UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

SISTEMA PARA LA GESTIÓN DE LAS TRANSPORTACIONES NACIONALES DE LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

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

AUTORES: RAMÓN FERNÁNDEZ RABILERO
ARIAN VARONA CARMENATES

TUTORES: ING. ARLENNYS S. VELÁZQUEZ HIDALGO
ING. YANIO GARCÍA VIDAL

CO-TUTOR: ING. JULIO CÉSAR ESPRONCEDA PÉREZ

LA HABANA, 7 DE JUNIO DE 2016.

Declaración de autoría

Declaramos ser los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente tesis a los ____ días del mes de _____ del año 2016.

Ramón Fernández Rabilero

_____

Firma de la autor

Arian Varona Carmenates

_____

Firma del autor

Ing. Arlennys S. Velázquez Hidalgo

_____

Firma de la tutor

Ing. Yanio García Vidal

_____

Firma del tutor

Agradecimientos

Agradezco a todas esas personas que apoyaron la realización de este trabajo y a mi formación en la Universidad. En especial a mis tutores: Arlennys que ha sido como nuestra compañera de tesis, pasando noches sin dormir junto a nosotros y dándonos su apoyo a deshoras. Nuestro cotutor Julio César que brindó su ayuda incondicional desde el primer momento. Yanio, que, aunque complicado siempre, encontró el tiempo para explicarnos todo lo que preguntamos, las veces que fuera necesario y sin perder la paciencia.

A mis familiares por la formación y apoyo, gracias a ellos me he convertido en lo que soy hoy. A mis amistades, por haberme dado la oportunidad de poder contar con ellas en estos 5 años de la carrera.

Arian Varona Carmenates

Hoy después de un largo recorrido este sueño se convierte en realidad. Ante todo, quisiera agradecer a las personas más especiales de este mundo, a mis dos madres, por todos los sacrificios que hicieron por mí, por ser guías y brindarme su confianza y cariño incondicional.

A mi padre y mi abuelo por brindarme siempre la ayuda que necesitaba.

A mi familia por su preocupación.

A mis amigos (Helen, Dany, Jose Carlos, Arian, Danilo) y compañero por las nuevas experiencias.

A nuestros tutores y co-tutor por su gran ayuda, paciencia y dedicación.

A Damián y Katy por sus excelentes correcciones y valiosas sugerencias.

En general a todas las personas que aportaron su granito de arena para ayudarme tanto en este trabajo como en mi formación personal y profesional.

Ramón Fernandez Rabilero

Dedicatoria

Arian

Dedico este trabajo a mis padres, por haberme dado la vida y el mejor ejemplo que un hijo puede esperar, por ser la fuente de fuerza e inspiración en mi vida.

A mi hermano, que por mucho que trate, no puedo imaginar uno mejor. Es el mejor regalo que me han podido dar mis padres y sin él hoy no estaría aquí.

A mi abuela que es mi segunda madre. Aunque hoy lejos, siempre la tengo muy cerca y sin su ayuda me hubiera sido difícil llegar tan lejos. A mi tía, que siempre se ha sacrificado por mi bienestar y por el de toda la familia y la quiero mucho.

A Danilo, mi mejor amigo. Que jamás lo ha pensado dos veces para ayudarme cuando lo he necesitado y soportarme todos estos años.

A mi amigo Manuel, que fue el primero que tuve en la UCI y desde ese día se convirtió en parte de mi familia. Ramón, mi compañero de tesis incondicional en todo momento.

Y la lista de amigos interminable que los quiero mucho a todos: Abelito, Dashiel, Kilmer, Julio Cesar, Yisel, Neyvis, Alberto, Julio, José Carlos, Lázaro, Marlon, Arian, Gassiot.

Ramón

A mis padres, hermanas y hermano.

Resumen

La Universidad de las Ciencias Informáticas, como parte de su estructura organizacional, cuenta con el Departamento de transportaciones nacionales, que se encarga de la gestión de la transportación del centro mediante un sistema informático. Al respecto se realizó un estudio que permitió identificar varios problemas en ese sistema, tanto funcionales como conceptuales.

La presente investigación estuvo orientada a fundamentar y documentar el proceso de desarrollo del Sistema para la gestión de las transportaciones nacionales. Se realizó un análisis de fuentes bibliográficas relacionadas con la reservación de transporte y los sistemas encargados de realizar esta actividad en la red. Se corrigieron una serie de deficiencias registradas en entrevistas con el cliente y se añadieron nuevas funcionalidades, para facilitar el trabajo de los especialistas de Transportación. La solución forma parte del Sistema de Gestión del Ciudadano, requisito indispensable del cliente que no podía ser cumplido con la anterior aplicación.

El proceso fue guiado por la metodología de desarrollo de software Desarrollo Ágil con Calidad y se utilizó el marco de trabajo Symfony en su versión 2.5. El sistema gestor de base de datos utilizado fue el PostgreSQL y NetBeans como entorno de desarrollo. Para lograr una aplicación con la menor cantidad de errores, se le realizaron pruebas funcionales de integración, unidad y sistema.

Palabras clave: confirmación de transporte, gestión de transporte, sistema de gestión de transportaciones, transportación nacional.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Los medios de transporte tienen una relevancia extraordinaria en la sociedad porque permiten la circulación de bienes y personas, logrando una integración social que favorece el desarrollo. Por este motivo, con el paso del tiempo se ha visto una mejora en la eficiencia de estos, con servicios renovados y una utilización de recursos menor. En este sentido, existe en la actualidad un verdadero interés en lograr que los medios de transporte utilicen menor cantidad de energía, circunstancia en parte relacionada con los problemas que pueden existir en el futuro en lo que respecta a provisión de combustible. La informatización de los sistemas dedicados a la gestión de la transportación, ha sido una estrategia adoptada por gobiernos y administraciones para hacer un uso racional del combustible y de todos los recursos relacionados en este complejo tema de movilizar personal o recursos materiales.

En Cuba, la gestión del transporte y sus respectivas reservaciones están teniendo un gran auge, por ello se solicita fomentar la expansión de proyectos que agilicen y hagan más eficientes los trámites requeridos, facilitando a su vez el acceso a la información y garantizando su calidad. Por tal razón varias entidades han dedicado tiempo y recursos para impulsar la informatización de los procesos asociados a la gestión del transporte. Una de ellas es la Universidad de las Ciencias Informáticas (UCI), institución educacional creada a partir de una idea del Comandante en Jefe Fidel, la cual tiene como meta la formación de ingenieros informáticos y la potenciación de la industria cubana del software (1).

Un precepto de esta Universidad, era que debía contar con una representación de estudiantes de todos los municipios de la isla, con el objetivo de que ningún lugar del país, por recóndito que fuera, se privara del privilegio de contar con especialistas de las ciencias informáticas comprometidos con la Revolución y la tarea de informatizar la sociedad cubana. El número de estudiantes, trabajadores y profesores de la institución fue ascendiendo a medida que las necesidades del país así lo requerían. Pronto el centro contó en su residencia con un elevado número de personas procedentes de todos los municipios y provincias de Cuba.

La UCI, realizando un esfuerzo reconocible, cada año garantiza la transportación de todo el personal desde y hacia sus provincias de origen en los períodos vacacionales de fin de año y verano. Una tarea que ha delegado en la Dirección de Transportes, específicamente en el Departamento de transportaciones nacionales, siendo esta la estructura interna creada para dicho fin (2).

En este proceso deben ser llevadas a cabo diversas actividades para obtener el final esperado: reducir el tiempo en que se realiza la gestión de la transportación de estudiantes y profesores, y disminuir la carga de trabajo de los especialistas de la Dirección de Transportación de la UCI. Desde el 2012, el centro utiliza el Sistema de Confirmación para la Transportación Masiva (SCTM) para la gestión de sus transportaciones. Esta aplicación fue desarrollada por especialistas de la Dirección de Informatización (DIN) pero hoy no cumple con todas las expectativas del cliente ni de la comunidad universitaria.

La UCI cuenta con una infraestructura tecnológica que brinda servicios a una gran comunidad de usuarios de las Tecnologías de la Información y las Comunicaciones. Con el crecimiento del centro, se hizo necesaria la implementación de un significativo número de sistemas informáticos para gestionar los procesos sustantivos que ocurren allí. Como sucede frecuentemente, la mayoría de estos manejan información que es común para todos, tenerlos dispersos origina complicaciones para el usuario, que debe memorizar varias direcciones electrónicas y aprender a utilizar sistemas con diferentes formas de trabajo. La Universidad también se ve afectada por este fenómeno, pues debe dedicar recursos distintos para poner en explotación todas estas aplicaciones, algo que sería menos complejo si estuvieran integradas en un solo sistema de gestión.

La integración de sistemas informáticos tiene por objetivo conectar aplicaciones con la intención de compartir información entre ellas (3). Persigue además una manera eficiente y flexible de combinar recursos y reutilizar soluciones con el objetivo de perfeccionar su funcionamiento. La integración brinda una solución a problemas comunes en el desarrollo del software, cuando es necesario conectar sistemas aislados y posibilitar su funcionamiento como un único sistema.

Por esta razón, surge la iniciativa de crear el Sistema de Gestión del Ciudadano (SGC), con la finalidad de unificar en un solo sistema, los servicios que brinda la Universidad a su comunidad. Se desea hasta el momento, integrar en el SGC los sistemas encargados de gestionar la Guardia Obrera-Estudiantil, el Gas Licuado, el Control de Acceso y la Transportación de la UCI.

El primer problema identificado relacionado con la Transportación que impide integrar el SCTM al SGC está dado por las tecnologías con que están desarrollados ambos sistemas, pues si bien no es imposible lograr la comunicación entre estos utilizando una arquitectura orientada a servicios (SOA), hasta el momento no se cuenta con este tipo de infraestructura para llevar a cabo la tarea. La interfaz que presenta la aplicación

está divorciada de la estrategia de identidad marcaria propuesta por la UCI en el desarrollo de sus soluciones para lograr una homogeneidad y organización de sus productos.

El SCTM fue desarrollado en un corto período de tiempo para cubrir necesidades puntuales en la Universidad, por lo que no se documentó el proceso de desarrollo debidamente y no se detalló el proceso de confirmación y funcionamiento del sistema, el cual debe ser modificado en varios aspectos por cambios ocurridos en la ejecución de los procesos del negocio. La carencia de documentación hace que el soporte y actualización del actual sistema sea algo difícil. Lo que significa que se debe estudiar el código para luego poder optimizarlo o cambiarlo, una tarea que demanda gran cantidad de horas de trabajo y personal. Otro problema analizado es que, si el usuario es inexperto utilizando la aplicación, no podrá evacuar sus dudas pues no existe un manual para los usuarios, que brinde ayuda oportuna en la utilización del SCTM.

Durante pruebas realizadas al SCTM, se pudo constatar que se puede cambiar la forma de realizar algunas actividades en el sistema que faciliten el trabajo con la misma. Algunos ejemplos son: la gestión de familiares asociados, el bloqueo de usuarios en el sistema y la generación de reportes. Este sistema cuenta con el módulo de Seguridad que agrupa las funcionalidades Usuarios, Roles, Trazas y Filtros, funcionalidades que no serán necesitadas en la implementación del Sistema de Transportaciones, pues toda esa información será manejada por el SGC al igual que la seguridad en sí misma del sistema.

Por todo lo anterior planteado, se llega a la conclusión de que integrar el sistema que se utiliza actualmente para gestionar la transportación al SGC es una propuesta compleja que demandaría tiempo y recursos cuantiosos. Con el fin de solucionar la problemática planteada se ha formulado el siguiente **problema de investigación**: ¿Cómo lograr una correcta gestión de las transportaciones nacionales desde el Sistema de Gestión del Ciudadano de la Universidad de las Ciencias Informáticas?

Para dar respuesta a ello se centra el **objeto de estudio** en el proceso de desarrollo de aplicaciones web en la Universidad de las Ciencias Informáticas; enmarcando el **campo de acción** de la investigación en el proceso de gestión de la transportación nacional en la Universidad de las Ciencias Informáticas. Para guiar la búsqueda de la solución al problema planteado se define como **objetivo general**: desarrollar el Componente de Transportaciones Nacionales del Sistema de Gestión del Ciudadano de la Universidad de las Ciencias Informáticas.

Teniendo además como **objetivos específicos**:

- Analizar los principios teóricos de la investigación enfocados a los sistemas de gestión y de reservaciones.
- Obtener la propuesta del sistema de gestión de transportaciones de la UCI.
- Validar la solución desarrollada a partir de las pruebas de software.

Las **tareas generales** de la investigación se dirigen a:

- Caracterización del proceso de transportación nacional en la Universidad de las Ciencias Informáticas, que permita dar un mejor entendimiento y orientación al tema de la investigación.
- Estudio de sistemas similares existentes a nivel nacional e internacional relacionados con el campo de acción, que permita reutilizar las ventajas y detectar posibles errores.
- Análisis de las tecnologías a emplear en el proceso de desarrollo del software para obtener un producto con la calidad requerida.
- Identificación y validación de los requerimientos generados según el proceso de desarrollo aplicado para la realización de una propuesta de solución de un producto de software.
- Implementación de las funcionalidades identificadas para cumplir con el objetivo planteado.
- Diseño de los casos de pruebas para el sistema, de manera tal que permita evaluar el correcto funcionamiento de las funcionalidades desarrolladas.
- Ejecución de los casos de prueba para lograr detectar los posibles errores de implementación del sistema.

Se tiene como **Idea a defender** que, con el desarrollo del Componente de Transportaciones Nacionales, se logrará una correcta gestión de las transportaciones nacionales desde el Sistema de Gestión del Ciudadano de la Universidad de las Ciencias Informáticas.

Los métodos científicos que se emplearon durante la investigación fueron (4):

Métodos teóricos:

- **Histórico-Lógico:** se empleó en el análisis de la base teórica, la aplicación de este posibilitó el estudio de las herramientas de modelado de procesos y diseño de prototipos a utilizar, permitió identificar las características que presentan cada una de ellas y las utilidades que ofrecen en cuanto a ventajas y desventajas. También se utilizó para estudiar los sistemas de gestión de reservación de

transporte, describir el objetivo de dichos sistemas y comprobar si se ajustan o no al objetivo de la investigación.

- **Analítico-Sintético:** se utilizó para el proceso de análisis y estudio de la bibliografía empleada, arribando así a conclusiones que contribuyeron a elaborar la propuesta de solución, además para el estudio y análisis de las herramientas a utilizar.
- **Modelación:** este método permitió efectuar el análisis de la realidad existente mediante diagramas que ayudaron a comprender y unir datos referentes a la reservación de transporte, se utilizó en la elaboración de los diagramas de procesos donde se explicaron las actividades que se realizan en la reservación de transporte de la Universidad de las Ciencias Informáticas.

Métodos empíricos:

- **Entrevista:** método para fundamentar y precisar el problema a resolver. Fue utilizado para establecer las necesidades del cliente, identificar los procesos de transporte que se llevan a cabo actualmente en la UCI y las deficiencias que presentan. Las entrevistas fueron realizadas al jefe del Departamento de transportaciones nacionales, a una selección de profesores internos de la Dirección de Informatización y a estudiantes de quinto año de la Facultad 1.
- **Observación:** posibilitó verificar información obtenida a través de las entrevistas realizadas. Este método fue empleado para comprobar la manera en que se realiza el proceso de reservación de transporte en otros ámbitos y cómo funciona en el actual sistema.

La presente investigación está compuesta por una introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos y anexos. Los 3 capítulos están desglosados de la siguiente forma:

Capítulo 1: Estudio de sistemas de gestión de transporte y herramientas a utilizar en la solución.

Contiene el marco conceptual de la investigación y un estudio del estado actual de los sistemas informáticos para la gestión y reservación de transporte. De igual forma muestra el proceso de desarrollo que se utiliza y los diferentes lenguajes, tecnologías y herramientas a utilizar.

Capítulo 2: Análisis y diseño del Componente de Transportaciones Nacionales para el Sistema de Gestión del Ciudadano.

Se detalla el flujo actual de los procesos involucrados en el campo de acción, se identifican los requisitos del sistema y se describen las especificaciones de requisitos con sus prototipos de interfaz de usuarios. Además, se exponen los patrones de diseño y de arquitectura que utiliza el marco de trabajo seleccionado y se muestra el diseño de la base de datos del sistema.

Capítulo 3: Implementación y prueba del Componente de Transportaciones Nacionales para el Sistema de Gestión del Ciudadano.

Se formaliza la implementación de la propuesta de solución y se modela la vista de despliegue del sistema. A su vez se confeccionan los casos de prueba a realizar al software, se ejecutan y se presentan los resultados que permiten validar la solución propuesta.

Capítulo 1: Estudio de sistemas de gestión de transporte y herramientas a utilizar en la solución.

Introducción

En este capítulo se describen aspectos sobre otros sistemas de gestión de transporte existentes en Cuba y en el mundo. Se realiza un estudio de las principales tendencias de desarrollo sobre las que se apoyará la propuesta de solución. Se abordan las herramientas seleccionadas para el análisis y diseño de la aplicación, de forma tal que se cumplan todos los objetivos trazados.

1.1. Conceptos asociados al dominio del problema

Sergio Luján Mora define como aplicación web a aquellas soluciones informáticas que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador (5).

Jarel Remick describe aplicación web, como una aplicación que utiliza la red y las tecnologías de los navegadores web con el fin de lograr una o más tareas en una red, por lo general a través de un navegador (6).

Según Alejandro Peña, un sistema web de gestión de información, es un conjunto de elementos interrelacionados con el propósito de prestar atención a las demandas de información de una organización (7).

Otros autores como Armando Duany, definen un sistema web de gestión de información como un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. El cual realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información (8).

Teniendo en cuenta los conceptos anteriormente planteados, en la presente investigación se asume la definición de los autores, la cual plantea que “un sistema web de gestión de información es un sistema automatizado para proveer información que sustente determinadas funciones de gestión de una organización”.

Uno de los tipos de sistemas de gestión de información con mayor aceptación por parte de los usuarios son los sistemas de gestión de reservaciones, que no son más que un medio para acceder a los servicios de una empresa o institución en general, durante un tiempo determinado con antelación a que este sea

brindado. Las reservaciones son de gran importancia, pues reducen el riesgo de que el cliente no reciba el servicio y disminuyen el tiempo que este debe esperar para tener acceso al mismo (10).

1.2. Sistemas automatizados de reserva

A través de los años se ha visto una mejora a nivel mundial en el ámbito de los sistemas de reservación, con el fin de obtener las ventajas competitivas que genera el uso de estas herramientas, siendo Internet la plataforma e instrumento dominante que se utiliza para la promoción y distribución de los mencionados sistemas.

Aumentar la calidad de los servicios prestados, es el principal objetivo que persiguen las empresas cuando deciden implementar un sistema de este tipo que, entre otros servicios, pone a disposición de los usuarios: información sobre todos los servicios disponibles, reservas, listas de espera, tarifas especiales y normales, confeccionar itinerarios deseados, emisión de tarjetas de embarque y boletos, cancelación o modificación de la reserva, billete electrónico. No es difícil entonces saber que, haciendo todos estos trámites de forma automatizada, se ahorran grandes cantidades de dinero y recursos, tanto de la empresa como del cliente (11).

Actualmente existe una gran variedad de aplicaciones encargadas de gestionar las reservaciones de transporte. Por esta razón se hace necesario analizarlas, con el objetivo de definir si cubren las necesidades planteadas en la problemática descrita y si pueden ser adaptadas a la Universidad. En caso contrario, este estudio permitirá analizar todos los aspectos que puedan ser de utilidad para el desarrollo de una propuesta de solución, ajustada a las necesidades de la institución.

1.2.1. Proceso de reservación de transporte en el ámbito mundial

Mundialmente existen varias empresas de transporte que se encargan de brindar, en línea, la posibilidad de reservar pasajes mediante sistemas informáticos, proporcionándole al pasajero, mayor comodidad y seguridad al consumir dicho servicio. A continuación, serán descritos estos sistemas:

- ***Edreams*** es un sistema español que brinda la posibilidad de realizar reservaciones de viajes mediante diferentes transportes. Permite que el usuario reserve viajes de ida y vuelta, del origen y destino que escoja. Además, posibilita al cliente la ventaja de escoger viajar en vuelos directos, o sea, sin hacer escala hasta el destino final. Ofrece información sobre el viaje para que el pasajero sepa las posibilidades de reservación que tiene de forma general (12).

Aportes: el estudio de *Edreams* arrojó características tangibles que han sido de utilidad para la realización de la presente investigación. Entre estas se puede destacar la posibilidad de reservar viajes de ida y vuelta, desde un origen y hacia un destino predefinido en la aplicación. Además, brinda un mejor entendimiento acerca de la gestión de paradas intermedias, donde el pasajero podrá decidir e informar si desea quedarse en un sitio intermedio entre el origen y el destino del viaje. Este aspecto aplicado al sistema propuesto sería de gran utilidad para que los trabajadores o estudiantes que realicen su confirmación de viaje, puedan llegar a su destino desde una parada intermedia que les sea conveniente o el destino final del transporte.

¿Por qué no se utiliza?: a pesar de ser un sistema que posee muchas comodidades para el cliente, no se ajusta a las necesidades de la UCI, debido a que, en la Universidad, solamente se realiza la confirmación de la reservación predefinida para cada estudiante o trabajador. Esta reservación la Universidad la realiza en dependencia de la dirección actual de la persona. No realiza procesos de permuta de destinos, entre otras funcionalidades requeridas. Es una herramienta adaptada a las necesidades de grandes empresas de viajes que ofrecen otros servicios, pero no es aplicable a las características de la transportación de la UCI. Además, su licencia es propietaria.

- **JR Software Transporte de Pasajeros y Encomiendas** es un sistema de gestión para empresas de transporte de pasajeros, cuenta con módulos de Boletería para la venta de pasajes. Entre las prestaciones que propone se destacan: venta de boletos o pasajes, mostrando el plano de los colectivos o coches con sus butacas o asientos libres y reservados, posibilitando la anulación y liberación automática del asiento reservado. Permite relacionar a los pasajeros con empresas, a fin de imputar los pasajes a la cuenta de las empresas relacionadas. Además, lleva el estado de cuenta general de los clientes, dando la posibilidad de ver el histórico del cliente seleccionado y las facturas con saldos (13).

  Aportes: este software es un sistema de gestión que ya utilizan varias empresas. Brinda varias prestaciones que resultan interesantes en el marco del presente trabajo investigativo, entre ellas, la venta de boletos o pasajes online, la cancelación de una reservación y la liberación automática que genera esta capacidad en un transporte, dándole la posibilidad de viajar a un pasajero que esté en espera de reservación. Además, proporciona información acerca del histórico que se lleva de todos los datos manejados en el sistema.

¿Por qué no se utiliza?: el negocio que posibilita informatizar este sistema, no se corresponde con el llevado a cabo por el Departamento de transportaciones nacionales de la UCI. Tampoco muestra las funcionalidades de permuta de destinos y distribución automática de los viajes por sus transportes, las cuales son sumamente importantes en la Universidad. Unido a estos inconvenientes es válido destacar que el sistema es propietario.

- **Aeroflot** es un sistema automatizado ruso que responde a la aerolínea más grande de Rusia. Permite a los usuarios viajar a cualquier parte del mundo reservando viajes de ida y vuelta o de solo uno de estos, brindando la posibilidad de seleccionar la fecha de viaje deseada. Por su parte, el pasajero podrá especificar si viaja con niños y la cantidad que estos son, la clase en la cual desea viajar y si desea que su vuelo tenga o no escalas en alguna ciudad (14).

  Aportes: este sistema aporta conocimiento de interés en cuanto a la gestión de los viajes y las reservaciones realizadas por los pasajeros independientemente del tipo de viaje que quieran hacer, ya sea reservar la ida, el regreso, o ambos. También tributó información a la investigación la gestión de paradas intermedias que realiza.

  ¿Por qué no se utiliza?: Aeroflot es un sistema de licencia propietaria que utiliza la tecnología de *Sabre Airline Solutions*, la cual, aunque ofrece una numerosa gama de servicios y comodidades para el viaje, es muy costosa. Además, las funcionalidades que ofrece son solo para viajes en avión, careciendo de distribución automática de personas en los transportes y gestión de permutas de destinos; determinando de esta forma que tampoco satisface las necesidades del centro en cuestión.

1.2.2. Proceso de reservación de transporte en Cuba

Cuba, no queda exenta de los avances tecnológicos y de los aspectos positivos que traen consigo, ya que se reduce el tiempo empleado en gestionar cada reservación. Estos servicios son brindados desde el país y el usuario desde cualquier parte del mundo puede acceder a ellos. Entre los sistemas con los que cuenta Cuba se encuentran:

- El sistema de reservación de **Cubana de Aviación** está dedicado exclusivamente a la reservación de billetes de viaje de avión. Brinda al usuario la posibilidad de seleccionar el origen y el destino de su viaje, la clase en la que desea viajar e igualmente la fecha. Las ventas directas comprenden la reserva y el pago, este último deberá efectuarse usando una tarjeta de crédito *Visa* o *Mastercard*, enviando notificación al pasajero vía correo electrónico si la transacción fue exitosa (15).

Aportes: de este software se pudieron concretar algunas ideas para la investigación, como los tipos de viaje que se pueden reservar y la posibilidad de que el pasajero pueda seleccionar el viaje de su preferencia.

¿Por qué no se utiliza?: la implantación del sistema Cubana de Aviación en la UCI, no respondería a las necesidades de informatización del Departamento de transportaciones nacionales, pues los servicios que presta están en función del tipo de cliente de dicha aerolínea y de las especificidades de su negocio. Por ejemplo, permite distinción entre clases de viajes y ofrece la posibilidad de reservar hospedaje y transporte dentro de la isla. Tampoco presenta opciones de permuta de destinos ni de distribución automática.

- El sistema de reservas en línea de **Viazul** avala la seguridad de las reservaciones y las transacciones efectuadas. Este sitio realiza a través de una pasarela de pagos segura la liquidación de los boletos de avión, que encripta los datos de la tarjeta de crédito conocidos solamente por el cliente. Las reservaciones en línea de Viazul están garantizadas siempre que los clientes realicen el pago a través del sitio web. Quedarán guardados los datos personales y los datos de las reservas efectuadas a través del sitio (16).

  Aportes: el sistema muestra la posibilidad de escoger destinos ya predefinidos, lo cual puede ser tomado como referencia en el caso de la transportación general, dándole al trabajador de Transportaciones Nacionales la posibilidad de que escoja la opción de viaje más factible para cada viajero y el punto donde debe quedarse.

  ¿Por qué no se utiliza?: está destinado a fines turísticos, por lo que incluye opciones específicas de este tipo de negocios que le permiten diferenciarse de otras empresas de transportes en el país. Un ejemplo de ello es que, para deleite del cliente, permite la inclusión de rutas turísticas en la programación de sus viajes. Características como estas evidentemente no son demandadas en la UCI y a la vez, también es un software propietario.

1.2.3. Proceso de reservación de transporte en la UCI

Desde el propio año de creación de la Universidad de las Ciencias Informáticas, surgió la necesidad de contar con un sistema que hiciera más sencilla la tarea de movilizar a miles de estudiantes, profesores y trabajadores por toda Cuba.

Sistema de Reservación de Pase Masivo

Es en el año 2007 que el primer sistema informático para gestionar las Transportaciones Nacionales de la UCI queda oficialmente puesto en uso. El Sistema de Reservación de Pase Masivo brindó soporte a los procesos de la Dirección de Transporte, incluyendo además una serie de funcionalidades de administración del sistema, que permitían gestionar toda la información con la que esta dirección trabaja. Para todas sus acciones consultaba la información en una base de datos central que proporcionaba la información respecto a las personas que podrían reservar.

Para el desarrollo de esa aplicación Web se usó PHP como lenguaje de programación del lado del servidor, PostgreSQL como sistema gestor de base de datos y Apache como servidor de aplicaciones web. Se usó JavaScript del lado del cliente para lograr la interactividad con el usuario en el navegador y específicamente la técnica Ajax. La metodología RUP fue la seleccionada para planificar y controlar todo el proceso de desarrollo. Se usó además la arquitectura Cliente-Servidor (11).

Luego de un estudio realizado sobre este sistema, se llega a la conclusión de que era necesaria su renovación, debido a una serie de cambios que habían ocurrido durante el tiempo de explotación, pues se implementó teniendo en cuenta un grupo de características que tenía la Universidad en años anteriores y la modificación de estas dio al traste con el funcionamiento de la aplicación. Lo que trajo consigo que cada movilización de la institución, ya sea masiva o de profesores, requiriera de mucho tiempo para preparar las condiciones y tratar de arreglar los problemas que ocurrían con el uso de la aplicación.

Sistema de Transportación Nacional para la UCI

Este sistema brindó sus servicios a partir del 2009 en la Universidad de las Ciencias Informáticas para la gestión del transporte del personal en diferentes etapas del curso escolar. Estaba compuesto por un grupo de funcionalidades: gestionar reservaciones, definir bloques de entrada y salida, gestionar viajes, y distribuir el personal por transporte; aunque en reiteradas ocasiones esta actividad debió ser corregida manualmente. De este sistema se identificaron varias funcionalidades y características que serán objeto de un profundo análisis, debido a que deben ser tomadas en cuenta para la solución que se proponga. Ellas son la reservación, la distribución, la gestión de bloques y la gestión de la lista de espera.

El Sistema de Transportación Nacional para la UCI (STN-UCI) era un sistema poco flexible en cuanto a la configuración de sus funcionalidades. Fue construido atado a características y necesidades que poseía la

Universidad en el momento de su desarrollo. En consecuencia, el mismo contenía funcionalidades obsoletas, como la gestión del pase de fin de semana y la distribución por semestre del pase de profesores. A su vez, también requería nuevas funcionalidades como la gestión de la lista de espera de pasajeros bloqueados por reservación, la permuta entre pasajeros y una pre distribución de las personas a viajar por ómnibus. El marco de trabajo usado en el desarrollo fue GUUD en su versión 1.0; integrado por los marcos de trabajo Codelgniter y jQuery (2).

Sistema de confirmación para la transportación masiva

En la actualidad la gestión del pase masivo de la Universidad se realiza a través del Sistema de Confirmación para la Transportación Masiva (SCTM). La aplicación gestiona de forma integrada la información generada por el Departamento de transportaciones nacionales. Si bien este sistema ha resuelto el problema existente desde hace unos años atrás, luego de realizarle un estudio con el fin de integrarlo en el Sistema de Gestión del Ciudadano, se detectan una serie de deficiencias en su funcionamiento, que permiten llegar a la conclusión de que es necesaria la implementación de un nuevo sistema.

Uno de los problemas identificados que impide integrar el SCTM al SGC es la diferencia de tecnologías con que están implementados, pues la UCI no cuenta con una infraestructura orientada a servicios (SOA) que permita realizar una comunicación correcta entre ambos sistemas. En entrevista realizada a especialistas de la Dirección de Transporte, manifiestan que la aplicación que utilizan presenta algunas deficiencias que pueden ser corregidas, estas están relacionadas con lo complejo que es en el sistema realizar la gestión de los familiares de los viajeros, la creación de los transportes, la gestión de las fechas del calendario y el bloqueo de los usuarios que no utilizaron la trasportación que se les había asignado. El módulo de Seguridad del SCTM se hace innecesario pues al ser parte del SGC, este último se encargará de gestionar los permisos de los roles, los usuarios y la autenticación.

La interfaz además de ser poco amigable, no es todo lo descriptiva que se desea por lo que los usuarios año tras año hacen las mismas sugerencias de cambios para una mejor utilización del sistema. El SCG será desarrollado teniendo en cuenta las pautas de identidad marcaria definidas por la Universidad para sus productos, algo en lo que también difiere el SCTM, pues fue diseñado sin seguir estas pautas.

El sistema no cuenta con una documentación que describa en detalle el proceso de reservación, funcionamiento del sistema o un manual de usuario, por lo que realizar un proceso de soporte, algo que es

necesario con cada transportación que se realiza en la Universidad, se torna un proceso complejo de largas horas de trabajo.

Por todos los elementos anteriormente descritos, se llega a la conclusión de que realizar un soporte a la aplicación existente sería extremadamente difícil y demandaría hacer muchos cambios sobre el código existente lo que podría generar errores y horas de trabajo innecesarias. Se considera que desarrollar un nuevo sistema de Gestión de las Transportaciones Nacionales para el SGC es la alternativa más viable en aras de resolver el problema planteado en la investigación.

1.2.4. Resumen sobre el estudio realizado a los sistemas gestores de reservaciones de transporte

El estudio de los sistemas anteriormente descritos, permitió afirmar que los mismos no pueden ser usados para satisfacer las necesidades actuales del Departamento de transportaciones nacionales. Así mismo, evidenció la necesidad de desarrollar un nuevo sistema que permita la informatización de los procesos de negocio de dicho departamento. No obstante, se tendrán en cuenta para el nuevo desarrollo, las buenas prácticas que quedaron como saldo del estudio realizado. A continuación, se presenta una tabla resumen donde se recogen los datos más significativos de cada uno de los sistemas analizados.

Tabla 1 Características generales de los sistemas estudiados

CARACTERÍSTICAS A RESALTAR DE LOS SITEMAS ESTUDIADOS						
Sistemas	eDreams	JR SOFTWARE	AEROFLOT	CUBANA	VIAZUL	UCI RESERVACIONES
Licencia	**Propietaria**	**Propietaria**	**Propietaria**	**Propietaria**	**Propietaria**	**Libre**
Tecnología que utiliza	**Costosa**	**Costosa**	**Costosa**	**Costosa**	**Costosa**	**Asequible**
Integrable	**No**	**No**	**No**	**No**	**No**	**No**
Multiplataforma	**No**	**No**	**No**	**No**	**No**	**Sí**

1.3. Tecnologías, herramientas y metodología

La solución de software, se desarrolla utilizando las tecnologías y herramientas definidas por la Dirección de Informatización de la UCI, más específicamente las utilizadas para el desarrollo del Sistema de Gestión del Ciudadano, a la que se desea integrar. Esta aplicación establece utilizar como lenguaje de programación PHP, Symfony como marco de trabajo, PostgreSQL como gestor de base de datos y se hace uso de la metodología Desarrollo Ágil con Calidad (DAC).

Tecnologías

Notación para el Modelamiento de Procesos de Negocio (BPMN) 2.0

Business Process Modeling Notation (BPMN) es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (*workflow*). BPMN tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación. BPMN está planeada para dar soporte únicamente a aquellos procesos que sean aplicables a procesos de negocios (17).

PostgreSQL 9.4.1

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional distribuida bajo la licencia BSD[1]. Es compatible con una gran parte del estándar SQL y ofrece muchas características modernas como: consultas complejas, claves externas, disparadores, vistas actualizables e integridad transaccional. También, PostgreSQL se puede extender por el usuario en muchas maneras, por ejemplo, mediante la adición de nuevos tipos de datos, funciones, operadores, funciones de agregado y métodos de índice. Debido a la licencia liberal, PostgreSQL puede ser utilizado, modificado y distribuido por cualquier persona de forma gratuita para cualquier propósito, ya sea privado, comercial o académico (18).

HTML4

Es un lenguaje de hipertexto que permite escribir texto de forma estructurada y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento. Un documento hipertexto no sólo se compone de texto, puede contener imagen, sonido, vídeo, y el resultado puede considerarse como un

[1] La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*).

documento multimedia. Los documentos HTML deben tener la extensión html o htm, para que puedan ser visualizados en los navegadores. Los navegadores se encargan de interpretar el código HTML de los documentos y de mostrar a los usuarios las páginas web resultantes del código interpretado. El HTML 4 desarrolla el lenguaje HTML con mecanismos para hojas de estilo, ejecución de scripts, marcos, objetos incluidos, soporte mejorado para texto de derecha a izquierda y direcciones mezcladas, tablas más ricas y mejoras en formularios, ofreciendo mejoras de accesibilidad para personas con discapacidades (19).

PHP 5.6

PHP es un acrónimo recursivo que significa ***P****HP* ***H****ypertext* ***P****re-processor* y está publicado bajo la *PHP License.* La *Free* Software *Foundation* considera esta licencia como software libre. Es un lenguaje de programación, interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (*server-side scripting*) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt[2] o GTK+[3]. Es un lenguaje multiplataforma y tiene capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL (20).

JavaScript 1.8

JavaScript es un lenguaje de programación interpretado, se define como orientado a objetos y basado en prototipos. Es dinámico, responde a eventos en tiempo real como presionar un botón, pasar el puntero del ratón sobre un determinado texto o el simple hecho de cargar la página o caducar un tiempo. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de

[2] Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario.

[3] GTK+ o *The GIMP Toolkit* es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME.

JavaScript del lado del servidor (*Server-side JavaScript*). JavaScript puede incluirse en cualquier documento y es compatible con HTML en el navegador del cliente, ya sea PHP, ASP[4], JSP[5] y SVG[6] (21).

XML 1.0

XML (*eXtensible Markup Language)* es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML[7] y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna. Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. XML mejora la compatibilidad entre aplicaciones. Se puede comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos (22).

CSS 3

Las hojas de estilo en cascada (en inglés *Cascading Style Sheets*, CSS) constituyen un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo *"<style>".* CSS permite el control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo. Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario (19).

---

[4] Active Server Pages (ASP) es una tecnología de Microsoft del tipo "lado del servidor" para páginas web generadas dinámicamente.

[5]Java Server Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas.

[6] Gráficos Vectoriales Redimensionables (del inglés Scalable Vector Graphics) o SVG son una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados.

[7] SGML son las siglas de Standard Generalized Markup Language o "Estándar de Lenguaje de Marcado Generalizado".

Herramientas

Suite de Visual Paradigm 5.0

Suite de productos para desarrollar software de manera eficiente, rápida y de forma colaborativa. *Visual Paradigm (VP) Suite* es una solución total de modelado de procesos de negocio para la generación de código. *VP Suite* está diseñado para una amplia gama de usuarios, incluidos los ingenieros de software, analistas de sistemas, analistas de negocios, arquitectos de sistemas por igual, que están interesados en la construcción de sistemas de software a gran escala de forma fiable. *VP Suite* se compone de todos los mejores productos de Visual Paradigm incluyendo: Visual Paradigm for UML 8.0 Enterprise Edition, Smart Development Environment 6.0 Enterprise Edition y DB Visual ARCHITECT 6.0 Professional Edition (23).

NetBeans 8.0

Licencia del producto: Doble licencia; *Common Development and Distribution License (CDDL)* y *GNU General Public License* versión 2 *with Classpath exception (GPL2).* El Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para otros lenguajes de programación. Existe además un número importante de módulos para extenderlo. Es un producto libre y gratuito, sin restricciones de uso. Es de código abierto, escrito completamente en Java usando la plataforma NetBeans. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE[8], web, EJB[9] y aplicaciones móviles) (24).

PgAdmin III 1.22

Es una aplicación de código abierto distribuida bajo licencia BSD para el diseño y manejo de bases de datos. La aplicación se puede utilizar para manejar PostgreSQL 7.3 y superiores, funciona sobre casi todas las plataformas. Por su diseño permite, desde escribir consultas SQL[10] simples hasta desarrollar bases de datos

[8] *Plataforma Java 2, Standard Edition* (J2SE) es una colección de interfaces de programación de aplicaciones del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.

[9] *Enterprise JavaBeans* (también conocidos por sus siglas EJB) son una de las interfaces de programación de aplicaciones que forman parte del estándar de construcción de aplicaciones empresariales de *Java Platform, Enterprise Edition o Java EE*.

[10] SQL (*Structured query language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. Incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, por citar algunos. La conexión al servidor se hace mediante conexión TCP/IP[11] y puede encriptarse mediante SSL[12] para mayor seguridad (25).

Evolus Pencil 2.0.5

Pencil es una herramienta para elaborar prototipos de código abierto, disponible para todas las plataformas, construida con el propósito de ofrecer una herramienta de creación de prototipos, libre y de código abierto de interfaz gráfica de usuario que sea fácilmente de instalar y utilizar para crear maquetas de plataformas de escritorio populares. Tiene entre sus características principales que los proyectos pueden ser exportados en los formatos HTML, PNG (*Portable Network Graphics*, "Gráficos de Red Portátiles"), documento de Word y PDF (*Portable Document Format,* "Formato de Documento Portátil"), además, permite la instalación de plantillas definidas por el usuario, las operaciones de dibujo estándar: alinear, escalar y rotar y la adición de los objetos externos (26).

Nginx

Nginx es un servidor *web/proxy* inverso ligero de alto rendimiento y un *proxy* para protocolos de correo electrónico (IMAP/POP3).

Es software libre y de código abierto, bajo la licencia BSD simplificada; también existe una versión comercial distribuida bajo el nombre de Nginx plus3. Es multiplataforma, por lo que corre en sistemas tipo Unix (GNU/Linux, BSD, Solaris, Mac OS X, etc.) y Windows.(27)

JMeter 2.12

JMeter es un proyecto de Apache que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. JMeter puede

[11] TCP/IP es un conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre computadoras.

[12] *Secure Sockets Layer* o *SSL* es un protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet.

ser usado como una herramienta de pruebas unitarias para conexiones de bases de datos, FTP, LDAP y Servicios web (28).

Marco de trabajo

Symfony 2.5

Symfony es un marco de trabajo (*framework*) que tiene como objetivo fundamental automatizar los patrones más utilizados en la elaboración de sistemas web; además de obligar a clarificar a los programadores el código, establece un estándar de código legible, encapsula operaciones complejas en simples líneas de código, ahorrando mucho más tiempo a la hora de mostrar datos directamente de la base de datos. Este *framework* está realizado o implementado bajo el lenguaje PHP 5, ha sido utilizado en varias aplicaciones web obteniéndose de esta manera resultados satisfactorios. Es compatible con la mayoría de gestores de base datos existentes, se puede comentar sobre MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. También otorga facilidades en diferentes plataformas.

Características particulares de Symfony:

- Es lo suficientemente flexible para amoldarse a cualquier proyecto.
- Extensible para utilizar librerías de terceros.
- Es independiente del gestor de base de datos.
- La vista, compuesta por plantillas y capas, puede ser fácilmente modificada por diseñadores sin conocimientos de este *framework*. La utilización de *Helpers*, ayuda a tener un código más minimizado en la presentación.
- Tiene un caché muy potente, que evita la carga del servidor y reduce el ancho de banda. Los generadores de ABMs (*Asynchronous Balanced Mode*) o CRUDs (*Create, Read, Update and Delete*) hacen fácil la creación y validación de formularios, en conjunto con listas de registros y búsquedas con filtros.
- Maneja un completo sistema de prueba para depurar en modo desarrollo (29).

Metodología de desarrollo

Metodología Desarrollo Ágil con Calidad (DAC)

La metodología DAC es un proceso colaborativo, recursivo-iterativo, incremental y guiado por procesos y requisitos. Su modelo del proceso es una adaptación del modelo en Cascada a los modelos Programación Extrema y Desarrollo Concurrente.

DAC plantea que el problema una vez identificado y definido debe ser descompuesto en problemas más pequeños, y si es necesario, realizar con estos la misma operación.

Cada sub-problema será resuelto mediante un componente y el problema resuelto será el software o producto final; por lo que las entregas en DAC son a nivel de iteración, en la que habrá obligatoriamente un incremento del producto a partir de la solución de un componente del mismo. Además, en cada iteración se define como mínimo un hito a cumplir por cada fase del proceso. Al finalizar cada iteración se realiza la integración del componente al producto obtenido hasta el momento realizando pruebas de integración. Al finalizar las iteraciones se pueden realizar liberaciones del componente y transición del mismo dentro de la fase cierre de iteración.

Las iteraciones no tienen que desarrollarse todas al mismo tiempo, sino que al contar con un equipo pequeño este se va a ir moviendo de una iteración a otra a medida que estas vayan terminando de acuerdo a un orden de prioridad establecido en el plan del proyecto.

Este proceso tiene 8 actividades del marco de trabajo del proceso común llamadas fases o procesos del ciclo de vida: inicio, análisis y diseño arquitectónico, requisitos, construcción, cierre de iteración, liberación, transición y cierre, ocurriendo las iteraciones concurrentes entre las fases de requisitos, construcción y cierre de iteración. Además, entre las fases de requisitos y construcción puede ocurrir un ciclo pues a medida que los requisitos son especificados estos pueden ir entrando a la fase de construcción (30).

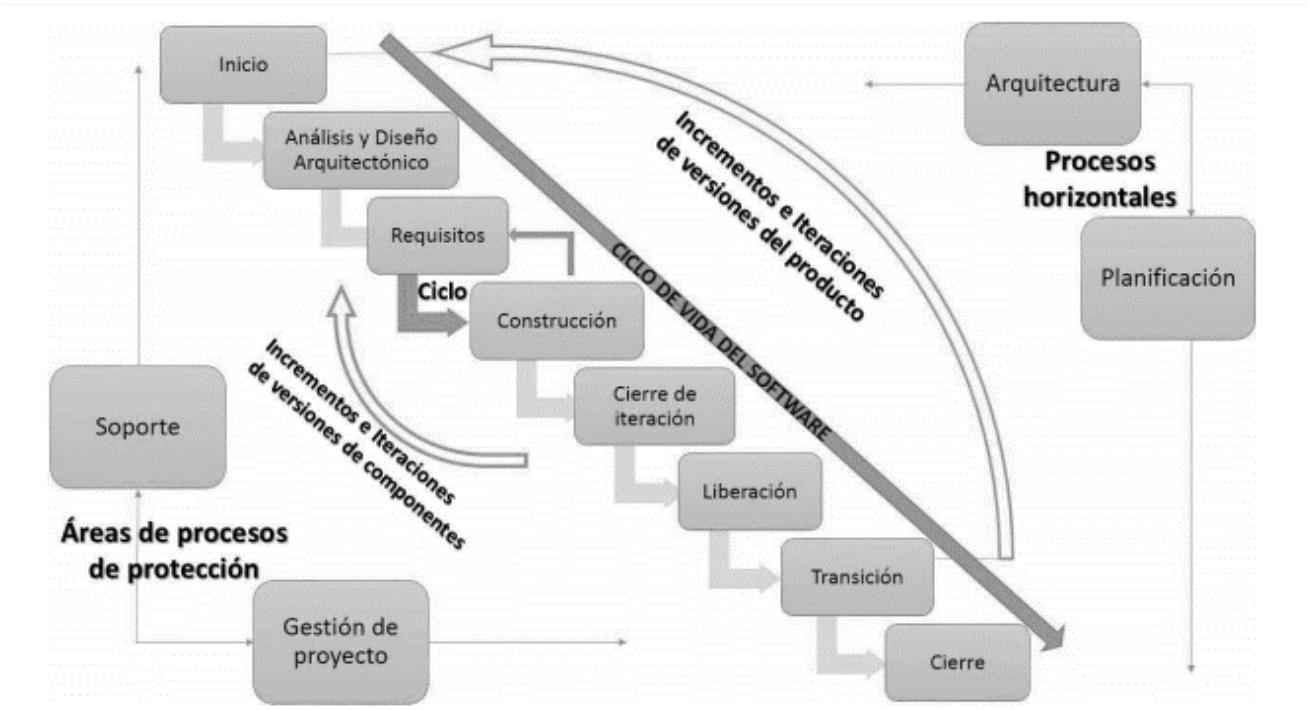


Figura 1 Metodología de Desarrollo DAC

1.4. Conclusiones parciales

El estudio de los sistemas de gestión de transporte existentes a nivel nacional e internacional, realizado en este capítulo, permitió conocer las principales características de dicho proceso. Se lograron fundamentar las bases teóricas de la investigación y se analizaron los conceptos básicos relacionados al objeto de estudio definido. La caracterización de las herramientas, tecnologías, metodología y lenguajes de programación, permitió conocer sus beneficios, así como formar las bases propicias para crear una propuesta de solución, que, a su vez, cumpla con el objetivo de la investigación.

Capítulo 2: Descripción y análisis de la solución propuesta

Introducción

En el capítulo que a continuación se presenta se exponen las tareas correspondientes al análisis y el diseño de la propuesta de solución. Se desarrolla el modelado del proceso de negocio para una mayor comprensión del entorno de investigación y se describen los conceptos fundamentales. Se especifican las técnicas de obtención de requisitos, identificando así los requisitos funcionales y no funcionales. Además, se realiza la descripción de la arquitectura, los patrones empleados y el diagrama de despliegue. Al finalizar el capítulo se podrá contar con una mejor comprensión acerca de la situación y características del negocio en cuestión.

2.1. Modelado del proceso de negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente, llevadas a cabo para generar productos y servicios. Modelar los procesos de negocio es una parte esencial de cualquier proceso de desarrollo de software. Permite capturar el esquema general y los procedimientos que gobiernan el negocio. Este modelo provee una descripción de dónde se va a ajustar el sistema de software considerado dentro de la estructura organizacional y de las actividades habituales. También provee la justificación para la construcción del sistema de software al capturar las actividades manuales y los procedimientos automatizados habituales que se incorporarán en el nuevo sistema (31).

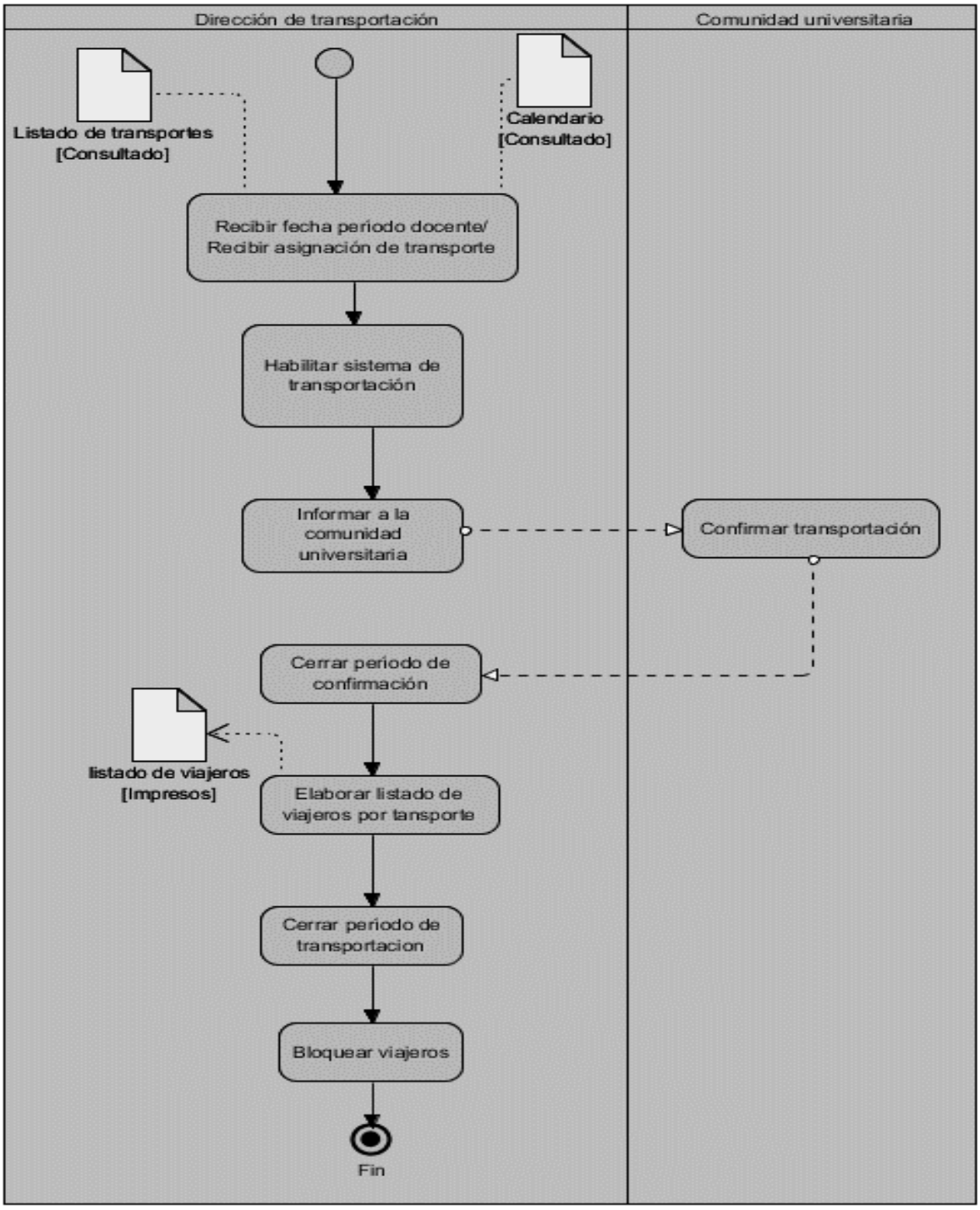


Figura 2 Modelo del Proceso de Negocio Transportación Nacional en la UCI.

2.1.1. Descripción del flujo del proceso de transportación

El proceso comienza cuando la Rectoría envía el ‘’Plan calendario de transportación’’ donde se especifican las fechas de la transportación masiva para los respectivos pases, entiéndase: pase fin de año y pase fin de curso. Con estas fechas se crea un plan de transportación, el cual es presentado al Ministerio de Transporte (MITRANS). De ser aprobado, es consultado por el MITRANS y se asignan los transportes a la Universidad, en caso contrario se cambian las fechas del plan, se confecciona uno nuevo y se presenta nuevamente.

Con los transportes asignados, la Dirección de Transporte habilita el sistema e informa a la comunidad universitaria que ya es posible realizar la confirmación de su transportación en el período establecido. Una vez que se cierra el período de confirmación, se elaboran los listados y se entregan a los responsables de cada transporte.

El responsable después de recibir los listados, verifica el uso de la transportación de cada viajero. Luego de culminado el período de transportación, el responsable de transporte entrega el listado a la Dirección de Transporte, que procede a bloquear el acceso al sistema a los usuarios que no usaron la transportación.

2.1.2. Reglas de negocio del proceso de transportación

Se han distinguido las siguientes reglas para lograr garantizar y cumplir las restricciones existentes en el negocio:

1. La Rectoría es la responsable de determinar las fechas del calendario de transportación.
2. El especialista de transporte es el único que puede elaborar el plan de transportación y luego darlo a conocer a la Universidad.
3. Todas las personas de la Universidad pueden obtener información sobre las salidas, tipo de transportes y personas que viajarán.
4. Solamente se podrá confirmar y cancelar la transportación en las fechas establecidas.
5. Cada viajero sólo puede hacer una confirmación por salida (ida/regreso).
6. El especialista de transporte es el único autorizado a realizar la distribución de los viajeros en los medios de transporte asignados (ómnibus, tren, barco).
7. Los familiares no pueden solicitar una permuta de destinos.
8. Una vez aceptada una permuta de destinos, se pierde el derecho a la capacidad de su destino origen, aunque se cancele la permuta.

9. Solamente se puede realizar una permuta de destinos entre dos viajeros que viajan en salidas con las mismas reglas.

2.2. Propuesta de solución

La propuesta de solución se basa en el desarrollo del Componente para la gestión de las transportaciones nacionales que será incluido en el Sistema de Gestión del Ciudadano de la Universidad de las Ciencias Informáticas, encargado de gestionar, entre otros, los procesos de reservación de gas licuado, la guardia obrera y el control de acceso. (Ver figura 3)

El sistema a desarrollar, facilitará en buena medida el trabajo de los especialistas de la Dirección de Transportaciones. Contará con nuevas funcionalidades que posibilitarán el aumento de la agilidad y organización de este proceso. Estará estructurado por 4 agrupaciones funcionales para un mejor orden de la aplicación. Confirmación: contendrá las funciones de confirmar y cancelar el transporte asignado, solicitar una permuta a un viajero específico y confirmar la transportación de los familiares asociados a un viajero. La permuta de destinos, es una opción dirigida a los viajeros que desean realizar un intercambio de los destinos de sus viajes. Se realizará a través del sistema por los propios usuarios.

El grupo funcional Gestión contendrá lo relacionado a gestionar los viajes, transportes, viajeros y salidas. La agrupación de Reportes será la encargada de gestionar los diferentes tipos de reportes que generará la aplicación. Por último, la agrupación Configuración será utilizada para gestionar diferentes datos que utilizará la aplicación para garantizar su funcionamiento, estos pueden ser: tipos de transporte, entidades transportistas y las reglas que son asociadas a las salidas, para especificar quién puede hacer uso de determinada salida y quién no.

En el sistema serán sustituidos los conceptos de reservación, bloque y salida por: confirmación, salida y viaje respectivamente. Esto se debe a que los especialistas de la Dirección de Transporte, acordaron que los nuevos conceptos describen de una mejor manera el proceso que representan en el sistema.

La solución será desarrollada teniendo en cuenta las pautas de identidad marcaria definidas por la Universidad, así como con tecnologías actualizadas, de código abierto y libres de licencia.

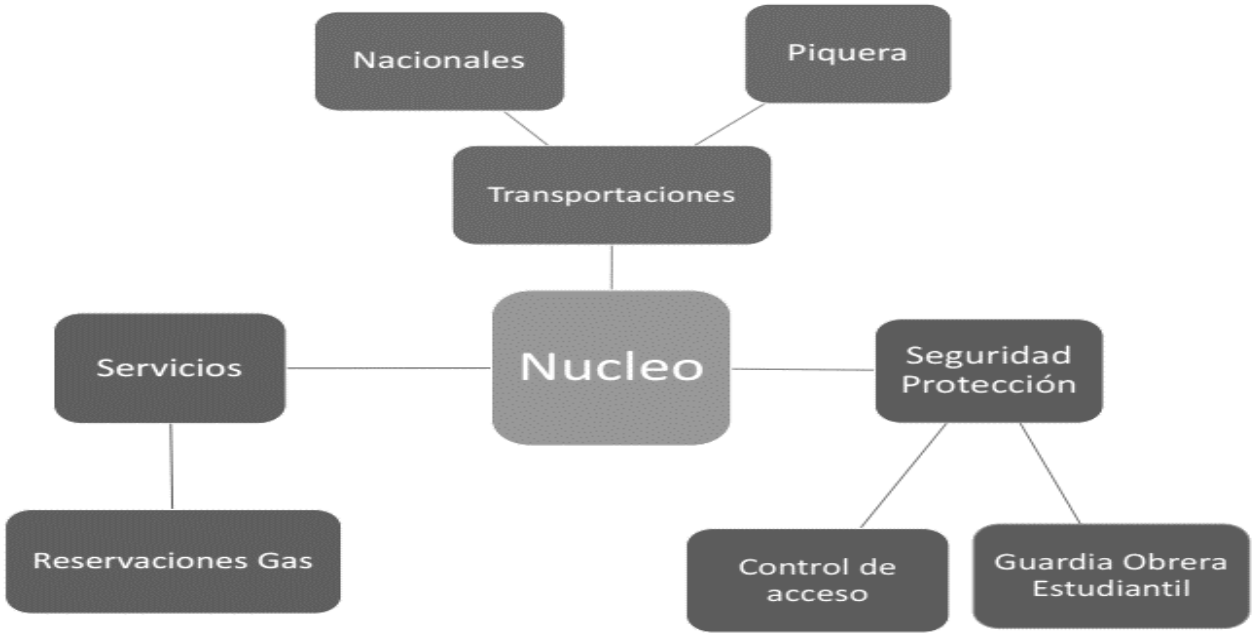


Figura 3 Estructura del Sistema de Gestión del Ciudadano

2.3. Requisitos de software

Un requisito de software es una característica que se debe exhibir en el software desarrollado o adaptado para solucionar un problema particular. A menudo los requisitos de sistemas de software se clasifican en funcionales y no funcionales (32):

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Definen los servicios que debe proporcionar el sistema, la forma en que este debe reaccionar a ciertas entradas y cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.

Los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona un sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

2.3.1. Identificación de requisitos

Para la identificación de requisitos existen varias técnicas que permiten establecer una correcta comunicación con los interesados y su equipo de trabajo. En principio se pregunta al cliente, a los usuarios y a los que están involucrados en los objetivos del sistema o producto, luego se investiga cómo los sistemas o productos se ajustan a las necesidades del negocio, y finalmente, cómo el sistema o producto va a ser utilizado en el día a día. A continuación, se describen las técnicas utilizadas durante el proceso de desarrollo del software para recopilar los requisitos de la propuesta de solución (33):

- **Entrevistas:** se realizaron a especialistas encargados del sistema que actualmente está en funcionamiento, lo que posibilitó la interacción con el experto en el área a tratar, para arribar en conjunto a la definición de los nuevos requisitos para el sistema a desarrollar. (Ver anexos 1 y 2).
- **Estudio de documentación:** varios tipos de documentación, como manuales y reportes, proporcionaron al analista información valiosa respecto a las organizaciones y a sus operaciones. La documentación, difícilmente refleja la forma en que realmente se desarrollan las actividades, o donde se encuentra el poder de la toma de decisiones. Sin embargo, pudo ser de gran importancia para introducir al analista al dominio de operación y el vocabulario que utiliza.
- **Sesiones de tormentas de ideas (*brainstorming*):** se realizaron reuniones en grupo con la participación de analistas, programadores, arquitectos e integrantes del proyecto con el objetivo de generar una gran variedad de vistas del problema y formularlo de diferentes formas.

A partir de la utilización de las técnicas anteriores se obtuvo un total de 48 requisitos funcionales y 25 no funcionales.

2.3.2. Requisitos funcionales

Para la descripción de los requisitos funcionales se utilizaron los artefactos propuestos por la metodología de desarrollo ágil DAC. En la Tabla #2 se muestran los requisitos funcionales que debe cumplir la solución para un total de 48, los cuales se dividen de acuerdo a la prioridad para el cliente en: 9 con prioridad alta, con prioridad media 19 y 20 con prioridad baja. A partir de la complejidad para desarrollar se clasifican en: 28 de complejidad media y 20 de complejidad baja.

Para determinar la complejidad de cada requisito funcional se tuvo en cuenta los criterios establecidos en el documento SGC_TRAN_ER_N_v1.1 del expediente de proyecto para la evaluación de requisitos de la metodología DAC. La complejidad de los requisitos se utiliza para estimar el esfuerzo de implementación

de estos y planificar en qué iteración se implementará. Para determinar la complejidad de los requisitos se analizó el número de transacciones realizadas, la complejidad por interfaces de comunicación con actores, el número de requisitos no funcionales asociados, el tipo de tecnología y la reutilización de los elementos ya existentes. En el caso de la prioridad se tuvieron en cuenta los beneficios para el cliente, la dependencia entre las funcionalidades, la estabilidad y la frecuencia con la que lo utilizará el cliente.

Tabla 2 Requisitos Funcionales. Prioridad y Complejidad

No.	Nombre	Prioridad	Complejidad
RF1	Crear viajero	Alta	Media
RF2	Listar viajero	Baja	Baja
RF3	Ver detalles de viajero	Baja	Baja
RF4	Adicionar familiar al viajero	Alta	Media
RF5	Eliminar familiar de viajero	Baja	Media
RF6	Ver detalles del familiar del viajero	Baja	Baja
RF7	Reservar a viajero	Alta	Media
RF8	Crear salida	Alta	Media
RF9	Eliminar salida	Baja	Baja
RF10	Modificar salida	Media	Media
RF11	Listar salidas	Baja	Media
RF12	Ver detalles de la salida	Baja	Baja
RF13	Crear Viaje	Alta	Baja
RF14	Listar viajes	Baja	Media
RF15	Modificar viaje	Media	Baja
RF16	Ver detalles de viaje	Baja	Media
RF17	Asociar reglas a salida	Alta	Baja
RF18	Adicionar transporte	Alta	Media
RF19	Listar transporte	Baja	Baja
RF20	Eliminar transporte	Baja	Media
RF21	Modificar transporte	Media	Media
RF22	Distribuir viajeros	Alta	Media

RF23	Bloquear usuario	Media	Media
RF24	Listar usuarios bloqueados	Baja	Baja
RF25	Ver detalles de bloqueo	Baja	Baja
RF26	Modificar bloqueo	Media	Media
RF27	Eliminar bloqueo	Baja	Media
RF28	Confirmar transportación familiar de viajero	Alta	Media
RF29	Cancelar transportación familiar de viajero	Media	Baja
RF30	Confirmar transportación	Alta	Media
RF31	Cancelar transportación	Media	Baja
RF32	Solicitar permuta de confirmaciones	Media	Media
RF33	Confirmar permuta de confirmaciones	Media	Media
RF34	Cancelar permuta de confirmaciones	Media	Baja
RF35	Crear tipo de transporte	Media	Media
RF36	Listar tipos de transportes	Baja	Baja
RF37	Modificar tipo de transporte	Media	Media
RF38	Ver detalles de tipo de transporte	Baja	Baja
RF39	Crear regla	Media	Media
RF40	Listar reglas	Baja	Media
RF41	Modificar regla	Media	Media
RF42	Ver detalles de regla	Baja	Baja
RF43	Crear entidad transportista	Media	Media
RF44	Listar entidades transportistas	Baja	Baja
RF45	Modificar entidad transportista	Media	Media
RF46	Ver detalles de entidad transportista	Baja	Media
RF47	Generar calendario	Media	Baja
RF48	Generar reportes de viajero	Media	Baja

2.3.3. Requisitos no funcionales

Los requisitos no funcionales, como su nombre sugiere, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la usabilidad, el tiempo de respuesta y la capacidad de almacenamiento (32).

El sistema propuesto contará con los siguientes requisitos no funcionales:

Usabilidad

1. Sólo se mostrarán a los usuarios aquellas acciones o informaciones del menú lateral a las que por su responsabilidad o rol dentro del negocio necesiten acceder.
2. El sistema provee una interfaz visual amigable, sencilla e intuitiva, que garantiza la facilidad para el trabajo de los usuarios. Para ello presenta una iconografía que se ajusta en cada caso con las funciones que representan, los botones tienen descripciones acordes a las funcionalidades que realizan y en los campos de textos editables por el usuario existen etiquetas flotantes que describen la acción que se debe realizar en cada caso.
3. El sistema debe presentar un menú lateral y una barra de íconos flotantes que permitan el acceso rápido a la información por parte de los usuarios.
4. La información que se registra estará protegida de modificaciones no deseadas de acuerdo a la estrategia de seguridad definida.
5. Las vistas del sistema deben indicar en cada momento la acción que se está realizando, así como los íconos deben estar representados por una imagen acorde a la acción que se realiza mediante el mismo.
6. Para el uso del sistema se requiere una PC cliente con los siguientes sistemas operativos: Windows o GNU/Linux.

Software

7. Para el uso del sistema se requiere una PC cliente con el navegador web Mozilla Firefox 26 o superior para el uso de la aplicación web.
8. Para el despliegue del sistema se debe contar en el servidor de aplicaciones web con: PHP v 5.6 con las librerías php5-ldap, php5-gd, php5-mcrypt, php5-pgsql, php5-xsl, php5-*openssl* y Apache 2.2 con el módulo *rewrite* activado.

Confiabilidad

9. El tratamiento de las excepciones permitirá un seguimiento hasta guardar información acerca del lugar dónde se produjo el error y de los parámetros de configuración del sistema que lo provocaron. Cuando ocurre una excepción el sistema mostrará un mensaje explicativo del error ocurrido y permanecerá en el mismo estado sin realizar ninguna otra operación.
10. El servidor de aplicaciones y de base de datos deberá mantener una seguridad mediante firewall para proteger el código y la información.

Eficiencia

11. El sistema deberá tener por cada transacción un tiempo de respuesta promedio de 1,5 segundos y un máximo de 10 segundos.

Soporte

12. El sistema se regirá por un estándar de código debidamente documentado en el expediente de proyecto.
13. Todos los productos de trabajo generados en el desarrollo del software se regirán por unas pautas de configuración debidamente documentadas en el expediente de proyecto.
14. El software estará acompañado de un manual técnico y de 5 vistas de arquitectura que apoyen el soporte de la solución.

Restricciones de diseño

15. El software deberá ser desarrollado en su totalidad con tecnologías y componentes de código abierto.

Interfaz

16. Existirá un componente interno para la validación correcta de las entradas/salida de datos de cada una de las interfaces de la aplicación.
17. El sistema deberá implementar una solución arquitectónica para permitir la integración segura entre componentes internos.
18. La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.

Hardware

19. Para la ejecución del sistema se requiere que la PC cliente tenga los siguientes componentes de hardware: Pentium 4 o superior, 512 MB RAM como mínimo.
20. La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTPS.

2.3.4. Descripción de requisitos

Todos los requisitos se describieron formalmente mediante las técnicas de escenario y prototipo. A continuación, se presenta como ejemplo la descripción del requisito “Crear salida”, el resto de las descripciones realizadas se encuentran en el expediente del proyecto.

Tabla 3 Descripción de requisito ¨Crear salida¨

Precondiciones	No existe la salida
Flujo de eventos	
Flujo básico	
1.	El usuario una vez autenticado en el Sistema de Gestión del Ciudadano selecciona el Sistema de Transportación. El usuario selecciona el módulo Nacionales. El sistema muestra todas las agrupaciones de contenido de este módulo. El usuario selecciona la agrupación Gestión. El sistema muestra todas las funcionalidades de esta agrupación funcional. El usuario escoge la opción Gestionar salida.
2.	En el área de íconos flotantes selecciona la opción Crear.
3.	Se muestra una interfaz con los campos a llenar por el usuario.
4.	El usuario deberá llenar los campos marcados como obligatorios.
5.	El sistema muestra un mensaje de información con el texto:¨ El elemento ha sido creado satisfactoriamente¨ y regresa a la interfaz crear.
Poscondiciones	
1.	Una vez creado el elemento se mantiene la misma interfaz por si se desea crear otro elemento.
Flujos alternativos	
1.	En caso que el elemento ya exista se muestra un mensaje de error “El elemento ya existe”.
2.	En caso de cancelar la acción se muestra un mensaje de advertencia “¿Está seguro de realizar la acción? “.
3.	En caso de que el usuario seleccione la acción de listar en el área de íconos flotantes el sistema mostrará el listado con todos los viajeros.
4.	En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo encima del componente “Campo requerido” en el campo que debe ser llenado obligatorio.
5.	Si se introducen más caracteres de los permitidos para una palabra el sistema muestra el mensaje “Ha excedido el número de caracteres permitidos para una palabra”.
6.	Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.
7.	Si se introducen menos caracteres del mínimo permitido se muestra el mensaje de error “Entre al menos 2 caracteres”.

Conceptos		
	Nombre de la salida	Se introduce el nombre de la salida
	Tipo de salida	Selecciona el tipo de salida que puede ser de Ida o Regreso.
	Inicio del período de confirmaciones.	Fecha a partir de la los usuarios podrán confirmar la utilización del transporte asignado.
	Fin de período de confirmaciones	Fecha hasta la cual los usuarios podrán utilizar la aplicación para confirmar su transporte.
	Cancelación de la confirmación	Fecha hasta la cual los usuarios podrán cancelar una confirmación realizada para el viaje de ida o regreso.
	Fecha inicio de los viajes	Fecha estimada en la que deberán comenzar los viajes
	Fecha fin de los viajes	Fecha estimada en la que deben haber ocurrido todos los viajes
	Entrega del pasaje	Fecha en la que serán entregados los pasajes a los viajeros que lo requieran (tren).
	Envío de correo de ubicación	Fecha de envío del correo a los usuarios confirmados con la información de su ubicación en los transportes.
	Habilitado	Se podrá seleccionar si la salida está activa o no en el sistema para ser utilizada.
	Concepto de regla	Se especifica la regla deseada.
	Valor	Se selecciona uno de los valores de la lista desplegable.

Prototipo de interfaz

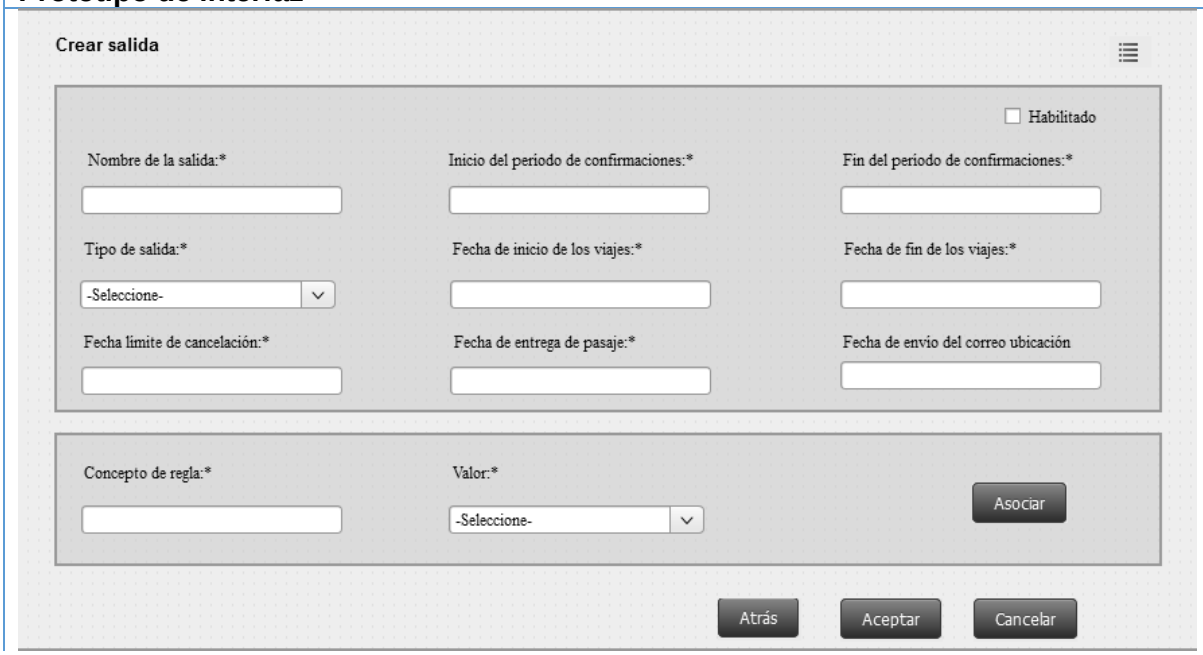


Tabla 4 Descripción de requisito ¨Crear viajero¨

Precondiciones	No existe el viajero
Flujo de eventos	
Flujo básico	
1.	El usuario una vez autenticado en el Sistema de Gestión del Ciudadano selecciona el Sistema de Transportación. El usuario selecciona el módulo Nacionales. El sistema muestra todas las agrupaciones de contenido de este módulo. El usuario selecciona la agrupación Gestión. El sistema muestra todas las funcionalidades de esta agrupación funcional. El usuario escoge la opción Gestionar viajero.
2.	Se muestra una interfaz con el listado de los viajeros registrados en el sistema.
3.	El usuario selecciona la opción Crear en el área de íconos flotantes.
4.	Se muestra una interfaz con campos a llenar para crear el viajero.
5.	El usuario introduce los datos del viajero y escoge la opción Aceptar.
6.	El sistema muestra un mensaje de información con el texto:¨ El elemento ha sido creado satisfactoriamente¨ y regresa a la interfaz crear.
Poscondiciones	
1.	Se puede gestionar el viajero
2.	Se mantiene en la interfaz crear viajero por si se desea crear otro.
Flujos alternativos	
1.	En caso que el elemento ya exista se muestra un mensaje de error “El elemento ya existe”.
2.	En caso de cancelar la acción se muestra un mensaje de advertencia “¿Está seguro de realizar la acción? “.
3.	En caso de que el usuario seleccione la acción de listar en el área de íconos flotantes el sistema mostrará el listado con todos los viajeros.
4.	En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo encima del componente “Campo requerido” en el campo que debe ser llenado obligatorio.
5.	Si se introducen más caracteres de los permitidos para una palabra el sistema muestra el mensaje “Ha excedido el número de caracteres permitidos para una palabra”.
6.	Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.
7.	Si se introducen menos caracteres del mínimo permitido se muestra el mensaje de error “Entre al menos 2 caracteres”.
Validaciones	
1.	**Nombre** Campo requerido. Solo admite letras, números y espacio o guion bajo entre las palabras. Cantidad de caracteres por palabra es 30. Cantidad de caracteres máxima es 250 y mínima es 2.
2.	**Carné de identidad** Campo requerido Solo admite números. Cantidad de caracteres máxima 11.
3.	**Provincia** Campo requerido. Campo de selección.

4.	**Municipio** Campo requerido. Campo de selección.	
Conceptos		
	Nombre del viajero	Nombre de la persona.
	Carné de identidad	Número de carné de identidad.
	Provincia destino	Provincia hacia dónde va a viajar la persona.
	Municipio destino	Municipio hacia dónde va a viajar la persona.
Prototipo de interfaz		

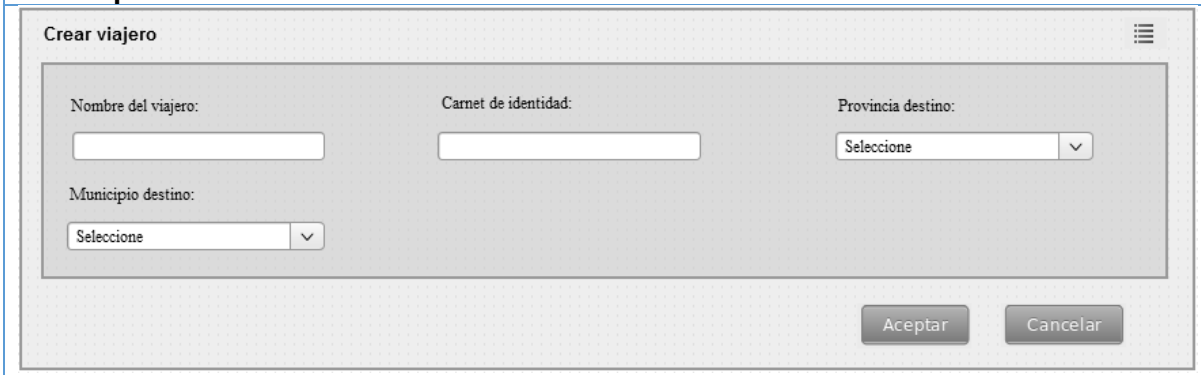
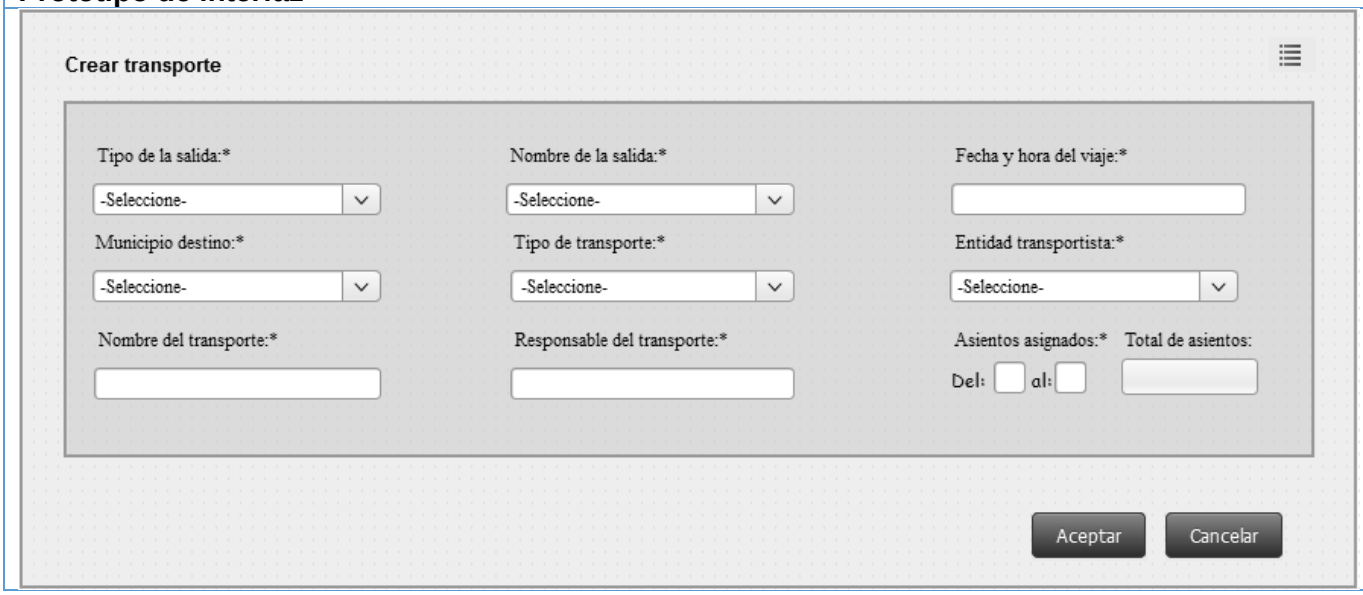


Tabla 5 Descripción de requisito ¨Crear transporte¨

Precondiciones	No existe el transporte	
Flujo de eventos		
Flujo básico		
1.	El usuario una vez autenticado en el Sistema de Gestión del Ciudadano selecciona el Sistema de Transportación. El usuario selecciona el módulo Nacionales. El sistema muestra todas las agrupaciones de contenido de este módulo. El usuario selecciona la agrupación Gestión. El sistema muestra todas las funcionalidades de esta agrupación funcional. El usuario escoge la opción Gestionar transporte.	
2.	Se muestra una interfaz con el listado de los viajeros registrados en el sistema.	
3.	El usuario selecciona la opción Crear en el área de íconos flotantes.	
4.	Se muestra una interfaz con campos a llenar para crear el transporte.	
5.	El usuario introduce los datos del transporte y escoge la opción Aceptar.	
6.	El sistema muestra un mensaje de información con el texto: ¨El elemento ha sido creado satisfactoriamente¨ y regresa a la interfaz crear.	
Poscondiciones		
1.	Se puede gestionar el transporte	
2.	Se mantiene en la interfaz crear transporte por si se desea crear otro.	
Flujos alternativos		
1.	En caso que el elemento ya exista se muestra un mensaje de error “El elemento ya existe”.	
2.	En caso de cancelar la acción se muestra un mensaje de advertencia “¿Está seguro de realizar la acción? “.	
3.	En caso de que el usuario seleccione la acción de listar en el área de íconos flotantes el sistema mostrará el listado con todos los viajeros.	
4.	En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo encima del componente “Campo requerido” en el campo que debe ser llenado obligatorio.	
5.	Si se introducen más caracteres de los permitidos para una palabra el sistema muestra el mensaje “Ha excedido el número de caracteres permitidos para una palabra”.	
6.	Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.	
7.	Si se introducen menos caracteres del mínimo permitido se muestra el mensaje de error “Entre al menos 2 caracteres”.	
Validaciones		
Conceptos		
	Tipo de viaje	Se selecciona en el menú desplegable el tipo de viaje que puede ser de Ida o de Regreso
	Viaje	Se selecciona el viaje al cual pertenecerá el transporte.
	Fecha de salida	Se selecciona la fecha y hora de la salida del transporte.
	Tipo de transporte	Se selecciona el tipo de transporte. Puede ser ómnibus o tren.
	Entidad	Se selecciona la Entidad a la cual pertenece el transporte especificado con anterioridad. Las entidades son definidas previamente mediante nomencladores.
	Nombre	Nombre para identificar el transporte.

	Responsable del transporte	Se introduce el nombre de la persona que será responsable del transporte.
	Asientos asignados	Se especifica el rango de asientos disponibles en ese transporte para ser utilizados.
	Total de asientos	Muestra el total de asientos con que cuenta el transporte.

Prototipo de interfaz



2.4. Descripción de la arquitectura

Según Pressman, la arquitectura de software de un programa o sistema de cómputo es la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de los componentes y las relaciones entre ellos. Es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de los componentes (34).

2.4.1. Arquitectura

La arquitectura Cliente-Servidor es un modelo de aplicación distribuida, donde las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de

tipo organizativo debido a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema (29).

Cliente: permite al usuario formular los requisitos y pasarlos al servidor. Maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario, además de acceder a los servicios distribuidos en cualquier parte de una red. Algunas de sus funciones son administrar la interfaz de usuario, interactuar con el usuario, procesar la lógica de la aplicación, hacer validaciones locales, generar requisitos de bases de datos, recibir resultados del servidor y formatear resultados.

Servidor: encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Las funciones del servidor son aceptar los requisitos de bases de datos que hacen los clientes, procesar esos requisitos, formatear datos para trasmitirlos a los clientes, procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

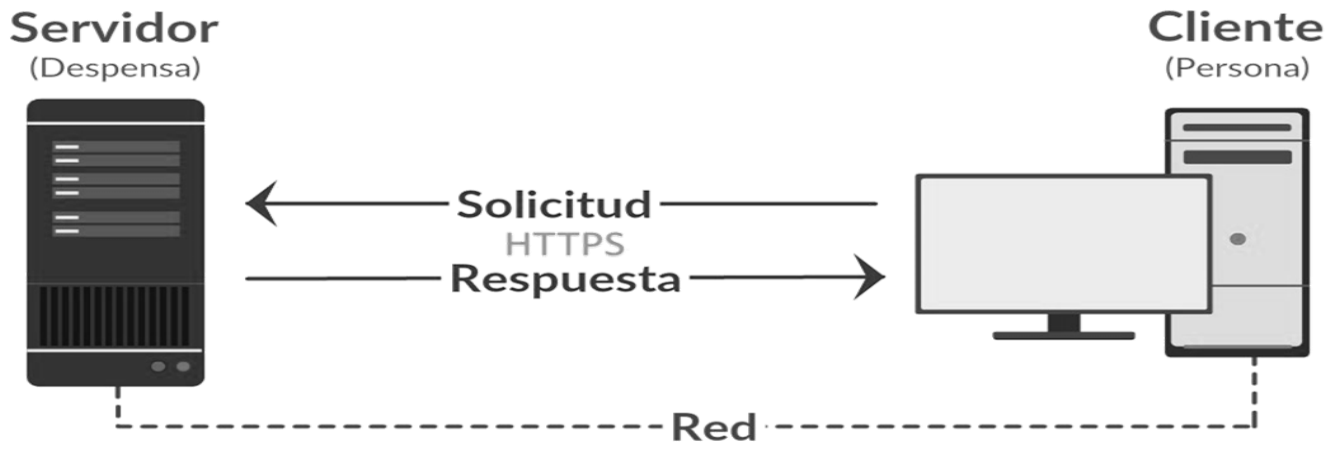


Figura 4 Arquitectura cliente-servidor

2.4.2. Patrón de Arquitectura

Los patrones arquitectónicos son patrones del software, que se encargan de definir la estructura de un sistema. Estos a su vez se componen de subsistemas con sus responsabilidades, también poseen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño. Un patrón arquitectónico se enfoca en dar solución a un problema en específico y abarca solo parte

de la arquitectura. Cuando un patrón de este tipo brinda una imagen general de un sistema, él no es una arquitectura como tal. Es por esto que un patrón arquitectónico es un concepto que captura elementos esenciales de una arquitectura de software (35).

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC (Modelo-Vista-Controlador), que está formado por tres niveles:

El **Controlador**, que contiene el código que liga la lógica de negocio con la presentación, está dividido en varios componentes que se utilizan para diversos propósitos:

- El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse.
- Las acciones contienen la lógica de la aplicación. Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación.
- Los objetos *request*, *response* y *session* dan acceso a los parámetros de la petición, las cabeceras de las respuestas y a los datos persistentes del usuario. Se utilizan muy a menudo en la capa del controlador.
- Los filtros son trozos de código ejecutados para cada petición, antes o después de una acción. Por ejemplo, los filtros de seguridad y validación son comúnmente utilizados en aplicaciones web. (36)

La **vista** se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparadas para que puedan ser fácilmente modificables por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones. (36)

En las aplicaciones Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; de esta forma nunca se accede de forma explícita a la base de datos. Este comportamiento permite un alto nivel de abstracción y permite una fácil portabilidad. (36)

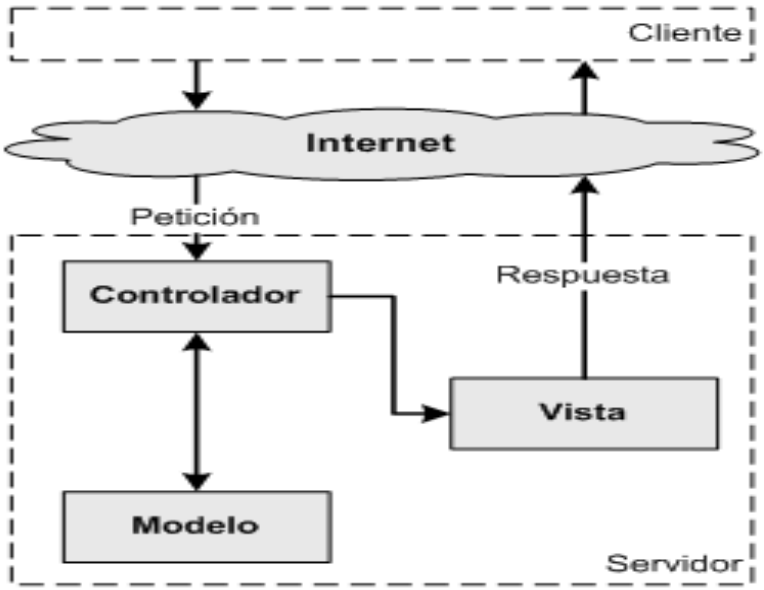


Figura 5 Patrón de Arquitectura Modelo-Vista-Controlador

2.5. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. El mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Estos modelos que se presentan como parejas de problema/solución con un nombre, codifican buenos principios y sugerencias relacionados con la asignación de responsabilidades, basados en la recopilación del conocimiento de los expertos en desarrollo de software (37).

2.5.1. Patrones Generales de Asignación de Responsabilidades de Sistemas

Los Patrones Generales de Asignación de Responsabilidades de Sistemas (GRASP, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar software orientado a objetos (38). Esos no introducen ideas novedosas, sino que son la codificación de los principios básicos más usados. Los patrones GRASP están compuestos por: experto, creador, bajo acoplamiento, alta cohesión y controlador.

A continuación, se ejemplifican los patrones utilizados en la propuesta de solución.

- **Experto**: es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Doctrine para mapear la Base de Datos. Symfony utiliza esta librería para realizar su

capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan (39).

\UCI\DIN\Trans\Masivas\ConfiguracionBundle\Entity\NTipoSalida

```php
<?php

/** Created by PhpStorm. ...*/
namespace UCI\DIN\Trans\Masivas\ConfiguracionBundle\Entity;


use Doctrine\ORM\Mapping as ORM;
use \DateTime;

/** Class NTipoSalida ...*/
class NTipoSalida
{
    /** @var integer ...*/
    private $id;

    /** @var string ...*/
    private $nombre;

    /** @var DateTime ...*/
    private $fechaRegistro;
    /** @var string ...*/
    private $descripcion;

    /** @var boolean; ...*/
    private $activo =  true;

    public function __construct(){...}

    /** @return int ...*/
    public function getId(){...}
```

Figura 6 Patrón experto

- **Creador**: este patrón como su nombre lo indica es el que crea, ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos. Es donde se asigna la responsabilidad de que una clase B cree un objeto de la clase A. (39)

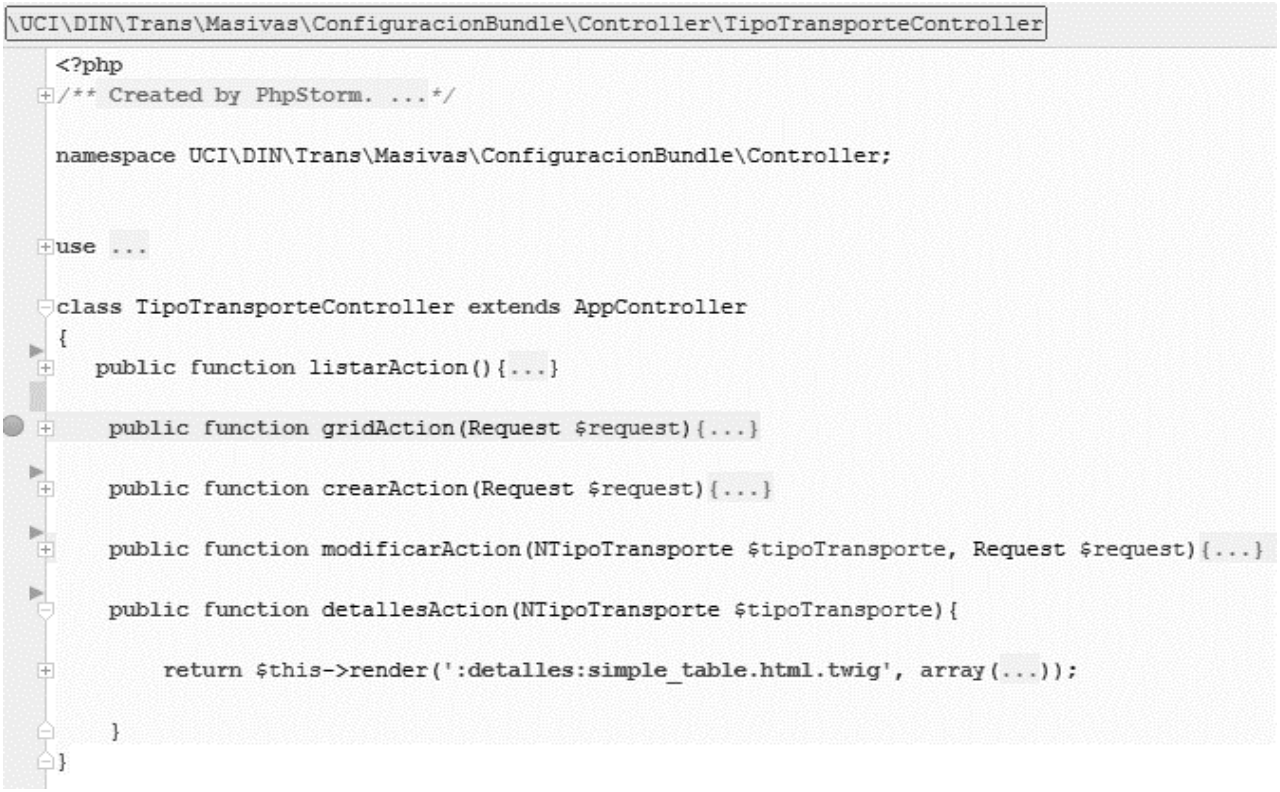


Figura 7 Patrón creador

- **Controlador**: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. El uso de este patrón se refleja en las clases controladoras pertenecientes al sistema. (39)

```
trans_masivas_configuracion_tt_homepage:
    path:     /index
    defaults: { _controller: TMasivasConfiguracionBundle:TipoTransporte:listar }

trans_masivas_configuracion_tt_grid:
    path:     /grid
    defaults: { _controller:  TMasivasConfiguracionBundle:TipoTransporte:grid }
    options:
      expose: true
trans_masivas_configuracion_tt_create:
    path:     /create
    defaults: { _controller: TMasivasConfiguracionBundle:TipoTransporte:crear }
    options:
      expose: true

trans_masivas_configuracion_tt_modificar:
    path:     /{id}/modificar
    defaults: { _controller: TMasivasConfiguracionBundle:TipoTransporte:modificar }
    options:
      expose: true
      write: true

trans_masivas_configuracion_tt_detalles:
    path:     /{id}/detalles
    defaults: { _controller: TMasivasConfiguracionBundle:TipoTransporte:detalles }
    options:
      expose: true
      read: true
```

Figura 8 Patrón controlador

2.5.2. Patrones GoF

Los patrones "Banda de los Cuatro" (*Gang-of-Four*) describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas. Existen tres tipos de patrones: de creación, estructurales y de comportamiento. Los patrones de creación abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas (40).

- **Instancia única (*Singleton*):** es un patrón diseñado para restringir la creación de objetos pertenecientes a una clase u objeto. Su intención consiste en garantizar que una clase sólo tenga

una instancia y proporcionar un punto de acceso global a ella. Este patrón se refleja en las clases controladoras que son instancias únicas para la interacción entre componentes.

- **Mediador (*Mediator*):** define un objeto que coordine la comunicación entre objetos de distintas clases. Se refleja en las librerías que funcionan como mediadoras entre las clases controladoras y las modelos de acceso a datos.
- **Observador (*Observer*):** Este patrón se encarga de definir dependencia entre objeto, de forma que, si alguno cambia su estado, automáticamente se notifica y actualizan todos los objetos que dependen de él.

2.6. Patrones de diseño de base de datos

Para el diseño y la construcción de una base de datos se requiere del mayor análisis posible pues a partir de este diseño se crea la base de datos, en la actualidad suelen ser muy grandes y a veces el trabajo con los patrones de diseño hacen que el trabajo sea más fácil además asegura un resultado correcto (41).

Llaves subrogadas

Este patrón es muy utilizado por facilitar la interacción con la base de datos en un futuro. El mismo plantea que se genere una llave primaria única para cada entidad, en vez de usar un atributo identificador en el contexto dado. Normalmente se usan enteros en columnas *identity* o GUID (*Global UniqueIdentifier*) que están demostradas que no se repiten o con una probabilidad extremadamente baja. Esto permite que las tablas sean más fáciles de consultar a partir del identificador, pues todos tienen el mismo tipo en cada una de las tablas.

Figura 9 Llave subrogada

2.7. Modelo de la base de datos

Un modelo de datos es una colección de conceptos y reglas que se emplean para describir la estructura de una base de datos que incluye entidades, atributos y relaciones entre estos (42). Para diseñar la base de datos se elaboró el modelo físico de datos. Las clases persistentes que a continuación se muestran cubren las necesidades de la propuesta de solución.

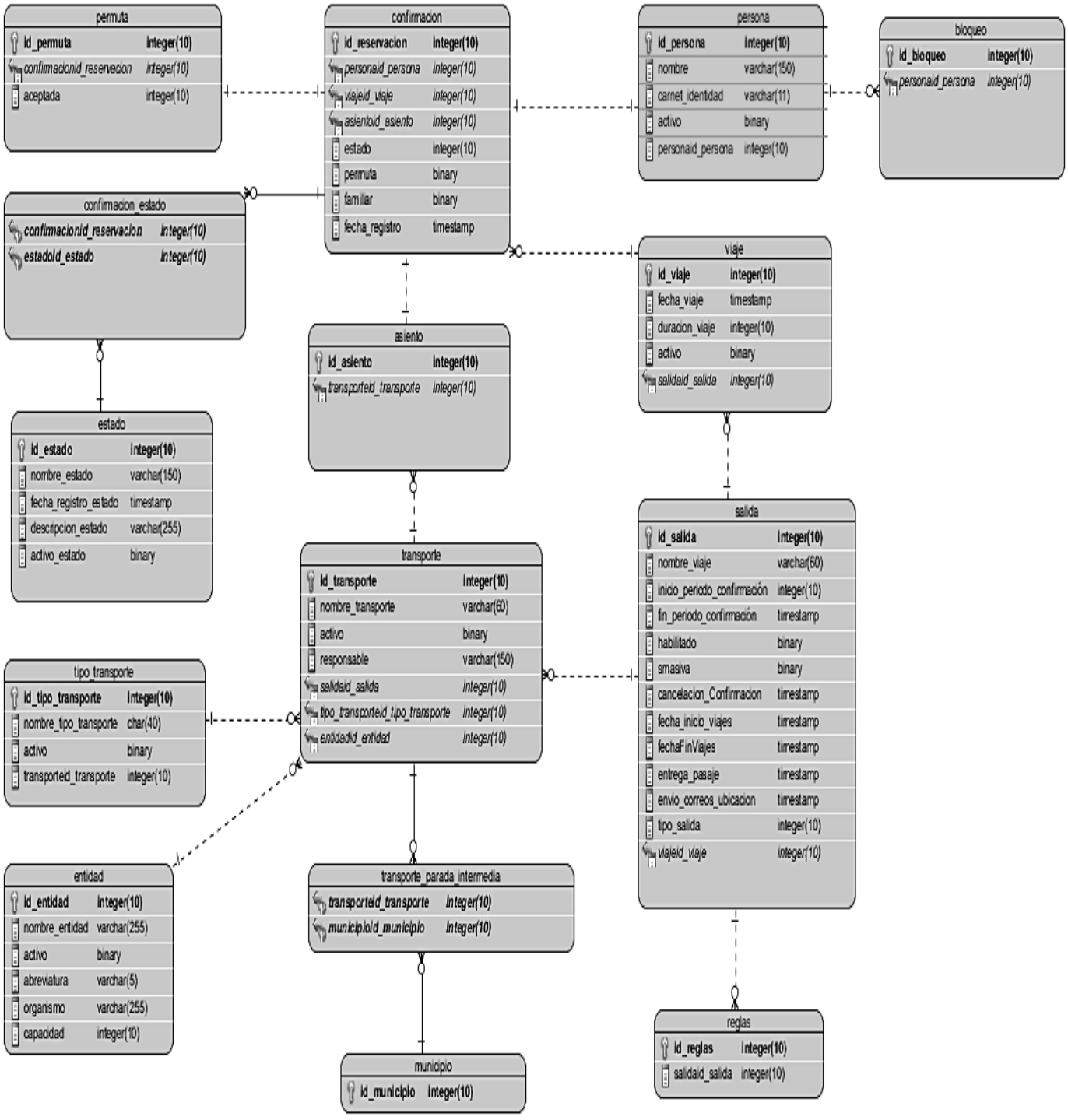


Figura 10 Modelo de Datos

2.8. Diagrama de despliegue

El diagrama de despliegue se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el *hardware* y el software que se despliega, estas relaciones son representadas por los protocolos de comunicación que se utilizan para acceder a cada uno. En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta:

Figura 11 Diagrama de Despliegue

El diagrama de despliegue representado muestra la siguiente distribución:

Cliente: dispositivo cliente capaz de conectarse al servidor de aplicaciones mediante el protocolo de comunicaciones HTTPS.

Servidor de aplicaciones web: computadora en que se encuentra el servidor web NGinx, este será el lugar en que se gestione todo el contenido de la aplicación. El mismo establecerá comunicación con los ordenadores clientes mediante protocolo HTTPS y con el servidor de base de datos por medio del protocolo TCP.

Servidor de base de datos: ordenador en que se encuentra el gestor de base de datos PostgreSQL capaz de mantener persistente la información generada y a utilizar.

HTTPS: protocolo de transferencia de hipertexto seguro, por sus siglas en inglés, *Hypertext Transfer Secure Protocol* (HTTPS), es un protocolo de red basado en HTTP por lo que está orientado a transacciones sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente y un servidor (43).

TCP/IP: base de Internet y sirve para enlazar computadoras que utilizan diferentes sistemas operativos. Familia de protocolos utilizada para la conexión entre el servidor web y el servidor donde se encuentra ubicada la base de datos (43).

USB: Universal Serial Bus (USB) (bus universal en serie) es un estándar industrial desarrollado a mediados de los años 1990 que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y periféricos y dispositivos electrónicos (44).

2.9. Conclusiones parciales

Luego de describir las características que debe cumplir el nuevo sistema basado en tecnologías libres y de realizar el análisis y diseño correspondiente, se concluye que los requisitos funcionales y no funcionales identificados a partir del proceso de obtención de los requisitos y los artefactos generados, constituyeron una guía fundamental para la construcción de la propuesta de solución. Con la realización del modelo físico de datos se hizo posible materializar la visión de la base de datos a utilizar. Además, con la utilización de la arquitectura Modelo-Vista-Controlador y el uso de los patrones de diseño y de base de datos propuestos, se garantizará una mayor organización, reutilización de funciones y código más legible. Por otro lado, la realización del modelo de despliegue ilustra la comunicación de los distintos componentes en los cuales se divide el sistema.

Capítulo 3: Construcción y validación de la propuesta de solución

Introducción

En el presente capítulo se definen los estándares de codificación que debe seguir el equipo de trabajo para desarrollar la aplicación, así como los métodos y técnicas para la realización de las pruebas, con el propósito de validar la solución. Se describe la implementación del software, fase donde finalmente se materializa el producto y cumple con los requisitos obtenidos al inicio de la investigación. El proceso de pruebas está dirigido a componentes del software, con el objetivo de medir el grado en que se cumplen los requisitos exigidos por el cliente y detectar la mayor cantidad de errores en el sistema para lograr su corrección.

3.1. Paradigmas de programación

Los paradigmas de programación indican las diversas formas que, a lo largo de la evolución de los lenguajes, han sido aceptadas como estilos para programar y para resolver los problemas por medio de una computadora. Indican un método para realizar un programa de cómputo y la manera en que se debe estructurar y organizar las tareas que se deben llevar a cabo en un programa (45).

3.1.1. Programación orientada a objetos (POO)

POO es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Los objetos son entidades que combinan estado (atributo), comportamiento (método) e identidad.

El marco de trabajo Symfony facilita la forma de programar en PHP y permite reutilizar el código con el paradigma de programación orientado a objetos que ya posee implementado. En él, cada objeto es responsable de inicializarse y destruirse de forma correcta, no existe la necesidad de llamar explícitamente al procedimiento de creación o de terminación debido a que el marco de trabajo se encarga de esto. Una de las principales características de esta programación es que proporciona la herencia lo que permite la reutilización del código, por lo que Symfony la utiliza entre los distintos tipos de clases como las modelos y las controladoras (46).

3.1.2. Programación dirigida por eventos (PDE)

La PDE es un paradigma de programación donde el flujo del programa está determinado por eventos o mensajes desde otros programas o hilos de ejecución. Es la base de lo que se denomina interfaz de usuario,

aunque puede emplearse también para desarrollar interfaces entre componentes de software o componentes de núcleos. Un programa dirigido por eventos debe haber sido creado en un lenguaje de programación orientado a objetos y cada objeto espera algún evento que realice el usuario sobre él (47). Symfony implementa este paradigma de programación. Es usada para el manejo de las interfaces de usuario y utiliza los principales eventos que provee el navegador para la interacción de sus componentes. Estos eventos pueden ser desencadenados por el usuario o por otros eventos en el sistema, permitiendo una mayor interoperabilidad usuario-sistema.

3.2. Estándares de codificación

Los estándares de codificación son reglas de codificación que permiten tener una programación homogénea, comprendiendo todos los aspectos de la generalización del código, pues la aplicación debe de estar implementada como si un único programador escribiera el código de una sola vez. La usabilidad de estos permite conservar el código fuente entendible y fácil de mantener, además de mejorar la forma en la que se programa (48). Para el desarrollo del sistema se utilizaron los estándares de codificación establecidos por la DIN con el propósito de estandarizar las nomenclaturas en su implementación y obtener un producto estable.

3.2.1. Indentación, llaves de apertura y cierre, y tamaño de líneas

El código debe usar cuatro espacios para la indentación en vez de usar el tabulado, esto minimiza problemas con otras herramientas de desarrollo.

Las líneas podrían tener 80 caracteres o menos evitando tener más de 120 caracteres. Las llaves de apertura deben ir en la siguiente línea y la llave de cierre debe ir en la siguiente línea después del cuerpo.

Los paréntesis en las estructuras de control, no deben usar espacios antes ni después. Se añade un solo espacio después de cada limitador de coma y alrededor de los operadores (==, &&, …).

Añadir una línea en blanco antes de una declaración de *return*, a no ser que esté dentro de una declaración como un grupo como *if*.

Usa llaves para indicar el control de la estructura sin tener en cuenta el número de declaraciones que el grupo pueda contener (48).

```
protected function getValores(NRegla $regla, $id = null){
    if ( $this->has($regla->getCaracteristica()->getService())) {
        $service = $this->get($regla->getCaracteristica()->getService());
        $method = $regla->getCaracteristica()->getMethodService();
        if(is_callable(array($service, $method))){
            return new Response(json_encode($service->$method($regla->getCaracteristica(), $id)));
        }
    }
    return self::showMessage(false, $this->get('translator')->trans('app.msg.servicio_no_definido'));
}
```

Figura 12 Indentación, llaves de apertura y cierre, y tamaño de líneas

3.2.2. Convención de nomenclatura

Variables: Se rigen por la nomenclatura *camelCase*[13] para declarar. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Se muestra en el método cronogramaAction(Request $request) un ejemplo donde se emplean las variables:

```
public function cronogramaAction(Request $request){
    $tipo = $this->getParameter('nucleo.nomencladores.tipo_evento_negocio.trans_masiva');
    $events = $this->get('nucleo_calendar.manager')->obtenerEventos($request->get('start'), $request->get('end'), $tipo);
    return new Response(json_encode($events));
}
```

Figura 13 Convención de nomenclatura: variable: camelCase

Clases: Se rigen por la nomenclatura StudlyCaps[14]. El patrón que sigue para declarar las clases es que siempre comienzan con mayúscula y en caso de nombre compuesto cada palabra comienza en mayúscula, sin espacios o guion bajo, como se muestra a continuación:

```
class TMasivasReportesExtension extends Extension
{
```

Figura 14 Convención de nomenclatura. Clase nombre compuesto. StudlyCaps

[13] camelCase: Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en camelCase se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.

[14] Studlycaps es una forma de notación de código en el que la capitalización de letras varía según un patrón, o arbitrariamente, por lo general también omitiendo espacios entre las palabras y, a menudo omitiendo algunas letras.

```
class Configuration implements ConfigurationInterface
{
```

Figura 15 Convención de nomenclatura. Clase nombre simple. StudlyCaps

Funciones: Se rigen por la nomenclatura *camelCase*. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa (ver figura 16).

Ficheros: Los ficheros deben usar solo las etiquetas de apertura php <?php, el formato de codificación de caracteres UTF-8.

```
public function menuLateral(){

    $itemsMenu = array();
    $itemsMenu [] = array(
        /** @Ignore */'label' => $this->container->get('translator')->trans('clave aqui 0'),
        'route' => 'trans_masivas_configuracion_homepage',
        'params' => array()
    );
    $itemsMenu [] = array(
        /** @Ignore */'label' => $this->container->get('translator')->trans('clave aqui 1'),
        'route' => 'trans_masivas_confirmacion_homepage',
        'params' => array()
    );
    $itemsMenu [] = array(
        /** @Ignore */'label' => $this->container->get('translator')->trans('clave aqui 1'),
        'route' => 'trans_masivas_gestion_homepage',
        'params' => array()
    );


    return $this->container->get('base.menu_route_builder')->filterMenu($itemsMenu);
}
```

Figura 16 Convención de nomenclatura: función: camelCase

3.2.3. Estructuras de control

Las estructuras de control incluyen *if, for, for each, while, switch*, entre estas estructuras y los paréntesis que encierran la condición debe de existir un espacio. Es recomendable utilizar en cualquier caso llaves de apertura y cierre al comienzo de una nueva línea, incluso en situaciones en las que técnicamente son opcionales.

```
if ( $this->has($regla->getCaracteristica()->getService())) {
    $service = $this->get($regla->getCaracteristica()->getService());
    $method = $regla->getCaracteristica()->getMethodService();
    if(is_callable(array($service, $method))){
        return new Response(json_encode($service->$method($regla->getCaracteristica(), $id)));
    }
```

Figura 17 Estructuras de control

3.2.4. Documentación

Todos los archivos deben tener su documentación asociada. Se debe de cumplir con el siguiente bloque al principio de cada clase, como se muestra a continuación:

```
<?php
/**
 * Created by NetBeans.
 * User: Ramon
 * Date: 25/1/2016
 * Time: 1:48 PM
 */

namespace UCI\DIN\Trans\Masivas\GestionBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use UCI\DIN\Nucleo\BaseBundle\Annotation\Eventos;

/**
 * Class DSalida
 * @package UCI\DIN\Trans\Masivas\GestionBundle\Entity
 * @ORM\Table(name="sq_transportacion.tb_dsalida")
 * @ORM\Entity(repositoryClass="UCI\DIN\Trans\Masivas\GestionBundle\Entity\DSalidaRepository")
 */
```

Figura 18 Ejemplo de archivo documentado

3.3. Validación de requisitos

La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proyecto y el producto (42).

3.3.1. Criterios para validar los requisitos

Para validar los requisitos del sistema según Pressman, se pueden chequear mediante un cuestionario guiado por un conjunto de interrogantes, con el objetivo de descubrir la mayor cantidad de errores posibles (34).

A continuación, se muestran las preguntas utilizadas, algunas fueron agregadas a las planteadas por Pressman por ser consideradas necesarias para la validación.

Interrogantes para la validación de requisitos:

¿Está el requisito claramente definido?

¿Puede interpretarse mal?

¿Está identificado el origen del requisito (por ejemplo: persona, norma, documento)?

¿El planteamiento final del requisito ha sido contrastado con la fuente original?

¿El requisito está delimitado en términos cuantitativos?

¿Qué otros requisitos hacen referencia al requisito estudiado?

¿El requisito incumple alguna restricción definida?

¿El requisito es verificable? Si es así, ¿se pueden efectuar pruebas para verificar el requisito?

¿Se puede seguir el requisito en el modelo del sistema que se ha desarrollado?

¿Está el requisito asociado con los rendimientos del sistema o con su comportamiento?

¿El requisito está implícitamente definido?

¿El requisito es modificable?

¿El requisito está completo?

¿El requisito puede ser implementado?

¿El requisito puede ser probado?

¿El resultado de la evaluación de impacto es positivo?

Resultado de aplicar los criterios de validación

Luego de aplicar el conjunto de interrogantes para validar los requisitos definidos para el desarrollo del sistema, se obtuvo el 100 % de aprobación por parte del cliente.

3.3.2. Técnicas de validación de requisitos

Con el objetivo de obtener una mayor calidad y demostrar que los requisitos definidos realmente describen el sistema que el cliente necesita; se utilizaron las técnicas para la validación de requisitos siguientes:

Prototipado de interfaz: permite al cliente entender fácilmente la propuesta de solución, al brindar la representación aproximada de la interfaz de usuario que tendrá el sistema. Existen dos tipos principales de prototipo de interfaz:

Desechables: se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en software.

Evolutivos: una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final. El tipo utilizado finalmente para la propuesta de solución fue Evolutivos pues de esta forma se reutiliza el prototipo diseñado en todo el proceso de desarrollo del sistema.

Generación de casos de prueba: se realizaron los diseños de casos de pruebas para cada uno de los requisitos obtenidos, permitiendo verificar que todos se pudieran probar y determinar en la medida de la complejidad del diseño de caso de prueba, identificando requisitos que deberían ser reconsiderados y cuáles pueden ser los más difíciles de implementar.

Resultado de aplicar las técnicas de validación

Como resultado de este proceso se identificaron inconsistencias en las especificaciones tales como la falta de concordancia entre la complejidad de la especificación y la registrada en el documento de Evaluación de requisitos. Descripciones de requisitos poco detalladas o ambiguas. Se encontraron además numerosos errores ortográficos.

3.4. Pruebas realizadas a la solución

Para evaluar la calidad del sistema que se está desarrollando y verificar el cumplimiento de los objetivos trazados, se aplicaron un conjunto de pruebas definidas por Pressman en su libro de Ingeniería del software

“Un enfoque práctico”, en su quinta edición. A continuación, se muestra la estrategia de prueba diseñada para aplicar en la solución desarrollada:

Tabla 6 Pruebas realizadas

Prueba	Método	Técnica
Prueba de unidad	Caja blanca	Camino básico
Prueba de integración	Caja negra	Incremental
Prueba de sistema (carga y *stress*)	Caja negra	Automático
Prueba de validación	Caja negra	Partición de equivalencia
Prueba de aceptación	Caja negra	Alfa

3.4.1. Prueba de unidad

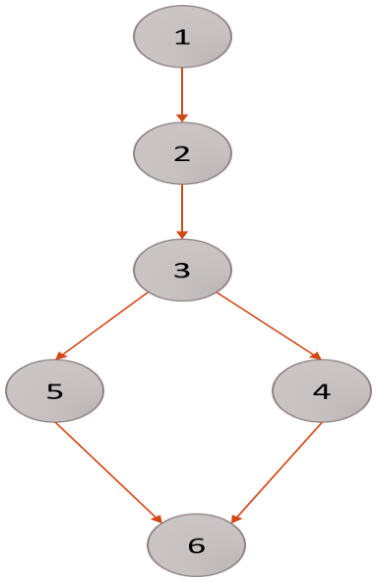
Las pruebas de unidad tienen como objetivo verificar la unidad más pequeña del diseño del software. Estas pruebas se concentran en la lógica del procesamiento interno y en las estructuras de datos tales como: código fuente, archivos binarios, archivos de datos, entre otros. Este tipo de prueba se puede aplicar en paralelo a varios componentes.

Las pruebas de unidad se realizan mediante el **método de caja blanca o estructural**, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. La técnica de prueba de caja blanca utilizada en la investigación es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (34).

Complejidad ciclomática

La Complejidad Ciclomática se realiza a todos los métodos o algoritmos de un sistema informático. A continuación, se presenta la prueba realizada al método guardarNuevoAction() ya que posee importancia en el contexto de la investigación, el cual posibilita que se cree un nuevo viaje.

Tabla 7 Prueba de unidad

<table>
<tr><th>Fórmula 1</th><th>Fórmula 2</th><th>Fórmula 3</th><td rowspan="4"> 
</td></tr>
<tr><td>V (G) = (A - N) + 2

V (G)=6-6+2=2</td><td>V (G)=P+ 1

V (G)=1+ 1 =1</td><td>V (G) = R

V (G) = 2</td></tr>
<tr><td colspan="3">A: es la cantidad de aristas.
N: la cantidad de nodos.
P: es el número de nodos predicado contenidos en el grafo de flujo G
R: representa la cantidad de regiones en el grafo</td></tr>
<tr><td colspan="3">Complejidad Ciclomática
Como se puede observar, después de aplicadas las fórmulas 1, 2 y 3 a la funcionalidad guardarNuevoAction, posee una complejidad ciclomática igual a 2, lo cual demuestra que las tres fórmulas son efectivas.</td></tr>
<tr><td colspan="4">Prueba de Camino básico
Una vez calculada la complejidad ciclomática se define como límite superior 2, lo que indica que hay que realizarle al código dos pruebas, para garantizar que este se ejecute completamente al menos una vez. La funcionalidad crear viaje posee poco riesgo debido a que el resultado arrojado por la métrica pertenece al intervalo entre 1 -10.

El total de caminos independientes establecidos fue de 2 y a continuación se muestran:

• Camino 1: 1,2,3,4,6.
El camino 1 inicia con el primer bloque de instrucciones seguido de una condicional y finaliza mostrando un mensaje de confirmación y la lista de viajes creados.
• Camino 2: 1,2,3,5,6.
El camino 2 inicia con el primer bloque de instrucciones seguido de una condicional y finaliza mostrando un mensaje de error.</td></tr>
</table>

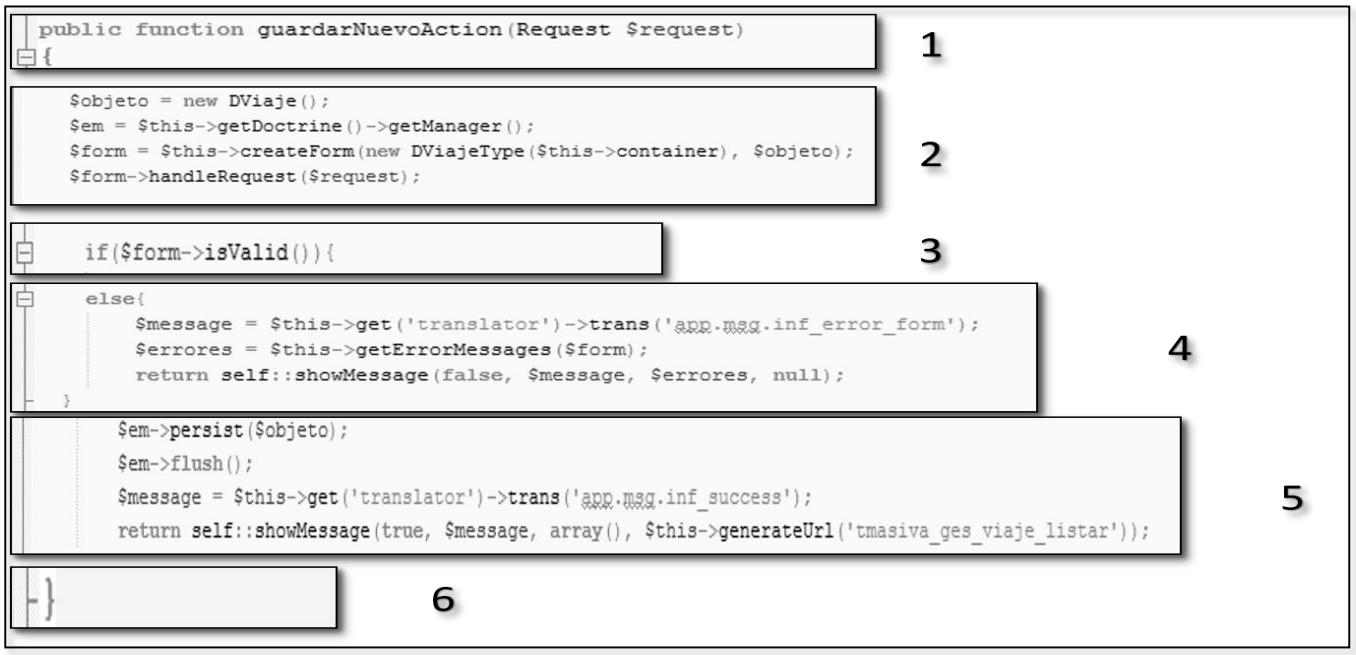


Figura 19 Ejemplo del código utilizado para calcular la complejidad ciclomática

3.4.2. Prueba de integración

La prueba de integración es una técnica sistemática para construir la estructura de un programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. Existen dos tipos de integración: no incremental e incremental. En el primer caso se combinan todos los módulos y se prueba el programa en su conjunto, como es lógico pensar el resultado puede ser caótico con un gran número de fallos y la consiguiente dificultad para identificar el módulo que los provocó. Por su parte, en la integración incremental el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir (34). Por esta razón se escogió el enfoque incremental para la realización de las pruebas de integración de la solución.

Resultados de la prueba de integración

Para la validación de la solución se probó la integración de la misma con el resto de los sistemas del Sistema de Gestión del Ciudadano (SGC). Los resultados de las pruebas que se describen están relacionados con los componentes de Seguridad y Base, probándose de este último la integración con el componente Personal. La siguiente tabla muestra el caso de prueba de integración realizado al componente de Personal.

Luego de haber aplicado las pruebas de integración, no se detectaron errores asociados a la interrelación entre el Sistema Transportación con los componentes del SGC.

Tabla 8 Resultados de la prueba de integración

Caso de prueba: Integración de Sistema de Transportación	
Componente al que se integra	Personal.
Condiciones de ejecución	En el componente Personal se debe haber introducido la información en la base de datos central y que en la misma exista conexión.
Descripción de la prueba	Verificar que el sistema Transportación es capaz de comunicarse con la base de datos central y hacer uso de la información del personal involucrado en el proceso de transportación.
Entradas/Pasos de ejecución	Desde el sistema Transportación se gestiona la información del personal involucrado en el proceso directamente con el componente Personal.
Resultado esperado	Se gestiona la información del personal involucrado.
Evaluación	Satisfactoria.

3.4.3. Prueba de validación

Tras la culminación de la prueba de integración, el software está completamente ensamblado como un paquete, seguidamente se puede comenzar la prueba de validación. La validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente. Una vez que se procede con cada prueba de validación, puede darse una de las dos condiciones siguientes: -1 - las características de funcionamiento o de rendimiento estén de acuerdo con las especificaciones y son aceptables; o -2- se descubre una desviación de las especificaciones y se crea una lista de deficiencias (49).

Para el desarrollo de las pruebas de validación se aplicó el **método de caja negra** con la **técnica de partición de equivalencia**, donde se demuestran la conformidad de los requisitos.

Los **métodos de caja negra**, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a base de datos externas, errores de rendimiento y errores de inicialización y de terminación (50).

La técnica de **partición de equivalencia** consiste en ejecutar el flujo básico de las funcionalidades utilizando datos válidos e inválidos, donde se generaron los artefactos “Diseño de casos de pruebas”. Por cada requisito funcional del sistema se generó un documento donde se recogen todos los datos necesarios para probar la interfaz (50).

Se ejecutaron un total de 48 casos de pruebas, a los cuales se le realizaron 3 iteraciones, hasta garantizar la eliminación de las no conformidades. A continuación, se muestra un ejemplo del caso de prueba (CP) para *SGC_TRAN_DCP_N_Crear_Regla* y una tabla con la información de los resultados obtenidos de las pruebas en cada iteración:

Tabla 9 CP Crear regla

Escenario	**Descripción**	**V1-Nombre**	**V2-Descripción**	**V3-Activo**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Insertar datos correctamente	Mediante este escenario se registra en el sistema una calidad.	V Prueba	V Prueba	V Seleccionado	Muestra el mensaje “El elemento ha sido creado satisfactoriamente”. El sistema muestra y actualiza el listado de tipos de acuerdo.	El usuario una vez autenticado en el Sistema de Gestión Ciudadano selecciona el Sistema de Transportación. El usuario selecciona el módulo Nacionales. El sistema muestra todas las agrupaciones de contenido de este módulo. El usuario selecciona la agrupación Configuración. El sistema muestra las funcionalidades de esta agrupación de contenido. El usuario escoge la opción Reglas. El sistema muestra el listado de las reglas registradas hasta la fecha. El usuario selecciona la opción Crear en el área de íconos flotantes. El sistema muestra un formulario. El usuario introduce los datos y escoge la opción Aceptar. El sistema muestra un mensaje de información "El elemento ha sido creado satisfactoriamente".

EC 1.2 Registrar con datos obligatorios vacíos	Mediante este escenario no se introducen todos los datos necesarios para registrar una regla.	I Vacío	NA	NA	El sistema muestra en rojo el texto "Campo requerido" en los campos que deben ser llenados de forma obligatoria.	El usuario una vez autenticado en el Sistema de Gestión Ciudadano selecciona el Sistema de Transportación. El usuario selecciona el módulo Nacionales. El sistema muestra todas las agrupaciones de contenido de este módulo. El usuario selecciona la agrupación Configuración. El sistema muestra las funcionalidades de esta agrupación de contenido. El usuario escoge la opción Reglas. El sistema muestra el listado de las reglas registradas hasta la fecha. El usuario selecciona la opción Crear en el área de íconos flotantes. El sistema muestra un formulario. El usuario introduce los datos y escoge la opción Aceptar. El sistema muestra en rojo el texto "Campo requerido" en los campos que deben ser llenados de forma obligatoria.
EC 1.3 Registrar con datos incorrectos	Mediante este escenario se introducen datos incorrectos al registrar una calidad.	I d	NA D	NA	El sistema indica en rojo el texto "Entre al menos 2 caracteres" en el campo correspondiente.	El usuario una vez autenticado en el Sistema de Gestión Ciudadano selecciona el Sistema de Transportación. El usuario selecciona el módulo Nacionales. El sistema muestra todas las agrupaciones de contenido de este módulo. El usuario selecciona la agrupación Configuración. El sistema muestra las funcionalidades de esta agrupación de contenido. El usuario escoge la opción Reglas. El sistema muestra el listado de las reglas registradas hasta la fecha. El usuario selecciona la opción Crear en el área de íconos flotantes. El sistema muestra un formulario. El usuario introduce los datos. El sistema indica en rojo texto las restricciones en el campo correspondiente, en caso de que el usuario introduzca de forma incorrecta el valor.
		I @~$	NA @~$	NA	El sistema indica en rojo los textos "Entre solo letras, números, y espacio o guion bajo entre palabras “o “Entre solo letras, números, espacios, guiones, paréntesis, apóstrofe o guiones bajos” en el campo correspondiente.	
		I	I	NA	El sistema indica en rojo	

		<Se introduce n más de 50 caracteres >	<Se introduce n más de 1000 caracteres >		los textos "No más de 50 caracteres" y "No más de 1000 caracteres" en los campos correspondientes.	
		NA	I <Se introduce n más de 30 caracteres consecutivos>	NA	El sistema indica en rojo texto "Ha excedido el número de letras permitidas para una palabra." en el campo correspondiente.	
EC 1.4 Cancelar operación	Mediante este escenario se cancela la acción de registrar una calidad.	NA	NA	NA	El sistema muestra un mensaje de confirmación: "¿Está seguro de realizar la acción?". En caso que el usuario seleccione Aceptar el sistema no registra la calidad y muestra el listado de las calidades registradas.	El usuario una vez autenticado en el Sistema de Gestión Ciudadano selecciona el Sistema de Transportación. El usuario selecciona el módulo Nacionales. El sistema muestra todas las agrupaciones de contenido de este módulo. El usuario selecciona la agrupación Configuración. El sistema muestra las funcionalidades de esta agrupación de contenido. El usuario escoge la opción Reglas. El usuario selecciona la opción Crear en el área de íconos flotantes. El sistema muestra un formulario. El usuario introduce los datos y escoge la opción Cancelar. El sistema muestra un mensaje de confirmación "¿Está seguro de realizar la acción?". En caso que el usuario seleccione Aceptar el sistema no registra la calidad y muestra el listado de las calidades registradas.

Tabla 10 Descripción de las variables del CP Crear regla

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Nombre de la regla	Campo de texto	No	Admite solo letras, números, y espacio o guion bajo entre palabras y requiere de 2 a 250 caracteres.
2	Descripción de la regla	Campo de texto	Si	Requiere como mínimo 2 caracteres y máximo 1000. No permite más de 30 caracteres consecutivos. Admite solo letras, números, espacios, guiones, paréntesis, apóstrofe o guiones bajos.
3	Activo	Casilla de verificación	Si	Activa o desactiva la regla.

Tabla 11 Iteraciones de los casos de prueba

Iteración	Total de no conformidades	Asociadas a
1	30	Errores de interfaz, validación y ortografía.
2	9	Errores de interfaz, validación
3	0	No se detectaron errores

3.4.4. Pruebas de sistema

Las pruebas de sistema están constituidas por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiada (34). Entre las pruebas de sistema que se ejecutan están las pruebas de rendimiento y resistencia.

Pruebas de rendimiento: están diseñadas para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. Lo que posibilita determinar cuán rápida es la respuesta de sistema ante un conjunto de peticiones concurrentes (34).

Pruebas de resistencia: están diseñadas para enfrentar a los programas con situaciones anormales. La prueba de resistencia ejecuta un sistema de forma que demande recursos en cantidad o volúmenes anormales (34).

Resultados de las pruebas de rendimiento y resistencia

Una prueba de rendimiento se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. El sistema se prueba con un número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la prueba. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Para llevar a cabo las pruebas de rendimiento y resistencia se utilizó la herramienta JMeter 2.12. La prueba consistió en realizar a una funcionalidad tres pruebas de 25, 50 y 75 hilos, los cuales simulan 25, 50 y 75 accesos de usuarios respectivamente. Se definió una lista de enlaces a los que se simuló el acceso aleatorio y a partir de ahí, se recolectaron los datos necesarios para su interpretación.

Para un mejor entendimiento de los datos que se verán a continuación, se explica cada parámetro que compone la tabla.

- **#Muestras:** cantidad de hilos utilizados para la URL.
- **Media:** tiempo promedio en milisegundos para un conjunto de resultados.
- **Min:** tiempo mínimo que demora un hilo en acceder a una página.
- **Max:** tiempo máximo que demora un hilo en acceder a una página.
- **Pet/seg:** hace referencia al número de peticiones que el servidor puede procesar en un segundo.
- **Kb/sec:** rendimiento medido en Kilobytes por segundo.

A continuación, se muestran en la Tabla 12 los resultados obtenidos con la herramienta:

Tabla 12 Resultados rendimiento y resistencia

Aplicado a	Cantidad de Hilos	Tiempo de ejecución (ms)			Rendimiento		
		Mín.	**Máx.**	**Media**	**% Error**	**Pet/seg**	**Kb/seg**
Crear salida	25	309	632	475	0.00%	17.5 seg	552.7
	50	185	1812	1256	0.00%	17.7 seg	560.1
	75	462	3095	1905	0.00%	18.3 seg	579.9

3.5. Conclusiones parciales

En este capítulo se detallaron los estándares de diseño y codificación empleados en la implementación del Componente para la Gestión de las Transportaciones Nacionales del Sistema de Gestión del Ciudadano de la Universidad de las Ciencias Informáticas permitiendo una mejor organización y comprensión del código. Con los resultados de la validación de los requisitos identificados se logró obtener un listado de requisitos correctamente redactados y descritos. La descripción de las pruebas utilizadas para asegurar la calidad del software, permitió obtener resultados satisfactorios, asegurando que el sistema implementado no contiene errores y tiene la aceptación requerida.

Conclusiones Generales

Al terminar la presente investigación, se obtuvieron resultados que permiten arribar a las siguientes conclusiones:

- El análisis de los principios teóricos de la investigación, enfocados a los sistemas de gestión y de reservaciones, permitió tener un mejor entendimiento de los procesos sustantivos de reservación de transporte.
- El estudio del sistema actualmente utilizado en la universidad permitió detectar y analizar sus deficiencias, posibilitando la realización del diseño e implementación de esta nueva propuesta, adaptada a las características del entorno actual de la institución.
- El empleo de un proceso de desarrollo con enfoque ágil, favoreció el trabajo en equipo de conjunto con el cliente y trajo como resultado la obtención de un sistema que cumple con los requerimientos definidos.
- El diseño y ejecución de los casos de prueba al software, permitieron identificar errores de implementación en la aplicación, que fueron corregidos posteriormente y entregarle al cliente un producto funcional, dándole así cumplimiento al objetivo general de esta investigación.

Recomendaciones

Para garantizar el perfeccionamiento progresivo de la solución propuesta se proponen las siguientes recomendaciones que tributarán a un producto de mejor calidad. Las funcionalidades que se recomiendan son:

- Implementar un algoritmo para nombrar automáticamente las salidas creadas en el sistema, tomando como referencia los datos con que es creada la salida.
- Tomar los nombres de los responsables de los transportes y viajes desde la base de datos de viajeros del sistema.

Bibliografía Referenciada

1. **Rodríguez González, Isied y Miras González, Luis Andrés.** Gestión de reservaciones de las transportaciones nacionales de la Universidad de las Ciencias informáticas. La Habana : s.n., junio de 2012.

2. **Garmendia López, Franklin Yannier y Rodríguez Medina, Yanet .** Análisis y Diseño de un Sistema de Transportación Nacional. La Habana : s.n., 2008.

3. **Titaeva, Irina, Mendoza, Luis E y Perez, Maria.** Modelos de Integración de Sistemas en Empresas Venezolanas: Estudios de Caso. Caracas : Universidad Simón Bolivar, 2014.

4. **Hernández Sampier, Roberto.** *Metodología de la investigación.* La Habana : Félix Varela, 2008. Vol. I.

5. **Luján Mora, Sergio.** *Programación de aplicaciones web: historia, principios básicos y clientes web.* Alicante : Editorial Club Universitario, 2002. ISBN: 84-8454-206-8.

6. **Cobo, Angel, y otros.** *PHP y MySQL. Tecnologías para el desarrollo de aplicaciones web.* Madrid : Rústica-Hilo, 2005. ISBN: 84-7978-706-6.

7. **Peña Ayala, Alejandro.** *Ingeniería de software: Guía para crear sistemas de información.* México, D.F. : IPN, 2009. ISBN: 970-94797-0-9.

8. **Duany Dangel, Armando.** *Gestión del Conocimiento: Una Herramienta Esencial para el Diseño de Sistemas de Información.* La Habana : Centro de Estudio de Desarrollo Agrario y Rural, 2009.

9. **BSI Group.** Inicio| BSI Group. [En línea] BSI Group. http://www.bsigroup.com/es-MX/.

10. **Naranjo, D. L. N.** *Aplicaciones Web Integrales. Breves Apuntes.* Holguín : Entre Líneas, 2012. ISBN 1818-3018.

11. **Baró Montenegro, Yohana y Hernández Rodríguez, Reynaldo.** Sistema de Reservación de Pase Masivo. La Habana : s.n., 2007.

12. **Empresa Española Edreams.** Edreams. *Edreams.* [En línea] 2016. http://www.edreams.es/edreams/espanol/ayuda/external0.jhtml.

13. **Romero, J J.** JR Software. *JR Software.* [En línea] 2016. http://Www.softwarejr.com.ar.

14. **Russian Airlines.** Aeroflot. *Aeroflot.* [En línea] 2015. https://www.aeroflot.ru/cms/es.

15. **Aviacion, C.D.** Cubana. *Cubana.* [En línea] 2016. http://www.cubana.cu/home/.

16. **Viazul & Gold Black Investments. SA .** Viazul. [En línea] 2016. http://www.viazul.com. CIF:A82971953.

17. **Object Management Group, Inc.** BPMN. *BPMN.* [En línea] 2016. http://www.bpmn.org/.

18. **DB Consulting Inc.** PostgreSQL. *PostgreSQL.* [En línea] http://www.pgadmin.org.

19. **W3C.** W3C. [En línea] 2013. http://www.w3.org/html/.

20. **The PHP Group.** The PHP Group. [En línea] 2013. http://us.php.net/manual/en/preface.php...

21. **Sánchez, Jorge.** *Java Script manual de referencia.* Ciudad de Mexico, D.F. : ECMA, 2003.

22. **Montalvo Melián, Marlene.** Biblioteca Virtual de las Ciencias en Cuba. [En línea] 2008. http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASH01 04/f016d031.dir/doc.pdf..

23. **Hong Kong Institute of Vocational Education.** Visual Paradigm. [En línea] 2016. http://www.visual-paradigm.com/aboutus/newsreleases/vpsuite50.jsp.

24. **Oracle Corporation.** NetBeans. [En línea] 2016. https://netbeans.org/features/index.html.

25. **PostgreSQL Global Development Group.** PGAdmin. [En línea] 2016. http://www.pgadmin.org/.

26. **Evolus.** Pencil Project. [En línea] 2016. http://pencil.evolus.vn/Features.html.

27. **Corona, Stephen.** *Nginx. A practical guide to high performance.* Sebastopol : O'Reilly, 2015.

28. **Foundation-Tika, The Apache Software.** Apache Tika. *Apache Tika.* [En línea] Foundation-Tika, The Apache Software, 2014. url: http://tika.apache.org.

29. **Potencier, Fabien y Zaninotto, Francois.** *Symfony la guia definitiva.* New York : Sensio SA, 2008. ISBN: 1-59059-786-9.

30. **Sánchez Méndez, Alelí.** *Proceso de desarrollo de software: Metodología DAC.* La Habana : s.n., 2013.

31. **Sparks, Geoffrey.** *Introducción al modelado de sistemas de software.* 2015.

32. **Sommerville, Ian.** *Ingeniería de Software.* Séptima edición . s.l. : Addison-Wesley, 2008. Vol. II. ISBN:978-0-321-31379-9.

33. **Pressman, Roger S.** *Ingeniería de software: Un enfoque práctico.* VI edición. Madrid : s.n., 2005.

34. —. *Ingeniería del Software. Un enfoque práctico.* VI Edición. Connecticut : McGraw Hill, 2013.

35. **Gamma, Erich.** *Design Patterns. Elements of Reusable Object-Oriented Software.* Sebastopol : Addison Wesley, 2009.

36. **Zaninotto, F. y Potencier, F.** Librosweb.es. *Symfony 1.1, la guia definitiva.* [En línea] 2009. http://librosweb.es/libro/symfony_1_1/.

37. **Gamma, E, Helm, R y Johnson, R.** *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison Wesley, 2000.

38. **Grosso, Andres.** *Practicas de software: Patrones GRASP.* 2011.

39. **EcuRed.** EcuRed. [En línea] 2016. http://www.ecured.cu/Patrones_en_Symfony.

40. **Craig, Larman.** *UML y patrones:una introducción al análisis y diseño orientado a objetos y al proceso unificado.* II Edición. La Habana : Prentice Hall, 3015. 565543455.

41. **Ecured.** EcuRed. [En línea] EcuRed, 2016. http://www.ecured.cu/Patrones_de_dise%C3%B1o_de_bases_de_datos.

42. **Alvarado, Jose Manuel.** *Bases de datos. Unidad 2. Modelo de datos.* Estelí : Universidad Nacional de Ingenieria, 2010.

43. **Stallings, William.** *Comunicaciones y redes de computadoras.* VI Edición. Barcelona : Prentice Hall, 2000.

44. **Boston Globe .** Boston Globe Online . *USB deserves more support.* [En línea] 12 de 12 de 2011. http://www.simson.net.

45. **Instituto Tecnológico de Celaya.** Paradigmas de programación. Celaya : s.n., 2003. ISBN: 978-3-16-148410-0.

46. **Wang, Paul S.** *Java con Programación Orientada a Objetos. .* 2001.

47. **Pavón, Santiago.** *Programación Orientada a Eventos. .* 2012.

48. **Garay, Joseba.** Convenciones en código Symfony2. s.l. : Colorlib, 2012.

49. **Pressman, Roger S.** *Ingenieria del software. Capitulo 18 Estrategia de prueba de software. Prueba de validación.* 2013.

50. —. *Ingenieria del software. Capitulo 17 Estrategia de prueba de software. Prueba de caja negra.* 2013.

51. **Shaw, Mary y Garlan, David.** *Software Architecture: Perspectives on an Emerging Discipline.* 1996.

52. **Barroso, J.** *Propuesta de pautas para el diseño de un Sistema de Gestión de Información en la empresa ECIMETAL.* La Habana : Departamento de Bibliotecología y Ciencia de la Información, 2009.

53. **Castro Carrillo, Yaremí.** *Integración de la Plataforma para la Gestión de Eventos Científicos con el Sistema de Gestión Universitaria y Repositorio Institucional de la Universidad de las Ciencias Informáticas.* La Habana : UCI, 2012.

54. **RDNS.** *Informe central: Integración de Sistemas de Seguridad.* s.l. : RDNS, 2012.

55. **Libros Web.** *Introducción a CSS.* 2014.

56. **Obe, Regina y Hsu, Leo.** *Up and Running.* Sebastopol, CA : O'Reilly, 2012. ISBN: 978-1-449-32633-3.

57. **W3C.** *HTML 4 Dominar el código fuente.* Barcelona : ENI, 2006. ISBN: 2-7460-3361-5.

58. **The PHP Group.** *PHP Reference: Beginner to Intermediate PHP5.* Colorado : s.n., 2008. ISBN: 978-1-4357-1590-5.

59. **McPeak, Jeremy y Wilton, Paul.** *Beginning JavaScript .* Indianapolis : Wrox, 2015. ISBN: 978-1-118-90333-9.

60. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* Harlow : Addison-Wesley, 2010.

61. **García Lima Lorena, Montero Reyes Isidro.** *Sistema para el registro y control de los procesos de la vida interna del PCC en la UCI.* La Habana : s.n., 2015.

62. **PRESSMAN.** *Ingeniería de software: Un enfoque práctico.* Quinta edición. Madrid : s.n., 2001.

63. **Bieberstein, Norbert et al.** *Service-Oriented Architecture Compass.* s.l. : Pearson, 2006. ISBN 0-13-187002-5.

64. **openSUSE Project.** OpenSuse. [En línea] 2016. https://es.opensuse.org/Apache.

65. **sensagent Corporation.** Diccionario Sensagent. [En línea] 2016. http://diccionario.sensagent.com/diccionario/es-es/.

66. **Buenas Tareas.** Buenas Tareas. [En línea] 2016. http://www.buenastareas.com/ensayos/Sistema-De-Reservaciones/1131441.html.

67. **Synergix.** Tecnología y Sinergix. [En línea] 2011. http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/..

Glosario de Términos

- **Sistema de gestión de información**: Conjunto de políticas y normas relacionadas entre sí, que se establecen para el acceso y tratamiento de los recursos de información.
- **Sistema de reservación:** Los sistemas de reservaciones son un medio para acceder a los servicios de una empresa o institución en general, durante un tiempo determinado con antelación a que este sea brindado. Las reservaciones son de gran importancia, pues reducen el riesgo de que el cliente no reciba el servicio y disminuyen el tiempo que este debe esperar para tener acceso al mismo
- **Reservación de transporte:** Es el acto de apartar un lugar en un medio de transporte, el cual va a ser usado en viajes para la transportación de personas hacia un destino específico. Este servicio es ofrecido en la Universidad de las Ciencias Informáticas destinado al personal que la compone, que pueden ser trabajadores o estudiantes, debido a que estos proceden de todas las provincias del país.
- **Reservación masiva:** Las reservaciones de carácter masivo, como su nombre lo indica, están destinadas para viajes masivos en los cuales son transportadas grandes cantidades de personas. Estas son efectuadas por todo el personal de la institución, ya sean estudiantes o trabajadores, hacia y desde sus provincias de origen. Las mismas son realizadas en los meses de fin e inicio de curso y en fin e inicio de año.
- **Componente de software** es un elemento de un sistema de software que ofrece un conjunto de servicios, o funcionalidades, a través de interfaces definidas.
- **Integración:** implica integrar aplicaciones y fuentes de datos empresariales tal que puedan fácilmente compartir procesos de negocio e información.
- **Identidad Marcaria:** Son un conjunto de pautas a tener en cuenta para la realización de los productos que serán desarrollados en los diferentes centros de la Universidad de las Ciencias Informáticas, con el objetivo de estandarizar el diseño para garantizar una coherencia entre estos y lograr establecer una política de identidad corporativa dentro de la Universidad en cuanto a sus aplicaciones. Los códigos cromáticos, tipográficos, iconografía y otros recursos gráficos que se emplean para el desarrollo de estas interfaces, se mantienen constantes en cada una de las líneas temáticas, siendo el color el elemento que marca la diferencia entre ellas.

Anexos

Anexo 1: Encuesta de Satisfacción a usuarios finales sobre el uso del Sistema para la Confirmación de la Transportación Masiva de la Universidad de las Ciencias Informáticas.

1. ¿Logra usted entender con facilidad la información brindada por la aplicación referente a las fechas de las transportaciones?
2. ¿Tiene la aplicación una interfaz amigable?
3. ¿Ha necesitado los servicios de la aplicación en algún momento y esta no ha estado disponible cuando debería estarlo?
4. ¿La aplicación se ejecuta con la velocidad adecuada?
5. ¿Ha sentido que, en algún punto del uso de la aplicación, no sabe cómo continuar?
6. ¿Los mensajes para prevenir errores son correctos?

Anexo 2: Encuesta de Satisfacción a especialistas sobre el uso del Sistema para la Confirmación de la Transportación Masiva de la Dirección de Transportación de la Universidad de las Ciencias Informáticas.

1. ¿La aplicación cumple con sus expectativas como cliente? En caso de ser negativa la respuesta, argumente por qué.
2. ¿Resulta de fácil ejecución la planeación y distribución de los transportes actualmente?
3. ¿La aplicación se ajusta a la ejecución actual de los procesos de Transportación en la Universidad?
4. ¿Los reportes que brinda la aplicación son lo suficientemente descriptivos y cubren sus necesidades de información?
5. ¿La aplicación se ejecuta con la velocidad adecuada?