identificar las clases de equivalencia y asignarle un identificador único a cada una.
2. Definir los casos de prueba intentando cubrir todas las clases de equivalencia.

Además se utilizó el siguiente formato:

Caso de uso: <Nombre>

Caso de prueba: <Nombre>

Entrada: <Descripción textual de lo que ocurre en el mundo real que hace necesario ejecutar el caso de prueba, precisando el dato de entrada y los comandos a dar por el actor. Descripción textual del estado de la información almacenada>

Resultado: <Descripción textual del estado en el que queda la información y las alertas que puedan generarse, una vez ejecutado el caso de uso con los valores y el estado especificado en la entrada>

A continuación, se muestra la prueba correspondiente al control realizado para comprobar que un afiliado no esté en otra comisión en el mismo momento que se quiere establecer una nueva mediante la solicitud de viáticos.

Es decir, la prueba de equivalencia se relaciona al caso de uso “Realizar tareas de control y preventivas para evitar fraudes en solicitudes y rendiciones de viáticos con respecto al dinero solicitado, los vehículos y los afiliados involucrados”. El sistema responde a través de su interfaz mediante un popup la respuesta indicada, además no permite que se agregue a la comisión que se va formando en un cuadro inferior de la pantalla.

Caso de uso: Realizar tareas de control y preventivas para evitar fraudes en solicitudes y rendiciones de viáticos con respecto al dinero solicitado, los vehículos y los afiliados involucrados.

Caso de prueba: Se prueba que un afiliado no pertenezca a otra comisión formada en solicitudes o rendiciones anteriores en el mismo rango de fecha.

Entrada: A priori se sabe que el afiliado seleccionado para probar pertenece a otra comisión. Se prueban cuatro rangos de fechas: que se superponga, que se superponga el inicio, el final y ninguna de ellas.

Resultado: Es un booleano, verdadero si se superponen las fechas y falso en caso contrario. Esto debe aparecer en el popup y no se debe agregar el afiliado en el cuadro inferior.

Condiciones: Se debe seleccionar el afiliado y debe estar cargada la fecha de inicio de la comisión y la duración en días. Si el caso corresponde a un afiliado de vialidad se deben

- **Análisis de valores frontera:** Otro de los métodos utilizados para probar el sistema dentro de la clasificación “caja negra” es el análisis de valores frontera, se complementa al de clases de equivalencia pero solamente utilizando valores extremos. Es decir se reutilizan las planillas utilizadas en el método anterior pero en las entradas se cargan solamente los valores extremos. Por ejemplo en el caso de uso del control de comisiones se carga la fecha exacta en la que se superpone el afiliado en otra comisión.

7.1.2. Pruebas de tipo caja blanca

A diferencia de las pruebas de caja negra estas prueban la estructura interna del sistema, para ello se necesita conocer el código fuente del mismo. La persona encargada de efectuar estas pruebas

elige diversos valores de entrada, examinando el flujo del programa y comprobando que se devuelven los valores correctos. Los casos de pruebas derivados, siempre que se ejecuten de manera exhausta, cumplirán lo siguiente:

- Garantizar que todas las rutas se revisan.
- Revisar todas las decisiones lógicas.
- Ejecutar todos los bucles con sus valores fronteras.
- Revisar las estructuras de datos internas.

Los métodos utilizados para probar este enfoque son: “prueba de condición” y “prueba de flujos de datos”.

- **Prueba de condición:** La prueba de condición (40) es un método de diseño de casos de prueba que revisa las condiciones lógicas contenidas en un módulo de programa. Una condición simple es una variable booleana o una expresión relacional, posiblemente precedida de un operador NOT (¬). Una expresión relacional toma la forma E1 <operador relacional> E2 donde E1 y E2 son expresiones aritméticas y <operador relacional> es uno de los siguientes: <, ≤, =, >, ≥. Una condición compuesta se integra con dos o más condiciones simples, operadores booleanos y paréntesis. Se supone que los operadores booleanos permitidos en una condición compuesta incluyen OR (|), AND (&) y NOT (¬). Una condición sin expresiones relacionales se conoce como expresión booleana.

Si una condición es incorrecta, entonces al menos un componente de la condición es incorrecto. Por tanto, los tipos de errores en una condición incluyen errores de operador booleano (operadores booleanos incorrectos/perdidos/adicionales), de variable booleana, de paréntesis booleanos, de operador relacional y de expresión aritmética. El método de prueba de condición se enfoca en la prueba de cada condición del programa para asegurar que no contiene errores.

A continuación se muestra la plantilla utilizada algunas de las pruebas realizadas para el SIRCV.

Escenario – Condición	Salida Esperada	Casos
1. Cantidad de afiliados en la comisión de solicitud = cantidad de afiliados en la comisión de rendición	Booleano : true	Módulo de control entre solicitud y rendición de una misma comisión
2. Cantidad de dinero solicitado = cantidad de dinero rendido	Booleano : true	Módulo de control entre solicitud y rendición de una misma comisión

- **Prueba de flujo de datos:** El método de prueba de flujo de datos (41) selecciona rutas de prueba de un programa de acuerdo con las ubicaciones de las definiciones y con el uso de variables en el programa. Para ilustrar el enfoque de prueba de flujo de datos, se supone

que a cada enunciado en un programa se le asigna un número de enunciado único y que cada función no modifica sus parámetros o variables globales. Para un enunciado con S como su número de enunciado,

$DEF(S) = \{X \mid \text{enunciado } S \text{ contiene una definición de } X\}$

$USE(S) = \{X \mid \text{enunciado } S \text{ contiene un uso de } X\}$

Si el enunciado S es un enunciado if o loop, su conjunto DEF es vacío y su conjunto USE se basa en la condición del enunciado S.

A continuación, se muestra la plantilla utilizada para estas pruebas con algunos ejemplos de ellas.

Nombre de escenario	Flujo Inicial	Flujo alternativo	Flujo alternativo	Salida
Obtener el dinero por día para un afiliado	Viático fuera de la provincia	Recibe comida	Recibe alojamiento	X dinero
Obtener el dinero por día para un afiliado	Viático fuera de la provincia	No recibe comida	No recibe alojamiento	X dinero
Obtener el dinero por día para un afiliado	Viático dentro de la provincia	Salida antes de las 12hs	Regreso después de las 18hs	X dinero
Obtener el dinero por día para un afiliado	Viático fuera de la provincia	Salida antes de las 12hs	Regreso entre 13,30hs y 18hs	X dinero

7.1.3. Pruebas unitarias

Estas pruebas verifican el funcionamiento de una pieza de software. El concepto de pieza abarca módulos individuales, componentes, subprogramas, etc. El objetivo es probar la funcionalidad de una parte del sistema. Al usar la descripción del diseño de componente como guía, las rutas de control importantes se prueban para descubrir errores dentro de la frontera del módulo. La relativa complejidad de las pruebas y los errores que descubren están limitados por el ámbito restringido que se establece para la prueba de unidad. Las pruebas de unidad se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente. Este tipo de pruebas puede realizarse en paralelo para múltiples componentes.

Además la interfaz del módulo se prueba para garantizar que la información fluya de manera adecuada hacia y desde la unidad de software que se está probando. Las estructuras de datos locales se examinan para asegurar que los datos almacenados temporalmente mantienen su integridad durante todos los pasos en la ejecución de un algoritmo. Todas las rutas independientes a través de la estructura de control se ejercitan para asegurar que todos los estatutos en un módulo se ejecuten al menos una vez. Las condiciones de frontera se prueban para asegurar que el módulo opera adecuadamente en las fronteras establecidas para limitar o restringir el procesamiento. Y, finalmente, se ponen a prueba todas las rutas para el manejo de errores.

Esta prueba se realizó por cada módulo del sistema utilizando la planilla que se muestra a continuación.

Entrada	Salida esperada
[Condiciones de entrada para realizar la prueba.]	[Salida esperada para la condición de entrada especificada en esta línea.]

7.1.4. Pruebas de integración

Una vez que todos los módulos se hayan probado de manera individual y todos ellos funcionan correctamente se debe probar su integración con el resto del sistema. Muchas veces un módulo funciona correctamente de forma individual pero en la interfaz con otro módulo se pierden datos por ejemplo. El objetivo es tomar los componentes probados de manera individual y probar un caso de uso que involucre a más de un componente. De esta manera se probará su interacción.

Esta prueba fue muy necesaria ya que el SIRCV contiene muchas relaciones (expresadas en el diagrama de entidad y relación) entre los módulos.

Existen muchas técnicas para realizar las pruebas de integración de los componentes. La utilizada para el sistema de viáticos es el de regresión.

Cada vez que se agrega un nuevo módulo como parte de las pruebas de integración, el software cambia. Se establecen nuevas rutas de flujo de datos, ocurren nuevas operaciones de entrada/salida y se invoca nueva lógica de control. Dichos cambios pueden causar problemas con las funciones que anteriormente trabajaban sin fallas. En el contexto de una estrategia de prueba de integración, la prueba de regresión es la nueva ejecución de algún subconjunto de pruebas que ya se realizaron a fin de asegurar que los cambios no propagaron efectos colaterales no deseados.

En un contexto más amplio, las pruebas exitosas (de cualquier tipo) dan como resultado el descubrimiento de errores, y los errores deben corregirse. Siempre que se corrige el software, cambia algún aspecto de la configuración del software (el programa, su documentación o los datos que sustenta). Las pruebas de regresión ayudan a garantizar que los cambios (debidos a pruebas o por otras razones) no introducen comportamiento no planeado o errores adicionales.

Las pruebas de regresión se pueden realizar manualmente, al volver a ejecutar un subconjunto de todos los casos de prueba o usando herramientas de captura/reproducción automatizadas. Estas pruebas fueron realizadas cada vez que se creó un nuevo módulo, por ejemplo cuando se creó el módulo solicitud se probó individualmente y su integración con dependencia y afiliado.

7.1.5. Pruebas de sistema

La prueba del sistema es una serie de diferentes pruebas cuyo propósito principal es probar por completo el sistema. Aunque cada prueba tenga un propósito diferente, se pretende verificar que

los elementos del sistema se hayan integrado de manera adecuada y que se realicen las funciones asignadas. Los distintos tipos de pruebas de sistema que se realizaron fueron: recuperación, seguridad, esfuerzo, rendimiento y despliegue.

Prueba de seguridad: Intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier acceso externo impropio. Estas pruebas se realizaron solamente en el entorno de producción probando entrar con usuarios inexistentes, contraseñas erróneas, acceder a links sin loguearse al sistema, etc.

Prueba de esfuerzo: Las pruebas de esfuerzo se diseñan para enfrentar los programas con situaciones anormales.

La prueba de esfuerzo ejecuta un sistema en forma que demanda recursos en cantidad, frecuencia o volumen anormales. Por ejemplo, pueden 1) diseñarse pruebas especiales que generen diez interrupciones por segundo, cuando una o dos es la tasa promedio; 2) aumentarse las tasas de entrada de datos en un orden de magnitud para determinar cómo responderán las funciones de entrada; 3) ejecutarse casos de prueba que requieran memoria máxima y otros recursos; 4) diseñarse casos de prueba que puedan causar thrashing (que es un quebranto del sistema por hiperpaginación) en un sistema operativo virtual y 5) crearse casos de prueba que puedan causar búsqueda excesiva por datos residentes en disco.

Esta prueba se realizó también en el entorno de producción (el servidor alojado en el Tribunal de Cuentas) donde se probó realizar varias acciones en simultáneo y aquellas que disparan eventos en el cron para comprobar el correcto funcionamiento.

Prueba de rendimiento: La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de ejecución, dentro del contexto de un sistema integrado. La prueba de rendimiento ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, se puede acceder al rendimiento de un módulo individual conforme se realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema.

Las pruebas de rendimiento con frecuencia se asemejan con las pruebas de esfuerzo y por lo general requieren instrumentación de hardware y de software, es decir, con frecuencia es necesario medir la utilización de los recursos (por ejemplo, ciclos del procesador) en forma meticulosa.

La instrumentación externa puede monitorear intervalos de ejecución y eventos de registro (por ejemplo, interrupciones) conforme ocurren, y los muestreos del estado de la máquina de manera regular. Con la instrumentación de un sistema, la persona que realiza la prueba puede descubrir situaciones que conduzcan a la degradación y posibles fallas del sistema.

Esta prueba se realizó solamente en el entorno de producción midiendo los recursos de hardware cuando se realizaron las pruebas de esfuerzo.

Prueba de despliegue: La prueba de despliegue, en ocasiones llamada prueba de configuración, prueba el software en cada entorno en el que debe operar. Además, examina todos los procedimientos de instalación y toda la documentación que se usará para introducir el software a los usuarios finales.

Esta prueba se realiza en el entorno de desarrollo (Windows) y en el entorno de producción (Linux), probando cada funcionalidad en distintos navegadores: Chrome, Mozilla e Internet Explorer.

7.2. Puesta en marcha del sistema

La puesta en marcha técnica consistió en plasmar en el servidor de producción todo lo realizado en el sitio de desarrollo.

El servidor, como se plasmó anteriormente, es un sistema operativo Linux distribución Debian que tiene instalado: Apache, MYSQL, Phpmyadmin y PHP5 (requerimientos esenciales para la ejecución del sistema integral de registro y control de viáticos) que se encuentra en el Tribunal de Cuentas de la provincia de La Pampa en la ciudad de Santa Rosa.

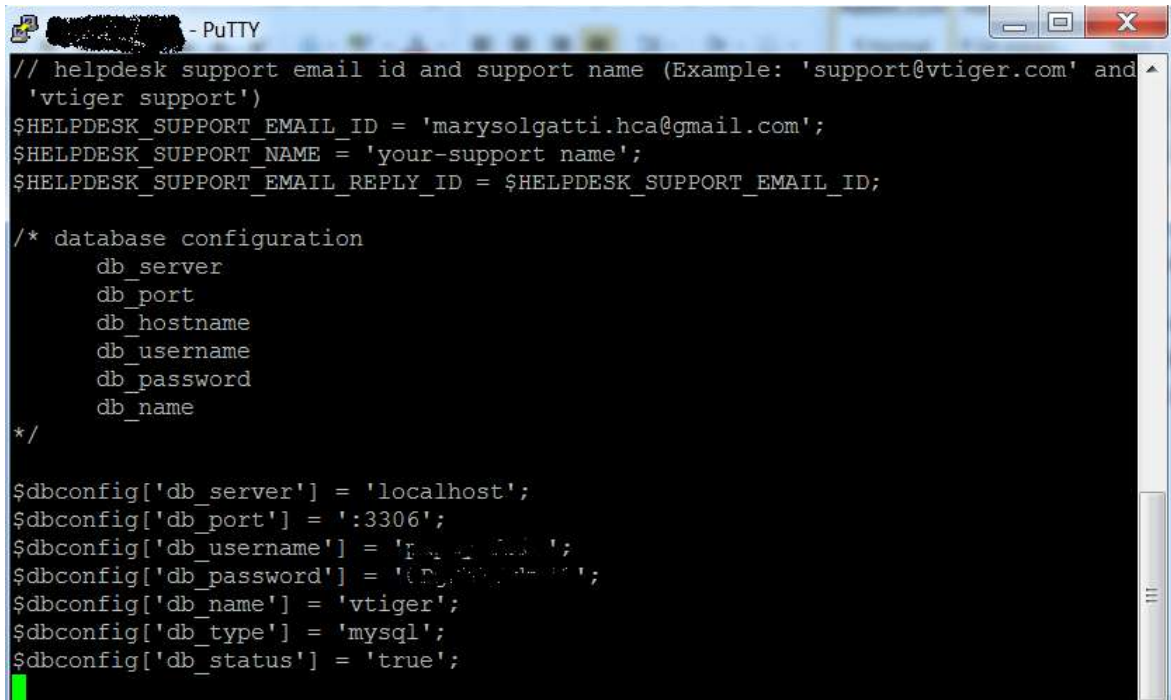
Por esto para modificar algunos archivos se utilizó un cliente SSH que permite el acceso remoto: Putty.

El primer paso fue instalar Git en el servidor, luego mediante comandos Git se clonó el repositorio alojado en BitBucket (actualizado mediante el desarrollo del sistema) en la carpeta root del servidor.

Dentro del proyecto hay una carpeta que se llama `baseDeDatos`, allí se encuentra la base de datos inicial del sistema que se realizó mediante un comando MYSQL (`mysql_dump (42)`) del sitio de desarrollo eliminando los registros de prueba. Esta base de datos contiene la estructura y los datos iniciales que el sistema requiere: afiliados, dependencias, usuarios, etc; y se importa en la base de datos de producción. Para esta acción se utilizan comandos MYSQL y se ejecutan a través del cliente Putty.

Luego, lo único que falta es editar el archivo de configuración del sistema para que apunte a la nueva bases de datos, para ello con Putty accedemos al archivo, mediante el editor de Linux Vim (43), `config.inc.php` y modificamos las variables relacionadas a la base de datos. En la Figura 52 se muestra dicha acción.

Luego de clonar el proyecto en el servidor, importar la base de datos y relacionarla con el sitio de producción el sistema integral de registro y control de viáticos se encuentra funcionando en el sitio de producción y es accesible desde cualquier computadora con internet o desde cualquier computadora conectada a la red local del mismo.



```
// helpdesk support email id and support name (Example: 'support@vtiger.com' and
'vtiger support')
$HELPDESK_SUPPORT_EMAIL_ID = 'marysolgatti.hca@gmail.com';
$HELPDESK_SUPPORT_NAME = 'your-support name';
$HELPDESK_SUPPORT_EMAIL_REPLY_ID = $HELPDESK_SUPPORT_EMAIL_ID;

/* database configuration
    db_server
    db_port
    db_hostname
    db_username
    db_password
    db_name
*/

$dbconfig['db_server'] = 'localhost';
$dbconfig['db_port'] = ':3306';
$dbconfig['db_username'] = 'root';
$dbconfig['db_password'] = '(D)33333333';
$dbconfig['db_name'] = 'vtiger';
$dbconfig['db_type'] = 'mysql';
$dbconfig['db_status'] = 'true';
```

Figura 52: Edición del archivo de configuración del sitio de producción

Se repiten las pruebas realizadas en el sitio de desarrollo y se analiza el rendimiento del mismo. Las pruebas fueron muy positivas ya que más allá del sistema el servidor contiene tecnología de alto rendimiento.

7.3. Cron, tareas automatizadas.

Como se especificó en el capítulo anterior una de las herramientas utilizadas fueron los flujos de trabajo que provee Vtiger y la vinculación con el cron del servidor.

Para ello una vez que se crearon los flujos de trabajo el paso siguiente es indicarle al cron del servidor que es lo que tiene que ejecutar y cada cuanto tiempo.

Cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab (44).

El demonio cron inicia de /etc/rc.d/ o /etc/init.d dependiendo de la distribución. Cron se ejecuta en background, revisa cada minuto la tabla de tareas crontab/etc/crontab o en /var/spool/cron en búsqueda de tareas que se deban cumplir.

Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el script o el comando, los permisos de ejecución y lo realizará en el background.

Para agregar los flujos de trabajo de Vtiger en el Crontab se realizaron los pasos que se explican a continuación. Se ejecuta la edición del crontab con “`crontab -e`” desde consola utilizando el cliente Putty para acceso remoto. El archivo crontab lucirá algo así: “`#m h dom mon dow user command`”. Donde tal como el blog de Linux (45) cada parámetro significa:

- m corresponde al minuto en que se va a ejecutar el script, el valor va de 0 a 59
- h la hora exacta, se maneja el formato de 24 horas, los valores van de 0 a 23, siendo 0 las 12:00 de la medianoche.
- dom hace referencia al día del mes, por ejemplo se puede especificar 15 si se quiere ejecutar cada día 15
- dow significa el día de la semana, puede ser numérico (0 a 7, donde 0 y 7 son domingo) o las 3 primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun.
- user define el usuario que va a ejecutar el comando, puede ser root, u otro usuario diferente siempre y cuando tenga permisos de ejecución del script.
- command refiere al comando o a la ruta absoluta del script a ejecutar, ejemplo: `/home/usuario/scripts/actualizar.sh`, si acaso llama a un script este debe ser ejecutable

Por lo tanto para agregar al cron las tareas automatizadas de él SIRCV se editó el **Crontab** como se muestra a continuación:

```
15 * * * * administrador /var/www/html/tribunal/cron/ vtigercron.sh.
```

De esta manera se chequea cada quince minutos todos los días si hay tareas pendientes a ejecutar.

7.4. Herramientas para la toma de decisiones.

Además de las funcionalidades que provee el sistema diseñadas y programadas, se añaden funciones de Vtiger que permiten tomar decisiones o mostrar en un momento exacto información necesaria. Estas herramientas son: Informes y Filtros.

Informes: Los informes permiten obtener la información que se requiera en formato PDF, Word y Excel. Se puede crear uno nuevo o establecer uno y guardarlo para su posterior impresión. Se debe seleccionar un módulo principal, se puede optar por uno o más módulos relacionados al módulo principal y también se eligen los campos de cualquier de estos módulos. Además se puede

añadir una o varias condiciones o cálculos. Por ejemplo, se listan algunos informes que se probaron y se dejaron para que el usuario los utilice:

- 1- Todas las solicitudes de la dependencia Salud creadas en los últimos 15 días, mostrando: número de solicitud, motivo, fecha y monto.
- 2- Todas las solicitudes que no tienen rendición creadas en el último mes, mostrando: número de solicitud, motivo, dependencia, fecha y monto.
- 3- Todas las rendiciones que tienen un monto superior a mil pesos, mostrando: número de rendición, motivo, dependencia, fecha y monto.
- 4- Todas las rendiciones que tienen diferencias con sus respectivas solicitudes y aún no han sido controladas.

En la Figura 53 se muestra el informe correspondiente al punto 1.

Comisión de Servicio_N	Comisión de Servicio_Motivo de la Comisión	Comisión de Servicio_Fecha	Comisión de Servicio_Total
678320	Reunión de ministerios	04-05-2015	2520.00
713300	Auditoría Profe	04-05-2015	3210.00

Figura 53: Informes de la dependencia Salud

Filtros: Los filtros permiten visualizar información de un módulo particular desde su vista “lista”. Desde esta vista el sistema permite crear nuevos filtros y establecerlos para su futura utilización. Por ejemplo desde el módulo solicitud un filtro que permita listar los registros (para su posterior acceso) que no tengan aún una rendición asociada, o todas las solicitudes de la dependencia Vialidad, etc. En la Figura 54 se muestra la creación de un filtro realizado desde el módulo solicitud.

Creando nueva Vista

Detalles Básicos:

Nombre Vista*: Solicitudes sin rendición

☐ Fijar por defecto ☐ Mostrar en métricas ☒ Hacer Público

Elegir columnas y orden (Máx 12):

☒ Nº ☒ Fecha ☒ Dependencia ☒ Motivo de la Comisión

Selecciona las condiciones del filtro:

Todas las Condiciones (Se deben cumplir todas las condiciones)

Con Rendición esta deshabilitado

Añadir Condición

Figura 54: Configuración de un filtro para el módulo solicitud

8. Conclusiones

Luego de que se realizaron todas las funcionalidades en el sitio de desarrollo se realizó una reunión con el cliente para mostrar las tareas realizadas y demostrar la facilidad que brinda la interfaz para poder operar y realizar las tareas requeridas.

Para esta etapa final se realizó un encuentro formal en la Facultad de Ingeniería de General Pico, donde participaron público en general, docentes de la facultad, parte del personal y el director del Tribunal de Cuentas de la provincia de La Pampa. Se expuso el resultado final del proyecto: mostrando partes del sistema, explicando cada una de las funcionalidades demostrando en todo momento que se habían cumplido los requerimientos funcionales y no funcionales mediante la utilización de un sistema amigable e intuitivo.

Luego de la exposición se realizó una conferencia de prensa, donde participaron distintos medios locales y provinciales, y se dio por finalizada la entrega formal del proyecto al cliente.

El proyecto cubrió todas las etapas del ciclo de vida del software, dada la participación que debía tener el cliente, la complejidad y los riesgos a asumir se seleccionó basarse un ciclo de vida en espiral junto con la mejora aportada por Boehm: Victoria – Victoria . Esta mejora permitió una constante participación del Tribunal a lo largo de todo el proyecto y una gran satisfacción de ambas partes con el producto logrado.

Desde el análisis se detectó que el sistema sería de gran envergadura y que se debían satisfacer varios requerimientos no funcionales. Además se debía considerar que el sistema es para el estado provincial y contiene datos muy sensibles. Es por esto, que fue muy conveniente reutilizar herramientas ya disponibles de código abierto que cubrieran algunos de los requerimientos funcionales y no funcionales presentados por el cliente. A su vez, al utilizar tecnologías de código abierto gracias a la documentación que estas presentan es más fácil y ventajosa el traspaso de conocimiento al equipo de desarrolladores del Tribunal de Cuentas que continuarán con el proyecto.

Por lo tanto, resultó beneficioso haber realizado una adaptación de Vtiger en vez de realizar un sistema desde cero ya que permitió cumplir uno de los objetivos planteados; desarrollar un sistema orientado a la web, seguro y estable. Además, otro de los objetivos logrados es la obtención de información para la toma de decisiones, lo cual fue logrado con los Informes y Filtros que posee Vtiger.

La opción de customizar Vtiger resultó exitosa, el cliente quedó muy satisfecho con la interfaz y de las facilidades que esta brinda. Esta condición permite que la capacitación para los usuarios finales sea más fácil también, que el sistema cuente con un entorno amigable. Hay que considerar que los usuarios no están utilizando ningún sistema y solo hacen este proceso en soporte papel, por lo que la migración será un gran proceso de cambio.

Además Vtiger no es una caja negra, al ser de código abierto se puede añadir código en cualquier parte del CRM o realizar modificaciones para que se invoque código fuente externo al modelo del mismo. Por ejemplo para ciertas funcionalidades como la carga de destinos a través de Google Maps, se invocó mediante un popup código externo al mismo. Esto demuestra que el código es flexible, reutilizable y cumple con los requerimientos no funcionales expuestos por el Tribunal.

Como conclusión de este informe se puede establecer que se cumplieron con los objetivos planteados, se pudieron realizar los requerimientos solicitados por el cliente y actualmente el sistema está listo para ser utilizado.

Tal como se planteó en los objetivos el sistema se asemeja al proceso de control de viáticos llevado a cabo por el Tribunal de Cuentas, contiene datos pre-cargados que evitan el error humano (por ejemplo nombre de afiliados, DNI, etc.), automatiza tareas de control para evitar el fraude, permite la impresión de planillas para poder seguir teniendo copias en papel y brinda muchísima información para la toma de decisiones.

Además, posee el soporte para que se pueda acceder no solamente desde la red local sino desde todos los puntos de la provincia.

Por último, la realización de este proyecto ha sido una gran experiencia profesional y personal, de a momentos compleja pero muy satisfactoria.

Anexo I: Especificación de requerimientos

Sistema Integral de Registros y Control de Viáticos para el Tribunal de Cuentas de la Provincia de La Pampa.

Especificación de Requerimientos de Software

Versión [1.0]

Historia de revisiones

Fecha	Versión	Descripción	Autor
[26/05/2015]	[1.0]	Primer versión del proyecto	Marysol Gatti

1. Introducción

1.1. Propósito

El presente documento tiene como propósito definir las especificaciones funcionales, no funcionales y del sistema para la implementación de un SIRCV para el Tribunal de Cuentas de la provincia de La Pampa. El Sistema en un principio será utilizado por personal del Tribunal de Cuentas y en un futuro no tan lejano por el personal de toda la provincia.

En el documento se encuentran las especificaciones del proyecto en general, las descripciones específicas de cada requerimiento fueron realizadas y presentadas en planillas con el formato descripto en el punto 1.4 y se complementan con los diagramas de casos de uso.

1.2. Alcance

- Análisis, diseño, desarrollo e implantación del sistema.
- El SIRCV será un sistema web que permita ingresar las solicitudes de viáticos de cada dependencia de la provincia, así como también las rendiciones correspondientes. Además deberá realizar controles para evitar casos de fraude en cada solicitud y rendición de viáticos con respecto a: comisiones (grupo de empleados que realiza un viático), vehículos, dinero solicitado/rendido, etc.
- El Sistema deberá contemplar que no todas las dependencias registran los viáticos de la misma manera ni todos los empleados reciben el mismo valor en pesos por día de viático realizado.
- El sistema deberá brindar información útil para la toma de decisiones. Por lo tanto, este sistema dará soporte a los siguientes procesos:
 - Ingreso de solicitudes de viáticos respetando el formato con el que actualmente trabaja el Tribunal en soporte papel.
 - Ingreso de rendiciones de viáticos respetando el formato con el que actualmente trabaja el Tribunal en soporte papel. Carga del itinerario y de la carga de gastos correspondientes a la comisión realizada.
 - Administración de usuarios, roles y perfiles según la jerarquía de los empleados de la provincia.
 - Cálculo del valor en pesos que le corresponde a cada afiliado (empleado) por día de viático según su jerarquía y las consideraciones específicas de la dependencia para la que viatica.
 - Control de vehículos y afiliados en la solicitud de viáticos.
 - Control de vehículos y afiliados en la rendición de viáticos.
 - Liquidación de viáticos
 - Controles entre solicitud y rendición de una misma comisión, información estadística.
- Será un entorno web amigable, donde cada usuario tiene un perfil definido. Donde los permisos se definirán por perfil.
- El sistema contendrá herramientas intuitivas para poder obtener información importante y oportuna.

1.3. Definiciones, siglas y abreviaturas.

1.3.1. Del negocio

Afiliado: Cada empleado de la provincia, se identifica por un número.

Jurisdicción: Entidad por la que se divide la entidad gubernamental, esta clasificación es por funcionalidad. Cada jurisdicción tiene x cantidad de dependencias. Por ejemplo una jurisdicción es “Ministerio de Cultura y Educación”.

Dependencia: Entidad por la que se divide la jurisdicción, estas clasificaciones es por sub-funcionalidad. Por ejemplo una dependencia de la jurisdicción antes nombrada es “Santa Rosa – Escuela N°2”.

Comisión: Conjunto de afiliados que realizan un mismo viático.

Itinerario: Recorrido realizado en un viático.

1.4. Descripción

- 8.1. En el presente documento se encontrará la información acerca de las características del producto de software, interfaces del usuario, interfaces del sistema, características de los usuarios, descripción de los requerimientos funcionales, no funcionales y del sistema, los cuales se representaran mediante el siguiente formato:

Sistema Integral de Registros y Control de Viáticos					
Código	Nombre		Fecha	Grado de Necesidad	
Descripción					
Entradas	Fuente	Salida	Destino	Restricciones	
Proceso					
Efecto					

8.2.

Código:

RF : Requerimiento Funcional

RFN : Requerimiento No Funcional

RI: Requerimiento de Interfaz

9.

2. Descripción general

2.1. Perspectiva del producto

El SIRCV permitirá realizar las siguientes funciones:

Administración de Usuarios y permisos: El administrador del sistema podrá gestionar los usuarios: agregar, modificar, eliminar, buscar, listar y definir permisos.

Administración de Roles y Perfiles: El administrador del sistema gestionará los roles y perfiles de los usuarios del sistema, de igual forma asignará los roles a los usuarios.

Navegación: Proceso por el cual una persona podrá utilizar el Sistema Integral de Registro y Control de Viáticos.

Administración de Vehículos: Alta, baja y modificación de los vehículos que serán utilizados en los módulos: Solicitud y Rendición de viáticos.

Administración de Dependencia: Alta, baja y modificación de las dependencias que serán partícipes y responsables de cada viático solicitado o rendido.

Administración Solicitudes y Rendiciones de viáticos: Alta, baja y modificación de solicitudes y rendiciones. Cada una de ellas tiene relacionado una comisión (grupo de afiliados) y uno o dos vehículos en una fecha determinada.

Obtención de información útil sobre los viáticos realizados.

Control entre Solicitudes y Rendiciones de un mismo viático.

9.1.1.

2.2. Interfaces de usuario

El Sistema tendrá distintos tipos de usuario: Administrador, Jefe de Relatores, Relator Mayor, Relator, Habilitante y Sub-Habilitante. El administrador será el usuario encargado que predisponga el Tribunal de Cuentas, el resto de los perfiles son los que actualmente posee esta entidad y tendrán los permisos en el sistema acordes a su funcionalidad.

Los empleados de otras entidades que en un futuro cercano ingresarán para dar de alta Solicitudes y Rendiciones (solo tendrán permiso para realizar esto) son: habilitantes y sub-habilitante.

Por otra parte los usuarios Administradores tendrán acceso a todo el sistema, como los Jefes de Relatores (excepto que no tienen permiso de ABM de usuario, permisos y campos). Los Relatores Mayores solo verán las solicitudes y rendiciones realizadas por sus relatores a cargo (y por ende los habilitantes y sub-habilitantes que se relacionan con esos relatores) y tendrán acceso a las funcionalidades de Control y Toma de decisión.

2.3. Interfaces con hardware

Para esta aplicación será necesario un computador servidor en el cual se instalará el servidor WEB Apache, MySQL, PHP5, Vtiger 6 y las funcionalidades añadidas.

2.4. Interfaces con software

Las conexiones necesarias para la utilización del servidor web, MySQL, PHP y un DNS, se hará por medio de la configuración de estos programas. Además el sistema se comunicará con funcionalidades de la librería Google Maps para dar soporte a las coordenadas geográficas que son los destinos de viáticos de vialidad.

2.5. Interfaces de comunicación

9.1.2. Se usará protocolos de comunicación TCP/IP, HTTP.

2.6. Funciones de control

El sistema debe controlar los permisos que tiene cada usuario para su accesibilidad de una manera correcta, de tal forma que pueda acceder a la información que le corresponde de acuerdo a su rol. Debe tener controles adecuados para la validación de datos, de igual manera la programación de las actividades específicas para cada funcionalidad.

2.7. Credibilidad del sistema

Para garantizar una buena credibilidad el sistema deberá ser sometido a una serie de pruebas para establecer que se encuentra acorde a los requerimientos que se plasman en este documento en tanto a la consistencia de datos como al rendimiento de la aplicación, tales como tiempos de respuesta.

2.8. Consideraciones de seguridad

Cada usuario deberá autenticarse y su acceso verificado por una sola Terminal para su respectiva labor de acuerdo a lo que su rol especifique. Todas las claves de seguridad deberán estar seguras y en su defecto encriptadas en la base de datos para dar una buena seguridad al sistema y su información.

Anexo II: Plantilla Casos de Uso

RF- <id del requisito>	<nombre del requisito funcional>	
Versión	<numero de versión y fecha>	
Autores	<autor>	
Fuentes	<fuente de la versión actual>	
Objetivos asociados	<nombre del objetivo>	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso { concreto cuando <evento de activación> , abstracto durante la realización de los casos de uso <lista de casos de uso> }	
Precondición	<precondición del caso de uso>	
Secuencia Normal	Paso	Acción
	1	{El <actor> , El sistema} <acción realizada por el actor o sistema>, se realiza el caso de uso < caso de uso RF-x>
	2	Si <condición>, {el <actor> , el sistema} <acción realizada por el actor o sistema>>, se realiza el caso de uso < caso de uso RF-x>
	3	
	4	
	5	
	6	
	n	
Postcondición	<postcondición del caso de uso>	
Excepciones	Paso	Acción
	1	Si <condición de excepción>,{el <actor> , el sistema} }<acción realizada por el actor o sistema>>, se realiza el caso de uso < caso de uso RF-x>, a continuación este caso de uso {continua, aborta}
	2	
	3	
Rendimiento	Paso	Cota de tiempo
	1	n segundos
	2	n segundos
Frecuencia esperada	<nº de veces> veces / <unidad de tiempo>	
Importancia	{sin importancia, importante, vital}	
Urgencia	{puede esperar, hay presión, inmediatamente}	
Comentarios	<comentarios adicionales>	

Anexo III: Planilla de Solicitud de Viáticos

REPUBLICA ARGENTINA
GOBIERNO DE LA PROVINCIA DE LA PAMPA

SOLICITUD AUTORIZACIÓN - COMISIÓN DE SERVICIOS N° 697277 / AÑO.....E
167

Lugar y Fecha QUATRACHO, Mayo 06

Dependencia Zona SUB

Motivo de la Comisión Reparto Combustible -- Trasl.Camioneta /

Fecha de Iniciación 06/05 Duración Prevista 5 /

Lugar de residencia durante la comisión

Medio de transporte a utilizar..... [Redacted]

Se indicará: 1) Si es vehículo oficial; su n° de legajo; 2) Si es vehículo arrendado; el n° de expediente;
3) Si es vehículo particular; número de póliza, vigencia y adjuntar fotocopia.-


Imputación Presupuestaria: Jurisd. U.de Organ. Carácter Cuenta

Fin. y Func. Sección P.P. P.p. S.p. CL. SC. C.

DETALLE DEL PERSONAL QUE REALIZARA LA COMISION DE SERVICIOS Y PLANILLA DE ANTICIPO PARA VIATICOS Y GASTOS:

Nº DE AFILIADO	APELLIDO Y NOMBRE	CAT.	VIATICOS			GASTOS		TOTAL	Recibi el importe (1)
			Dias	Espor dia	Total \$	Bs. Clase	Servicios	ANTICIPO	
[Redacted]	[Redacted]	[Redacted]				-	s/a		
[Redacted]	[Redacted]	[Redacted]				-	s/a		
TOTALES							s/a		

(En caso de no solicitarse anticipo, deberá cumplimentarse el detalle del personal que realizará la comisión y se consignará en la columna "Recibí el Importe (1)": "SIN ANTICIPO").



Ing. MARCELO MA RINCO
AJC DIRECCION
DIRECCION CONSERVACION
DE ENTRENAMIENTO Y COORDENACION
Firma y Sello Jefe Unidad de Ejecucion

Autorizado
[Signature]

Firma y Sello Funcionario

Ing. RAUL A. URRA
INGENIERO JEFE
DIRECCION PROVINCIAL DE VIALIDAD
[Signature]

Firma y Sello Funcionario

Fern uso de la Habilitación: Nº de Cheques emitidos pagos anticipos: _____

Pera uso de la Repartición que solicitó la Comisión de Servicios: Fecha y Nº de planilla de rendición de la comisión: _____

MESA DE ENTRADAS: recibí original - Fecha Hora Firma

Form. nº 80455/B

Anexo V: Planilla de Carga de Itinerario y Gastos

ITINERARIO							
APELLIDO Y NOMBRE	Dia	Mes	Salida de	Llegada a	Horario de		Importe
					Salida	Llegada	
74.221 SCHEFFER, Fabian	06	05	Guatr.- Ruta 30 - C.Gé.				
<i>Día Laboral</i>	07	05	C.Gé - Guatr.				" "
	08	05	Guatr.- R.1-24-11-Guatr				" "
<i>Descanso</i>	09	05	Guatr.- C. Gé - Guatr				411.CT
	10	05	Guatr.- Ruta 555 - Guatr				1577.F
			TOTAL \$				411.CT

NOTA: El día 06 del corriente mes y año, no pernoctó en Dependencia o casilla de la Repartición, ni en casa habitación para personal de Insp. de Obras.--



H. GARCIA ALVARADO
DIRECTOR GENERAL DE REGIONES
DE AGRICULTURA Y GANADERIA
REPUBLICA DEL PARAGUAY

FE CERTIFICA que los datos
fornecidos son veraces.
DIRECCION GENERAL DE REGISTROS

GASTOS EFECTUADOS DURANTE LA COMISION				
Fecha Factura	Firma Comercial	Concepto	Importe	Pagada por


 SCHERER, Fabian

Anexo VI: Documento Generado por el Sistema (Planilla de Rendición de Viáticos)

REPUBLICA ARGENTINA GOBIERNO DE LA PROVINCIA DE LA PAMPA

RENDICION DE COMISIÓN DE SERVICIOS

Corresponde a Comisión de Servicios N° : 678320 AÑO 2015

Lugar y Fecha : ZONA CENTRO - SANTA ROSA , 04-05-2015

Dependencia : DIRECCION DE PROGRAMAS DE SALUD

Motivo de la Comisión : REUNION

Fecha de Iniciación : 04-05-2015 Fecha de Finalización : 2015-05-07

Medio de transporte a utilizar : Legajo 2715 # N° de Poliza + Vigencia: A23112, Dominio : aaa000

Imputación Presupuestaria : Jurisd.:..... U. de Org..... Caracter..... Cuenta.....

Finc.y Func: Sección: P.P. S.p :CL. : SC :C :

LIQUIDACION Y ORDEN DE PAGO

Lugar y Fecha : SANTA ROSA 2015-05-04

Afiliado	Nombre y Apellido	Dias	\$por día	Total\$	Gastos Bs	Gastos Serv	Total	Menos Anticipo	Cobrar	Devolver
38357	MORO ROBERTO ESTEBAN	3	420	1260	0	0	1260	1260	0	0
56829	DELGADO CARLOS ALBERTO	3	420	1260	0	0	1260	1260	0	0

Total: \$2520 Gastos Bienes: \$0 Gastos Servicios: \$0 Total: \$2520 Menos Anticipo: \$2520 Cobrar: \$0 Devolver: \$0

.....
V° B° Jefe Unidad de Organización

.....
Firma Funcionario que aprueba Com.Serv.

Bibliografía

1. **Boehm, Barry, y otros, y otros.** Using the WinWin Spiral Model: A Case Study. [En línea] 1998. http://www.nyu.edu/classes/jcf/g22.3033-007_sp04/handouts/UsingTheSpiralModel.pdf.
2. **Vtiger** <https://www.vtiger.com/open-source-downloads/>. *Vtiger.com*. [En línea]
3. **Español, CRM.** <http://www.crmespanol.com/crmdefinicion.htm>. *CRM Español*. [En línea]
4. **Barry Boehm Biography.** http://sunset.usc.edu/Research_Group/barry.html. [En línea]
5. **Pressman, Roger S.** *Ingeniería del Software - Un enfoque práctico*. Mexico , D.F : McGRAW-HILL INTERAMERICANA, 2010.
6. **Falgueras, Benet Campderrich.** *Ingeniería del Software*. Barcelona : UOC, 2003.
7. **Plantilla de Documentacion en la Especificación de Requerimientos.** http://www.academia.edu/9607556/Plantilla_de_Documentaci%C3%B3n_en_la_Especificacion_de_Requerimientos. [En línea]
8. **Start UML.** <http://staruml.io/>. [En línea]
9. **Planilla de Casos de Uso.** http://www.academia.edu/7609370/Plantilla_Casos_de_Uso. [En línea]
10. **UML.** <http://www.uml.org/>. [En línea]
11. **Pressman, Roger S.** *Ingeniería del Software*. [aut. libro] R. McGlaughlin. *Some Notes on Program Design*. s.l. : Software Engineering Notes, 1991.
12. **Cabello, Antonio Luís Cardador.** *Implantación de aplicaciones web en entornos internet, intranet y extranet* . Malaga : IC, 2014.
13. **Foundation, jQuery.** <https://jquery.com/>. [En línea] .
14. **Gerencia, Revista.** <http://www.emb.cl/>. *Gerencia*. [En línea]
15. **Vtiger.** <https://www.vtiger.com/customers/>. *Cientes de Vtiger CRM*. [En línea]
16. **Empresas colaborativas de Vtiger.** <https://www.vtiger.com/partner-directory/>. [En línea]
17. **SL, Nubelo Solutions.** Nubelo. <http://www.nubelo.com/>. [En línea] .
18. **Sitio web de la comunidad española de vtigerCRM.** <https://crmevolutivo.com>. [En línea]
19. **PHP.** <https://www.php.net/>. *PHP*. [En línea]
20. **HTML W3Schools.com.** <http://www.w3schools.com/html/default.asp>. [En línea]

21. **Smarty template engine.** <http://www.smarty.net/>. [En línea]
22. **Javascript** <http://www.w3schools.com/js/>. [En línea]
23. **Phpmyadmin.** <https://www.phpmyadmin.net/>. [En línea]
24. **MYSQL .Affiliates, Oracle Corporation and/or its. MYSQL.** <https://www.mysql.com/>. [En línea] 2015.
25. **OpenSuse.** en.opensuse.org/Git . [En línea]
26. **Subversion.** <https://subversion.apache.org/>. [En línea]
27. **Git.** <https://git-scm.com/>. [En línea]
28. **Eclipse.** <https://eclipse.org>. [En línea]
29. **Netbeans.** <https://netbeans.org/>. [En línea]
30. **Google.** Google Maps APIs. <https://developers.google.com/maps/>. [En línea]
31. **Mashup** . <http://www.jackbe.com/enterprise-mashup/>. [En línea]
32. **Chrome, Google.** <https://www.google.es/chrome/browser/desktop/index.html>. [En línea]
33. **Mozilla.** <https://www.mozilla.org>. [En línea]
34. **Microsoft. Internet Explorer.** <http://windows.microsoft.com/es-xl/internet-explorer/download-ie>. [En línea]
35. **Opera.** <http://www.opera.com/es-419>. [En línea]
36. **XAMPP - Apache Friends.** <https://www.apachefriends.org/es/index.html>. [En línea]
37. **Cron - Wiki, Gentoo.** <https://wiki.gentoo.org/wiki/Cron/es>.
<https://wiki.gentoo.org/wiki/Cron/es>. [En línea]
38. **FPDF librería PHP.** <http://www.fpdf.org/>. [En línea]
39. **Paz, Rafael Luis Granados La.** *Desarrollo de aplicaciones web en el entorno servidor*. Málaga : IC , 2014.
40. **Tai, K. C.** What to Do Beyond Branch Testing. [aut. libro] Roger S. Pressman. *Ingeniería del software un enfoque practico*. 2010.
41. **Frankl, P. G. y S. Weiss.** An Experimental Comparison of the Effectiveness of Branch Testing and Data. [aut. libro] Roger S. Pressman. *Ingenieria del Software un enfoque practico*. 2010.
42. **Comando MYSQL - mysqldump.** <http://dev.mysql.com/doc/refman/5.7/en/mysqldump.html>. [En línea]

43. **Vim.** Vim. <https://es.wikipedia.org/wiki/Vim>. [En línea]
44. **Crontab.** <http://crontab.org/>. *Crontab*. [En línea]
45. **Linux, Blog.** <http://blog.desdelinux.net/cron-crontab-explicados/>. [En línea]
46. **Grady, R. B. y D. L. Caswell.** *Software Metrics: Establishing a Company-Wide Program*. s.l. : Prentice Hall, 1987.
47. **Alonso, Fernando, Martinez, Loic y Segovia, Fco Javier.** Introducción a la ingeniería de software. *Introducción a la ingeniería de software*. España : Delta, 2005.