Universidad de las Ciencias Informáticas

Facultad 1

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

Sistema Integrado de Transportación (SIT). Implementación de los módulos Reportes, Reservaciones y Gestión de la Información

AUTORES: David Leonardo Nieves Naranjo.

Leonardo Uria Sánchez**.**

TUTORES: Ing. Alexander Rodríguez Mompié.

Ing. Yariel Ramos Negrín.

Ciudad de la Habana, 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

Dedicatoria

A mis difuntos abuelos que este es el regalo prometido de hace mucho tiempo.
A mamá y a papá por encaminarme correctamente y confiar en mis decisiones.
A mi hermanito Esley que sus consejos son los mejores del mundo y que no se imagina lo mucho que lo quiero.
A mi Lily preciosa por guardarme un lugarcito ahí en su corazón.
Y especialmente a esa gran familia hermosa que tengo y que me apoya.

David.

A mis queridos padres que no tengo como agradecerles todo su apoyo y dedicación y que todo lo que he logrado ha sido pesando en ellos.

Leonardo

Agradecimientos

A nuestros tutores Yariel y Alexander por apoyarnos en todos los sentidos y por luchar día, tarde, noche y madrugada para mejorar mucho más la aplicación.

Al resto del equipo de programadores “letales” del proyecto y especialmente a Yoan Carlos Machado que es una mente maravillosa.

A todas las personas que nos ayudaron y apoyaron en estos 5 años de universidad y a los que estuvieron con nosotros en buenos y malos momentos.

Declaración de autoría

Por este medio declaramos que David Leonardo Nieves Naranjo y Leonardo Uria Sánchez somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.
Para que así conste firmo la presente a los _ _ días del mes de _ _ _ del año _ _ _.

David Leonardo Nieves Naranjo

_____
Firma del Autor

Leonardo Uria Sánchez

_____
Firma del Autor

Ing. Alexander Rodríguez Mompié

_____
Firma del Tutor

Ing. Yariel Ramos Negrín

_____
Firma del Tutor

Opinión del usuario del trabajo de diploma

El Trabajo de Diploma titulado “Sistema Integrado de Transportación (SIT). Implementación de los módulos Reportes, Reservaciones y Gestión de la Información”, fue desarrollado en la Universidad de las Ciencias Informáticas. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

Totalmente _____

Parcialmente en un _____%

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes:

_____

_____

_____

_____

_____

_____

_____

_____

_____

Y para que así conste, se firma la presente a los ___ días del mes de _____del año _____

_____	_____
Representante de la entidad	Cargo
_____	_____
Firma	Cuño

Opinión del tutor del trabajo de diploma

Título: “Sistema Integrado de Transportación (SIT). Implementación de los módulos Reportes, Reservaciones y Gestión de la Información”.

Autores: David Leonardo Nieves Naranjo y Leonardo Uria Sánchez

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de ____ puntos.

Ing. Alexander Rodríguez Mompié

_____

Firma del Tutor

Ing. Yariel Ramos Negrín

_____

Firma del Tutor

_____

Fecha

Resumen

En la Universidad de las Ciencias Informáticas (UCI) existe la necesidad de efectuar la transportación de personal a varias escalas, lo cual genera una planificación de los recursos puestos a disposición para dicha actividad. Para lograr una mejor eficiencia y optimización del uso de dichos recursos, es necesario contar de forma organizada con la información concerniente a este proceso, debido a ello, el Departamento de Transportaciones Nacionales UCI solicitó un sistema que le facilitara la gestión de toda la información generada en los procesos de solicitudes de transportes, surgiendo así el Sistema Integrado de Transportación (SIT), el mismo está compuesto por varios módulos; de ellos este trabajo aborda la descripción de la implementación que se llevó a cabo sobre Reportes, Reservaciones y Gestión de la Información. Su novedad e importancia práctica radica en que por primera vez se logra en la universidad un software capaz de gestionar de forma integrada todas las transportaciones que la misma presenta y que es posible configurar dada su gran flexibilidad.

Este documento recoge todos los resultados de la investigación realizada, en el cual se estudian las características esenciales de sistemas similares. Además de toda la documentación que genera la implementación de los módulos Reportes, Reservaciones y Gestión de la Información del SIT.

Palabras Claves: Reservación, Gestión, Módulo, Información, Reportes

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

En vista al ascendente desarrollo de las Tecnologías de la Informática y las Comunicaciones (TICs), nuestro Comandante en Jefe Fidel Castro Ruz se dio cuenta de la necesidad de que Cuba no podía quedar exenta a dicho progreso, y decidió crear una entidad que garantizara el desarrollo de la informática en el país. Así fue creada la Universidad de las Ciencias Informáticas (UCI), proyecto que constituiría, además del centro de la industria del software en Cuba, la primera ciudad digital del país como prototipo de una posible ciudad del futuro, donde todos o la mayoría de los procesos que ocurren en ella serían controlados por sistemas automatizados los cuales se producirían en la misma.

Hoy en día en la UCI se forman profesionales de la informática de todo el territorio cubano, donde muchos de sus trabajadores y estudiantes proceden de todas las provincias del país, a los cuales la universidad les garantiza el transporte para viajar a sus respectivos destinos; pero el mundo y en especial nuestro país enfrenta una crítica situación con el combustible, situación que obligó a optimizar el uso de los transportes destinados a estos viajes, por lo que se hizo necesario brindar dicho servicio mediante reservaciones para un mejor control. El Grupo de Informatización de la universidad el cual rige los proyectos destinados a crear los programas que automatizan los servicios que se brindan a las más de diez mil personas que conviven dentro de la UCI decidió informatizar el antes mencionado proceso de reservaciones, para que los usuarios pudieran reservar su transporte y los directivos del Departamento de Transportaciones Nacionales UCI obtuvieran reportes concretos que le permitieran tomar mejor las decisiones. De estas reservaciones existen tres modalidades:

- Transportación Nacional para estudiantes, profesores y trabajadores en fin de año y fin de curso.
- Transportación Nacional para los profesores y trabajadores internos una vez por semestre.
- Transportación de Fin de Semana hacia todos los municipios de Ciudad de la Habana para los estudiantes.

Hasta ahora se habían implementado algunas soluciones para estas modalidades que de forma separada funcionaban pero no cubrían todas las necesidades del Departamento de Transportaciones Nacionales UCI, en el caso de la Transportación Semestral de Trabajadores Internos aún no existía solución alguna. Los procesos de reservaciones de las aplicaciones no cumplían con todos los requerimientos funcionales que este departamento necesitaba. En el caso de la Transportación Nacional no se tenían en cuenta los datos de los familiares y el proceso de distribución de las solicitudes se realizaba solo por el orden de las reservaciones. La Transportación Estudiantil de Fin de Semana no permitía reservar en un periodo que no fuera desde el sábado por la tarde hasta el domingo por la noche y la Transportación Semestral de Trabajadores Internos se realizaba manualmente auxiliándose del Microsoft Excel y el correo electrónico.

¿Qué está ocurriendo realmente con la Transportación Semestral de Trabajadores Internos?

Toda la información es recogida en una hoja o plantilla que no presenta, en muchas ocasiones, las características necesarias para dar a conocer las informaciones o los reportes que se necesitan, no solo desde el punto de vista de los usuarios, sino para cada uno de los involucrados directamente o no en los procesos de reservaciones. Lo que hace que el tráfico de la información sea lento y poco seguro en la mayoría de los casos, es decir, que hasta el momento no existe una manera estándar para el manejo de esta información, esto imposibilita el cumplimiento de la entrega rápida de la misma hacia los niveles centrales de la universidad y el propio Departamento de Transportaciones Nacionales UCI.

Por otra parte, los usuarios no cuentan con un sistema a través del cual puedan informarse acerca de sus reservaciones, cuándo y cómo se efectuarán las mismas y valorar sus intereses al respecto. Todos los involucrados en el proceso no cuentan con una herramienta capaz de tener una actualización rápida y eficiente de la información ya que los reportes de cada aplicación no tienen forma alguna de vincularlos entre ellos. El cliente no cuenta con una fuente de datos ajustada a sus necesidades para obtener los datos precisos de los usuarios para evitar errores.

Por tanto el **Problema a Resolver** queda formulado de la siguiente forma: Necesidad de implementar los módulos Reportes, Reservaciones y Gestión de la Información para el Sistema Integrado de Transportación que automatice los procesos de gestión de las reservaciones de la Dirección de Transporte en la Universidad de las Ciencias Informáticas.

El **Objeto de Estudio** lo constituyen los Procesos de gestión y reservaciones del Departamento de Transportaciones Nacionales en la Universidad de las Ciencias Informáticas. Todo ello vinculado a las herramientas y metodologías para el diseño de sistemas informáticos, y el campo de acción que abarca este trabajo, son los módulos Reportes, Reservaciones y Gestión de la Información.

Partiendo de la **Hipótesis** de que la implementación de los módulos Reportes, Reservaciones y Gestión de la Información del Sistema Integrado de Transportación en la UCI incrementará la eficiencia en el desarrollo de los procesos de reservaciones y gestión de la información en el Departamento de Transportaciones Nacionales, surgen las siguientes variables:

Variables independientes:

- Módulos Reportes, Reservaciones y Gestión de la Información del Sistema Integrado de Transportación en la UCI.

Variables dependientes:

- Eficiencia en los procesos que se desarrollan en el Departamento de Transportaciones Nacionales UCI.

El **Objetivo General** de esta investigación es: Implementar los módulos Reportes, Reservaciones y Gestión de la Información, para un sistema que gestione los procesos de reservaciones de transportes en la Universidad de las Ciencias Informáticas.

Del cual se derivan los siguientes **Objetivos Específicos:**

1. Implementar la obtención de reportes en el sistema.
2. Concebir una propuesta de solución a los procesos de reservaciones de transportes para los usuarios del dominio UCI y la gestión de viajeros de forma manual.
3. Implementar un sistema de búsqueda para visualizar la información de los usuarios registrados o no en el sistema.

4. Implementar algoritmos para el envío automático o personalizado de correos en modo de aviso o de información a los usuarios del sistema.
5. Crear servicios web para otros sistemas de la comunidad universitaria.
6. Implementar funcionalidades que permitan revisar historiales de la base de datos.

Para dar cumplimiento a los objetivos específicos se trazaron las siguientes **Tareas Científicas.**

- Entrevistar a personas que estén inmersas en el proceso para identificar las necesidades y llevar a cabo una correcta automatización del mismo.
- Estudiar la metodología RUP (Rational Unified Process) en los flujos de trabajo de implementación, prueba y despliegue.
- Estudiar las herramientas a utilizar en el desarrollo del proyecto.
- Estudiar frameworks de desarrollo.
- Estudiar e investigar las funcionalidades e importancia de la arquitectura de MVC (Modelo Vista Controlador) que propone el framework Symfony.
- Implementar las funciones que distribuirán a los estudiantes y trabajadores en los transportes de la Transportación Nacional.
- Implementar la librería que permita brindar servicios web sobre la información que se almacena en la base de datos del sistema.
- Implementar métodos de búsquedas por diferentes parámetros de entrada para consultar la información almacenada en el sistema.

Para llevar a cabo estas tareas se emplearán métodos empíricos y teóricos de la investigación científica. Dentro de los **Métodos Empíricos** está la **entrevista**, la cual posibilitará obtener la información referente a cómo se espera que funcione la aplicación. Las entrevistas serán realizadas no solo a estudiantes, sino también a todo el personal involucrado en el proceso y la **encuesta** para conocer la opinión de las personas que van a interactuar con la aplicación y poder complementar los requerimientos del cliente.

Todo esto permitirá obtener información referente al tema, criterios y corregir malos conceptos que se puedan tener para cumplir con los requerimientos deseados.

En los **Métodos Teóricos** se usará el **análisis** y la **síntesis**, a partir de condiciones específicas se llegarán a ideas generales. La **inducción** y la **deducción** se utilizarán en diferentes etapas de la investigación, especialmente cuando de las ideas generales se deduzcan las particulares. El **histórico – lógico** para centrarse en los problemas que a lo largo de la historia han presentado estos procesos de reservaciones; posibilita rectificarlos y buscar entre todas las soluciones existentes la más óptima. Se utiliza la **modelación** porque se hace necesario explicarle al cliente mediante modelos, cómo se tiene pensado que quede el sistema para saber si cumple con sus necesidades. Se acude a la **medición** como **Método Matemático** para hacer pruebas a la aplicación y medir tiempos de respuesta ya que generalmente muchos datos son devueltos al usuario consultándose, además de la base de datos local, otros sistemas mediante servicios web.

Una vez concluida la investigación se esperan como **Posibles Resultados** los Módulos Reportes, Reservaciones y Gestión de la Información del Sistema Integrado de Transportación funcionales. Esto permitirá facilitar y mejorar los procesos de reservaciones de transporte, la información asociada a la misma referente a los usuarios y la entrega inmediata de toda la información necesaria ya sea cuantitativa o cualitativa generada en el Departamento de Transportaciones Nacionales UCI. Además, permitirá una centralización de todos los datos vinculados al proceso, rapidez en la comunicación y las búsquedas de información por parte de todo el personal involucrado, así como una disminución del consumo de material de oficina.

Este trabajo consta de tres capítulos: En el **Capítulo 1**, comenzando por la “Fundamentación Teórica”, se muestran los principales conceptos manipulados en el transcurso de la investigación y una breve referencia al estado del arte de las herramientas utilizadas en el mundo para dar solución a problemas similares. El **Capítulo 2** “Descripción y Análisis de la Solución Propuesta” se orienta a la implementación del sistema, además de las descripciones de clases y algoritmos complejos desarrollados o utilizados en la investigación. En el **Capitulo 3** “Validación de la Solución Propuesta” se realiza el diseño de los test de unidades que permitan validar la solución propuesta mediante una librería dedicada; se describirán los valores utilizados para las pruebas y finalmente se realizará la evaluación de la ejecución del mismo;

además de las Conclusiones, Recomendaciones, Referencias bibliográficas, Bibliografía, Glosario de términos y Anexos.

FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El proceso de selección de las condiciones, ambiente de trabajo y la forma en que se va a elaborar un software es fundamental para el progreso adecuado de la confección del mismo puesto que puede influir tanto en el tiempo como en la calidad del trabajo. En este capítulo se realiza un análisis de los sistemas de reservaciones del mundo y sus principales procesos de los cuales se tomaron las características que podrían ser compatibles al sistema de la universidad ya que hacía falta una visión más profesional del tipo de producto que se pensaba desarrollar. También se explican las herramientas y metodologías que se seleccionaron dentro de aquellas que más se adaptaban a las condiciones requeridas por la universidad, mencionando sus principales características, ventajas y desventajas.

1.2 Conceptos asociados al dominio del problema

1.2.1 Transportación Nacional o Masiva

Enmarca la parte de la aplicación que se relaciona con las reservaciones y la administración de las salidas y entradas masivas que ocurren para fin de año y fin y nuevo curso para todo tipo de persona, incluye definición de viajes, transportes, generación de listados y reportes.

1.2.2 Transportación Semestral de Trabajadores Internos

Se refiere a las reservaciones que únicamente pueden hacer los trabajadores de la UCI para los viajes semestrales a que tienen derecho. Incluye el concepto de lista de espera, administración y generación de listados y reportes.

1.2.3 Transportación Estudiantil de Fin de Semana

Reservaciones a las que solo los estudiantes tienen acceso. Las salidas y entradas son para municipios de Ciudad Habana.

1.2.4 Reservación

Solicitud de un transporte en un viaje específico. Una reservación puede estar compuesta por los datos de la persona que la solicita, los datos del viaje y, en el caso de la Transportación Masiva y de Trabajadores Internos, los datos del transporte.

1.2.5 Transporte

Entidad que representa a un ómnibus, tren o barco. Tiene como atributos esenciales nombre, tipo de transporte, punto de salida, fecha de salida y de llegada, viaje a que pertenece y la capacidad total.

1.2.6 Viaje

Entidad principal por la cual se rigen las reservaciones. La fecha es el atributo más importante. Puede ser nombrado también como **salida y entrada**. Es el pilar del sistema sobre el cual se realizan distribuciones y se generan reportes.

1.2.7 Modalidad de Reservación

Reservaciones para la Transportación Nacional o Masiva, Transportación Semestral de Trabajadores Internos y la Transportación Estudiantil de Fin de Semana.

1.2.8 Gestión de Información

Acciones que realizan la aplicación y/o el usuario para mostrar y manipular información referente a las modalidades de reservación; incluye el envío de correos, generación de historiales y manipulación de todo tipo de mensajes que se le muestran al usuario.

1.2.9 Reportes

Listados con reservaciones y distribuciones que se generan a partir de un viaje o salida. Están organizados por distintos criterios como transporte, área, rutas, etcétera.

1.2.10 Servicio web

Extensión que brinda el Sistema Integrado de Transportación a otras aplicaciones autorizadas para gestionar reservaciones y obtener reportes mediante una interfaz basada en la Arquitectura Orientada a Servicios.

1.2.11 Sistema

Aplicación web capaz de gestionar todos los procesos del Departamento de Transportaciones Nacionales UCI.

1.2.12 Módulo

Conjunto de funcionalidades que se agrupan por ser comunes; son empaquetadas recursivamente en carpetas con nombre específico. Un módulo puede estar compuesto por módulos más pequeños.

1.2.13 Historial

Información que se extrae del resultado de consultas a la base de datos que generalmente posee carácter temporal y que con la cual se puede llegar a conclusiones.

1.3 Procesos de reservaciones

Uno de los beneficios más grandes que Internet proporciona son los sistemas de reservaciones en línea. En la actualidad son muchas las empresas que utilizan aplicaciones web para ofertar mayor comodidad a los clientes, dentro de las cuales se pueden mencionar las cadenas hoteleras, restaurantes y principalmente las aerolíneas. Estas aplicaciones pueden adaptarse a diversos tipos de establecimientos a la vez y constituyen una herramienta útil para el Marketing ya que son los que dan la cara ante el mundo. Por su facilidad de uso, bajo costo en procesos, tiempo y dinero para los usuarios es por lo que han tenido una gran aceptación en la población constituyendo una necesidad su uso en el mercado. Algunos de los grupos o empresas que utilizan estos medios son:

Reservaciones.com: http://www.reservaciones.com

Radixx: http://radixx.com

Viaju: http://www.viaju.com

Conviasa: http://www.conviasa.aero/

American Airlines: http://www.aa.com.ve/aa/homePage.do?locale=es_VE

En nuestro país también se han creado las condiciones para desarrollar este tipo de tecnologías y muchas instituciones las han adoptado empezando por las relacionadas con el turismo cubano dentro de las que podemos mencionar:

Travelnet Cuba: http://www.travelnetcuba.it/es/

Travels2 Cuba: http://www.travels2cuba.com/

All ways travel: http://www.allwaystravel.cu/

Cubana de Aviación S.A: http://www.cubana.cu/

Vamos a Cuba: http://www.vamosacuba.net

Las modalidades de las reservaciones que caracterizan a los sistemas utilizados por este tipo de empresas comerciales están dadas por el tipo de transporte, tipo de persona y el precio de la reservación según la distancia y/o el confort del viaje. Para capturar los datos de los usuarios utilizan un sistema de registro de cuentas las cuales reutilizan cada vez que la misma persona entra a reservar y es a través de su correo electrónico el medio donde se le hacen llegar la notificación y la confirmación de su reservación.

El sistema utilizado por Radixx es uno de los más dinámicos en cuanto a reservaciones de viajes, aunque se enfoque en reservaciones de vuelos solamente. El mismo permite efectuar reservaciones estándares, aceptar lista de esperas y reservar pasajeros sin cargo. Las reservaciones pueden provenir de varias fuentes, ya sean directamente del cliente, agente de viaje u otro tipo de fuente asignada por el usuario. La modalidad de reservación desbalanceada ofrece al pasajero viajar en diferentes itinerarios pero en un mismo record reduciendo así la necesidad de tener que hacer reservaciones fragmentadas. Se puede actualizar rápidamente cualquier reservación y hacer cambios en varios idiomas. Este sistema guarda el historial de la reservación incluyendo todos los cambios que se le efectuaron a la misma desde su creación en una base de datos histórica para una verificación fácil y recuperar un PNR (Passenger Name Record) o registro de nombre de los pasajeros. Según las necesidades Radixx posee múltiples niveles de acceso vía Internet. Los trabajadores de reserva y sus supervisores pueden acceder al módulo central de reservaciones para hacer cambios, ver reportes, manejar el inventario y otras funciones básicas todo ello

a través de Internet y usando una computadora personal estándar. El consumidor puede ver la disponibilidad de asientos, solicitar en línea y elegir su asiento.

Radixx.com cuenta con un sistema de avisos que consta de una pantalla administrada por el usuario que contiene temas de importancia para mantener al personal al día con los cambios recientes en la compañía o en la industria. Los avisos pueden ser creados o modificados fácilmente y se les puede asignar una fecha de vencimiento a los mismos. Todos los reportes del sistema se pueden exportar en formatos comunes de bases de datos para mayor análisis, o para la extracción y depuración de la información.

El marco más cerrado que se puede analizar lo constituye la Universidad de las Ciencias Informáticas en la cual se han desarrollado varias aplicaciones web destinadas a las solicitudes de algunos de los servicios que se brindan dentro de la universidad, con características muy particulares ajustadas a las condiciones tecnológicas del centro las mismas son:

Reservación del gas de la UCI: http://serviciosg.uci.cu

Reservación de la transportación estudiantil para el fin de semana: http://pase.uci.cu

Reservación de la transportación masiva de estudiantes y trabajadores: http://smasiva.uci.cu

Estas constituyen aplicaciones básicas que se centran principalmente en el proceso de reservación y lo hacen de una manera trivial, además no abarcan todos los aspectos de la parte administrativa que requieren los clientes. El sistema de reservación del transporte estudiantil para los fines de semana constituye la aplicación más competente de todas ellas. El proceso de reservación de este sistema empieza cuando el usuario se autentica con el usuario y contraseña del dominio de la institución en este caso la UCI, donde los datos personales son tomados directamente de la base de datos de la primera versión del Sistema de Gestión Académica (AKADEMOS) de la universidad.

El estudiante tiene la posibilidad de reservar ida y/o regreso especificando el punto de llegada para la ida y el de salida para el regreso, dentro de las rutas que están definidas en la aplicación. Una característica que presentan las reservaciones de este centro es que requieren un margen de tiempo para realizarlas y en el de este sistema particularmente es fijo, por lo que se presentan problemas para extender la duración del pase. El tiempo definido para realizar las reservaciones es de lunes a partir de las 1:00 AM hasta el viernes a las 8:00 AM y para el caso de la cancelación de las reservaciones el límite se extiende hasta el

sábado a las 9:00 AM, es por ello que para que la aplicación coincida con el pase el mismo debe durar solamente desde el sábado por la tarde hasta el domingo por la noche. Una vez efectuada la reservación el usuario podrá modificarla dentro del período. El sistema es capaz de enviar un correo con la confirmación de realización, modificación o cancelación de una reservación, también genera los boletines de cada pasajero de toda la UCI, por facultades, por grupos e individual para un estudiante. A modo de información el sitio en su página principal cuenta con un mapa de los puntos de salida de la UCI y con un listado de las rutas de los ómnibus. Los administradores pueden consultar reportes como la cantidad de reservaciones por rutas, por años y por facultades y cuáles son las mismas.

Las aplicaciones anteriormente expuestas solo aportan ideas de cómo funcionan los negocios de las empresas que utilizan estos tipo de sistemas. Esto constituye un avance en la toma de decisiones a la hora de cómo hacer alguna aplicación parecida. Ninguna de ellas cumple con todos los requerimientos que se necesitan para gestionar y controlar los procesos de todas las modalidades que existen en la universidad.

1.4 Tendencias y tecnologías actuales

1.4.1 PHP5

El lenguaje de programación de páginas web PHP inicialmente fue concebido para darle solución rápida a problemas que surgían en cuanto a las aplicaciones. Actualmente mantiene su concepto y, además, ha evolucionado. Esta nueva versión contiene cambios importantes que afecta cómo los programadores trabajarán. Las modificaciones principales están relacionadas con la programación orientada a objetos que aunque estaba soportada desde versiones anteriores, solo abarcaba una parte muy pequeña de las características de este paradigma de la programación. Se ha escogido desarrollar este sistema sobre PHP5 para así cumplir con demandas del cliente y garantizar la rapidez, robustez y mantenimiento extendido que caracteriza a las aplicaciones desarrolladas con este lenguaje.

1.4.2 Aplicaciones Web

Hoy, el principal uso de de la World Wide Web es para el acceso interactivo a documentos y aplicaciones **(1).** En casi todos los casos, estos accesos son hechos por los usuarios generalmente trabajando a través de navegadores, reproductores de música u otras interfaces interactivas. La Web ha crecido significantemente mejorando los soportes para las comunicaciones entre aplicaciones. Existe una

tendencia a llevar el comportamiento de las aplicaciones web a la forma en que las aplicaciones de escritorio son diseñadas, es decir, se evalúa una interfaz dinámica de un sitio y seguidamente la navegabilidad y la experiencia del usuario reafirman la nueva clasificación que han denominado Web 2.0. Se trata de aplicaciones que poseen características especiales independientemente de que sean tiendas electrónicas, bancos online, formularios de registros, planificaciones online, buscadores, chats, mercados o subastas. Este concepto abarca las nuevas tendencias de diseñar y programar lógicas del negocio lo cual se refiere a incluir interacciones complejas con el usuario y todas las posibilidades que ofrecen los navegadores Web para la manipulación de peticiones y multimedia.

1.4.3 Seguridad en Aplicaciones Web

1.4.3.1 Inyección SQL

La Inyección de SQL (Lenguaje de Consulta Estructurado) es un ataque en el cual el código malicioso es insertado en cadenas de texto que luego son pasadas al servidor o instancia de SQL para ejecución. Es inminente que los procedimientos que construyen las declaraciones deban ser revisados ya que el servidor SQL ejecutará todas las consultas válidas sintácticamente que reciba. Los ataques suelen ser muy complejos e inteligentes y principalmente dirigidos hacia los datos parametrizados.

Inicialmente la inyección SQL consiste en la inserción directa de código a variables que toman valores de acuerdo a lo que el usuario introduce y que luego son concatenados con comandos SQL y luego ejecutados. El proceso de inyección es prematuramente terminar una cadena de texto y luego anexar un comando. Esto tiene una muy cercana relación con los ataques que son dirigidos al almacenamiento de datos o metadatos en las tablas, cuando las cadenas son guardadas y subsecuentemente atadas a un dinámico comando SQL, entonces es cuando el código malicioso es ejecutado.

Por lo tanto, se deben tomar medidas y planear la forma en que se validará la integridad del sistema en contra de este tipo de ataques ya que hay que tener en cuenta las habilidades que pueden tener los hackers:

Implementar múltiples capas de validación. Las precauciones que se toman contra los usuarios maliciosos pueden resultar ser inefectivos contra atacantes determinados.

Una buena práctica es validar las entradas en la interfaz del usuario y así todos los puntos y capas por donde pasan los valores. Validaciones de datos en una en una aplicación del lado del cliente puede prevenir inyecciones simples de scripts. De todas formas, si en la próxima capa de validaciones se asume que la entrada ya se ha validado, cualquier usuario malicioso que pueda sobrepasar un cliente puede tener acceso no restringido al sistema. La solicitud de servicios online como viajes y transportes por parte de los usuarios, por muy básica que sea, es muy vulnerable.

1.4.3.2 XSS (Cross-site Scripting)

Cross-site Scripting (Ataques a navegadores mediante scripts) o XSS es una vulnerabilidad usualmente encontrada en aplicaciones web en la cual se usan códigos scripts para atacar a otros usuarios. Pueden ser utilizadas para sobrepasar controles de acceso y corromper el comportamiento del navegador en general. El ejemplo más común de ataques XSS es en una entrada de Blog o Fórum donde el usuario final puede adicionar contenido a la aplicación web para que otros la consulten. En una entrada donde se permitan comentarios cualquier usuario puede escribir contenido dentro del campo de comentarios que luego va a ser insertado en la base de datos o, más allá de esto, el código malicioso escrito puede redireccionar la página mostrada a otro sitio incluyendo los datos de la sesión y las cookies registradas. **(2)**

Hay que llevar a cabo políticas de seguridad en la etapa de desarrollo de las aplicaciones e ir construyendo la conciencia de utilizar al máximo las funciones y utilidades que nos brinda PHP para el reconocimiento y filtrado de la información manipulada ya que los ataques no solo pueden ser a través de campos de entrada o selección, sino también por vías mucho más complejas de manipular que son las peticiones y las cabeceras; los atacantes aprovechan la dependencia de aplicaciones de manejar datos confidenciales y de solicitudes que son enviadas a través de la red y con la ayuda de software y scripts que generan código y lo adjuntan en el momento preciso en que la información se encuentra “en el aire”.

La gestión de Reservaciones e Información que realiza el Sistema Integrado de Transportación incluye validaciones contra este tipo de ataques y tiene máxima prioridad ya que estos módulos tienen estrecha relación el usuario final y son los encargados de presentar contenidos que pueden variar en dependencia de lo solicitado.

1.4.4 Arquitectura orientada a servicios

Es esencialmente una colección de servicios los cuales se comunican entre sí; esta comunicación puede estar vinculada a traspaso de datos simplemente o a varios servicios coordinando actividades; deben estar caracterizados por ser efectivos, bien definidos y no dependientes del estado o contexto de otros.

Los inicios de la utilización de esta arquitectura están unidos a las personas que comenzaron a utilizar CORBA, para especificar las interfaces que los objetos presentarán al mundo exterior. Las conexiones robustas para los servicios son garantizadas por XML (Lenguaje de Marcado Extensible) ya que este lenguaje es interpretado por cualquier entidad. En la Arquitectura Orientada a Servicios, se puede reemplazar un servicio sin tener que preocuparse por la tecnología fundamental; la interfaz es lo que importa, y está definida en un estándar universal lo cual garantiza flexibilidad a través de la interoperabilidad. **(3)**

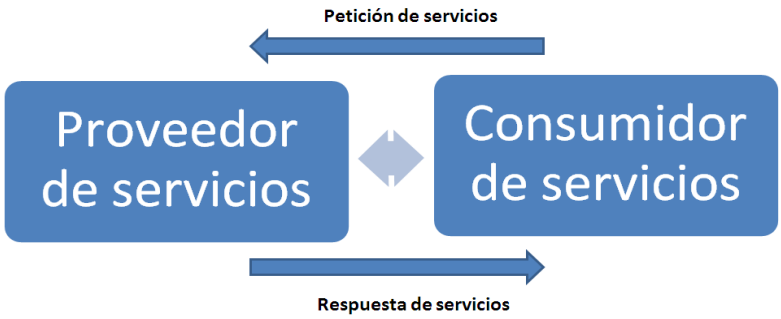


Figura 1 Arquitectura orientada a servicios

La estructura de los servicios web en la UCI puede resultar un poco complejo de entender, pero hay que tener en cuenta que la información que se brinda varía mucho con el tipo de persona (Estudiante, Trabajador, Profesor, Técnico, Egresado, etc.), por lo tanto, se dispone de un sistema de Gestión Académica que garantiza los datos referentes a los estudiantes y su vinculación a la docencia, los cuales son utilizados por el Sistema Integrado de Transportación para gestionar los datos de los viajeros y, al

mismo tiempo, por el Sistema Integral de Comedores para su diaria verificación y gestión de comensales. Guiados por los lineamientos de arquitectura definidos por la UCI, los desarrolladores de estos servicios web los han dotado de documentación para que los programadores los integren, así, estos módulos pueden funcionar como fueron definidos e integrados a la aplicación y además se comportan y gestionan información con la misma capacidad al ser expuestos, es decir, las funcionalidades son aprovechables sin tener que operar directamente en la aplicación.

Estas facilidades que nos brinda la Arquitectura Orientada a Servicios y sus especificaciones permiten desacoplar el módulo Reservaciones, Reportes e Información y adaptarlos e integrarlos a otros sistemas sin perder su esencia y ganando, a la misma vez, propuestas para ser mejorados constantemente ya que interactúan con numerosos usuarios y sistemas.

1.4.5 Framework

Desde principios de este siglo el lenguaje de programación PHP ha sido encapsulado casi en su totalidad. Aprovechando las amplias posibilidades que brinda para el trabajo con redes, servicios web, bases de datos y desarrollo general de aplicaciones, muchas Comunidades de Desarrollo, compañías y programadores han creado esta nueva tendencia de encapsular y organizar la mayoría de las funcionalidades a pequeños paquetes llamados Frameworks, caracterizados por tener gran flexibilidad, reutilización, extensibilidad y rendimiento.

Los frameworks a valorar contienen características comunes que los reúnen en la categoría de Desarrollo Rápido de Aplicaciones. Soportan conexiones a varios sistemas gestores de bases de datos como MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL y SQLite. Muchos están basados en el patrón Modelo-Vista-Controlador y por lo tanto, tienen buenos subsistemas que gestionan plantillas o vistas y proveen al desarrollador facilidades para manipular peticiones y acoplar la información que brinda el modelo de datos con la forma en que es presentada.

1.4.6 Zend Framework

Creado por la compañía Zend Technologies la cual posee la autoría del propio PHP, es muy querido por las comunidades de desarrollo debido a su arquitectura caracterizada por un bajo acoplamiento entre sus módulos, es decir, componentes como el potente sistema de búsqueda e indexado llamado Lucene o las

Listas de Control de Acceso (ACL) que posee pueden ser desacoplables e incorporables en cualquier otro framework.

1.4.7 CodeIgniter

Es muy valorado por su sencillez y rapidez para crear aplicaciones Web elegantes y completamente funcionales. Su compatibilidad con versiones antiguas de PHP hace que su flexibilidad sea considerable. Este Framework en sí es muy ligero (kilobytes), característica que puede convenirle a aquel programador que prefiere desarrollar sin herramientas.

1.4.8 CakePHP

Aparte de su rapidez, posee una buena implementación de seguridad y control de acceso mediante ACLs (Listas de Control de Acceso). Trae consigo una modesta capa para el modelo de datos la cual provee de funcionalidades necesarias para lograr una buena abstracción. En la capa de los Controladores y las Vistas dispone de una serie de ayudantes que agilizan la elaboración del sistema lo cual, unido al buen performance que lo caracteriza, el tiempo de respuesta es bastante bajo mejorando así la experiencia del usuario.

1.4.9 Symfony

Hasta ahora es el framework que más documentación oficial posee. Con varios libros publicados por sus creadores como “The Definitive Guide To Symfony” y “Practical Symfony”, cada programador tiene disponible una ayuda constante sobre cada detalle de la innumerable cantidad de clases, funciones y utilidades que brinda.

Su grado de madurez mejora con la utilización del I18N (Internacionalización y traducción de interfaces) y un novedoso sistema de plugins lo cual lo hace mucho más extensible. Ya que el Sistema Integrado de Transportación utiliza el diseño y arquitectura que brinda esta herramienta, es importante mencionar cuánto se ha aprovechado su capacidad de abstraer el modelo de datos y de sugerirnos de qué manera pueden ser mostrados y gestionados.

Se adapta a las nuevas tecnologías soportando distintos tipos de formatos para peticiones como JSON, XML, RDF, ATOM y YAML, es decir, una aplicación desarrollada en Symfony puede ser tan extensible como para comunicarse y brindar servicios web a dispositivos móviles de última generación como el iPhone. **(4)**

Su sistema de helpers (ayudantes) es tan amplio que abarca todas las fases y modalidades de la construcción de aplicaciones web:

1. Mecanismos de escape de valores para contrarrestar ataques XSS.
2. Manipulación del sistema de cache del PHP para optimizar la rapidez del sistema.
3. Formateo de fechas y cifras que guían al usuario a una mejor comprensión de la información mostrada.
4. Encapsulación de numerosas funciones Javascript para mejorar el trabajo con las plantillas y modernizar el sistema.
5. Validadores basados en expresiones regulares para el uso de formularios que manipularán datos obtenidos de los usuarios.

Es importante mencionar que los datos almacenados son muy útiles en la misma medida que se puedan transformar en información adecuada para las personas que los necesitan. Esta plataforma tecnológica que se utiliza tiene facilidades para convertir los datos en información y poder entregarlos a los usuarios de forma que sean útiles y reusables.

1.4.10 ORM (Mapeo Objeto-Relacional)

Existen frameworks que encapsulan solamente algunas de las capas en que una aplicación web se puede dividir (Acceso a datos, Seguridad, Presentación, Negocio, etc.), Propel y Doctrine liderean en el grupo de frameworks que garantizan el Acceso a datos, soportando todo tipo de conexión a los diferentes gestores como PostgreSQL, Oracle, MSSQL, MySQL, SQL Lite y otros. Una de las ventajas de utilizar estas capas de abstracción de objetos-relacional es que evita utilizar una sintaxis específica de un sistema de bases de datos. Sin importar el gestor que se esté utilizando en el momento, las llamadas a métodos de objetos que representan al modelo de datos se transforman en consultas SQL muy optimizadas y que luego se ejecutan en el momento preciso para evitar conexiones innecesarias al servidor.

Symfony aprovecha la integración con en lenguaje YAML para su configuración general para brindarle al usuario la posibilidad de escoger el ORM a utilizar, ya sea Propel o Doctrine.

1.4.11 TCPDF

TCPDF es una librería Open Source para PHP que permite crear ficheros PDF dinámicamente. Las dos cualidades más valoradas son su simplicidad a la hora de crear archivos PDF y la capacidad de interpretar código XHTML. Otras características que resultan ventajosas en aplicaciones web son los métodos para la interpretación de código HTML y que incluye Javascript a las formas de apoyo. Además, soporta configuración incluyendo los métodos para creación de cabeceras y pies de páginas, quiebre y número de hojas automático, quiebre de línea, justificación automática y compresión de página. Es capaz de soportar colores e imágenes, incluyendo gráficos y métodos de transformación. Todas estas funcionalidades y las básicas son resueltas sin necesidad de utilizar bibliotecas externas. Gracias a la integración hecha con el Symfony, los módulos Gestión de la Información y Reportes usan esta herramienta para la generación de listados finales de los viajeros y sus datos de las reservaciones, boletines y reportes específicos que recogen información variada acerca de los datos guardados.

1.4.12 Frameworks para Javascript

El lenguaje de programación de páginas web conocido como Javascript(ECMAscript oficialmente) está muy unido al desarrollo con (X)HTML, PHP y todos los lenguajes del lado del servidor, por lo tanto, todo programador tiene disponible varias facilidades que brinda Javascript por el lado del cliente; aún más, frameworks como JQuery, Prototype, Script.aculo.us, ExtJs, YUI, DOJO y Mootools se han caracterizado por encapsular completamente el DOM (Document Object Modeling) y el BOM (Browser Object Modeling) de todos los navegadores actualmente existentes (Internet Explorer, Opera, Netscape, Firefox, Safari, etc.) y los estándares que siguen mostrándonos posibilidades muy creativas y novedosas de construir páginas web.

Prototype y Script.aculo.us están inicialmente integrados a Symfony, esto no quiere decir que sea obligatoria su utilización. Muchos ayudantes han sido implementados para su uso durante la etapa de desarrollo y, para una mayor integridad, no solo se dispone de calendarios con curiosos diseños y efectos de última generación, también se cuenta con la internacionalización (I18N) y localización (I10n) para adaptar estos contenidos a las necesidades de las personas que consumirán la solución.

Es de resaltar el trabajo que han hecho las Comunidades de desarrollo para que se cumplan algunas filosofías que abundan alrededor de los programadores como el concepto DRY (No te repitas o No repitas las cosas) y KISS (Hacer las cosas simples o sencillas) **(5)**; lo cual convierte a los frameworks en poderosas herramientas que como unos de los principales atributos es la velocidad que ganan y que le brindan al sistema en el que estén integrados.

1.4.13 Patrones de Diseño

Definidos como una solución reusable para un problema que comúnmente ocurre durante el diseño del software, son plantillas o descripciones para saber cómo resolver un problema que puede ser utilizado en contextos distintos. **(6)**

Ellos en sí forman un lenguaje que puede ser utilizado para describir soluciones clásicas al diseño de problemas comunes orientados a objetos; nos permiten analizar sistemas de objetos como entidades encapsuladas, lo cual nos hace darnos cuenta de la estrecha relación con la programación orientada a objetos. Pueden aumentar la rapidez del proceso de desarrollo proveyendo paradigmas sólidos. Un diseño efectivo de software requiere que se consideren problemas que no puedan ser detectados hasta que llega la Implementación; lo cual conlleva a mejorar la legibilidad para los programadores y arquitectos.

Usando *Singleton* para garantizar la exclusividad de determinados recursos como la conexión a la base de datos, la conexión a los diferentes servicios web que consume el sistema y el trabajo con las sesiones de usuario; Symfony le recuerda al programador cuán importante es la organización en el código.

1.4.14 Patrones de Arquitectura

Los patrones de arquitectura ofrecen soluciones bien establecidas a problemas arquitecturales dentro de la Ingeniería del software; describen los elementos y las relaciones junto a especificaciones de cómo pueden ser usados. La Arquitectura del Software comprende los primeros pasos de las decisiones del diseño para un sistema, por lo tanto, se acude a la separación de capas para contrarrestar las consecuencias de las modificaciones futuras. Patrones como MVC, PAC, Seeheim, Arch/Slinky están basados en separar la interfaz del usuario de las demás capas.

Un sistema de software estará estructurado según el esquema que brinda su patrón de arquitectura.**(7)** El Modelo-Vista-Controlador, enmarcado en los dos tipos de patrones, tiene una gran ventaja y es importante mencionar las posibilidades que brinda para mantener una aplicación web que pueda ser utilizada en diferentes entornos como diferentes navegadores en computadoras o en dispositivos móviles.

1.4.15 Open LDAP

LDAP www.openldap.org (Implementación libre del Protocolo ligero de acceso a directorios) es un protocolo para comunicaciones entre servidores y clientes LDAP. Estos servidores almacenan “carpetas” las cuales son accedidas por los clientes LDAP. Se le llama ligero (lightweight) porque es un pequeño y fácil protocolo que fue derivado del X.500 DAP (Protocolo de acceso a directorios) definidos en la capa de red del modelo OSI (Interconexión de Sistemas Abiertos). No se limita a información de contacto o incluso a información acerca de las personas; se usa también para buscar certificados encriptados, apuntadores a impresoras y otros servicios en la red, además, tiene la posibilidad de que una contraseña de un usuario pueda ser utilizada para acceder a varios servicios. Este protocolo es muy apropiado para información almacenada en forma de carpetas donde generalmente se usen búsquedas rápidas y se actualice de forma no frecuente. LDAP no define cómo los programas funcionan, sino el lenguaje usado por programas clientes para comunicarse con el servidor, o servidores con servidores.

La mayoría de los clientes LDAP solo leen de un servidor. Las habilidades de búsquedas que tienen algunos clientes como los programas de correos pueden variar mucho, pocos pueden actualizar o introducir información. La encriptación de datos no está incluida y además, generalmente se requiere de una conexión SSL (Protocolo de Capa de Conexión Segura) para acceder al servidor.

El directorio LDAP de la UCI provee, aparte del sistema de autenticación, información básica y oficial en cuanto a los usuarios del dominio. El Sistema Integrado de Transportación utiliza atributos ofrecidos para verificar la integridad de cada individuo y el estado actual del registro que posee en la Universidad. De la información que manejan los módulos de Reservaciones, Reportes y Gestión de la información, mucha es consultada de este directorio, evitando la copia y repetición de datos y aprovechando la velocidad de respuesta que generalmente caracteriza a este protocolo.

1.5 Conclusiones

Uno de los grandes problemas que se pueden presentar en la elaboración de un software es la selección de una herramienta inadecuada sobre todo si se trata del lenguaje que se piensa utilizar ya que cualquier inconformidad generada por insuficiencias del mismo conlleva a retrasos del flujo de actividades. Es por eso que este paso dentro de la planificación de la aplicación debe hacerse de manera cuidadosa para no tomar decisiones erróneas que puedan traer resultados catastróficos durante todo el proceso de desarrollo. Las tecnologías y formas de desarrollo evolucionan rápidamente, por lo tanto, las aplicaciones web evolucionan, son mantenidas y se adaptan a los nuevos entornos; pero es tarea de todo buen desarrollador guiarlas teniendo en cuenta la actualización en los conocimientos adquiridos.

DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En este capítulo se documentan las principales funcionalidades y procesos que ocurren en el sistema en cuanto a los módulos Reportes, Reservaciones y Gestión de la Información. Se pretende mostrar cómo está aprovechada la arquitectura y las posibilidades que proporcionan el framework y las librerías utilizadas en la programación de aplicaciones web en el producto que se está tratando, con el objetivo de facilitar la comprensión del funcionamiento de los módulos anteriormente mencionados.

2.2 Valoración crítica del diseño propuesto por el analista

El Sistema de Integrado de Transportación no contó con un previo análisis y diseño específicos destinados a su implementación, el mismo fue implementado combinando y ampliando la arquitectura de los diseños hechos para los anteriores sistemas que se encontraban por separado. Al fin y al cabo las antiguas aplicaciones existentes tenían un propósito común y constituían sistemas que de maneras parecidas administraban las distintas modalidades de reservación, por lo que a la hora de integrar estos módulos de administración se siguieron los mismos requisitos.

El único elemento tenido de forma integrada desde el análisis fue el modelo de la base de datos, aún en aquel entonces no se pensaba en un sistema integrado para todas las reservaciones, se pensó de manera que las distintas aplicaciones aprovecharan algunas tablas nomencladoras y de datos que almacenaban información, incluso podemos mencionar que el sistema de reservación del gas que se pensaba implementar en la universidad estaba asociado a la misma. Como medida para la compatibilidad de los datos se decidió desechar muchas de las tablas nomencladoras y algunos datos que podían ser obtenidos de los servicios web que son brindados por otros sistemas en la universidad como el Sistema

Gestión Académica (Akademos) el cual aporta la mayor información de profesores y estudiantes registrados en el centro, pero sin dejar de reconocer el buen trabajo hecho por el arquitecto en su diseño inicial que logró una integración lo suficientemente flexible capaz de ser compatible a la nueva integración.

La gran diferencia entre los negocios de cada modalidad radica en las formas de reservar ya que no siempre están destinadas al mismo tipo de personal, ni con las mismas opciones de reservación y las formas de salida y el tiempo de duración también convergían en cada sistema. Se decidió tratar a cada una de las distintas modalidades de forma separada pero en una misma página dinámica y a nivel de base de datos la herencia tomaría parte en la inserción de los datos en tablas de reservaciones especializadas de una reservación generalizada, lo que posibilitó la integración esperada. Fue muy útil la descripción hecha por los analistas de los tres sistemas pera llegar a una concepción de cómo sería la mejor forma de llevar la navegabilidad del sitio para lograr que el usuario lo viera como un solo sistema, pero sin dejar de cumplir con los requerimientos del mismo.

La principal estrategia utilizada para unir cada transportación fue la creación de los viajes donde se creó una estructura genérica que asimilaba las tres variantes de transportaciones, de los mismos se extraerían los reportes, boletines y listados asociados y de esta manera se podía tener en cuenta a qué tipo de viaje se entraba y por el mismo se sabía en qué modalidad se encontraba. La estructura por viajes posibilitó posicionar los mismos reportes que se describían en los análisis existentes que aún no estaban integrados, en su aplicación correspondiente; de esta manera se aprovechaban las mismas descripciones.

Dentro del módulo Gestión de la Información surgieron funcionalidades junto a la integración como la implementación de servicios web y los historiales pero existían funcionalidades que coincidían en todas las aplicaciones como era el caso de uso Buscar Usuario y otras que ya estaban descritas para algunos casos específicos de transportaciones y que mediante la estructura de los viajes se lograron expandir a otros tipos de reservaciones.

El punto donde más avance se obtuvo en cuanto a la implementación de las aplicaciones anteriores lo constituyeron los métodos que distribuyen los pasajeros reservados en sus asientos correspondientes. El código de los mismos fueron reutilizados lo que tuvo gran importancia ya que dichos métodos son los más complejos del sistema donde se tienen en cuenta diversas condiciones manejando a su vez grandes cantidades de información y accediendo con mucha frecuencia a la base de datos.

De manera general gracias al análisis y al diseño propuesto por los analistas de las anteriores aplicaciones se avanzó más rápido a la hora de implementar, teniéndose en cuenta las necesidades del cliente para las tres modalidades de reservación y los procesos que se llevan a cabo por el personal encargado de la transportación en la universidad dados los levantamientos de requisitos y descripción de los mismos propuestas por los encargados del análisis y el diseño. También tuvo un gran aporte el hecho de que dos de esas modalidades ya estuvieran implementadas ya que constituyen las guías más concretas que podría tener los programadores.

2.3 Descripción de las modalidades de reservaciones

2.3.1 Reservación de la transportación estudiantil de fin de semana

Para esta modalidad el sistema debe tener en cuenta que solamente le mostrará la posibilidad de reservar a los estudiantes y que solo se podrá realizar durante el plazo de tiempo definido por el administrador por lo general se realiza de lunes a la 1:00 AM hasta el viernes a las 8:30 AM**.** Se le da la posibilidad de realizar una reservación personalizada donde se hace necesario que el estudiante para reservar llene los datos del familiar que va a visitar. En el instante que el estudiante termina la reservación se le envía un correo generado automáticamente como confirmación de la reservación y mostrándole a su vez los datos de su viaje. Una reservación de esta modalidad está conformada por el viaje a que pertenece, la ruta seleccionada y el punto final de llegada.

2.3.2 Reservación de la transportación semestral de trabajadores internos

Para que el sistema le muestre esta modalidad de reservación deberá reconocer a la persona que se autenticó como trabajador interno de la UCI, si es de Ciudad Habana o de provincia Habana tampoco se le dará la opción de reservar ya que la transportación es limitada y se prioriza a los trabajadores que realmente necesiten viajar a sus provincias. La reservación es personalizada tanto en la ida como en el regreso pero en este caso lo que se reserva es un transporte específico para su destino, los transportes disponibles deberán estar definidos antes del periodo de reservación. En el caso de que el usuario no logre reservar para alguna capacidad del transporte en el que quiere viajar, entonces podrá anotarse en la lista de espera del mismo esperando la posible cancelación de una reservación por algún otro usuario; el sistema automáticamente le reservará un asiento del transporte especificado tomando el primero en la cola de la lista de espera. La aplicación solo permite reservar un transporte semestralmente, solo se podrá tener acceso a los viajes y sus listas de espera cuando aún no se ha reservado o cuando se cancela una

reservación. En el caso de que no logre reservar por las vías formales el sistema da la posibilidad de reservar en todas las listas de espera que desee. Una vez que se termina el periodo de reservación definido por el administrador se envía un correo recordándole que debe confirmar la reservación; si no lo hace, cuando termina el plazo de confirmación el sistema no lo agrega a la lista de pasajeros, perdiendo el derecho de viajar el semestre completo y a su vez toma la primer solicitud de la lista de espera y ocupa dicha capacidad automáticamente. Como en los otros dos casos de reservaciones la aplicación para esta modalidad envía un correo confirmando su reservación y otro cuando ya tiene una capacidad asignada enviándole los datos del transporte en el que viajará y los datos de la salida.

2.3.3 Reservación de la transportación masiva

Una vez que el usuario es autenticado en el sitio, se muestran los bloques de salida a los que tiene derecho por su tipo de persona según la regla y en el tiempo de reservación que haya definido el administrador, que no son más que aquel conjunto de viajes que saldrán un día especifico. Las reglas consisten en condiciones declaradas una vez se define un viaje; en otras palabras, son los permisos que otorga el administrador a estos viajes para que las personas según las condiciones que cumplan como pueden ser su tipo de persona, área, provincia de la que proviene, si es estudiantes su año y otras características que son tenidas en cuenta. Estas reglas también son aplicadas a los viajes ya que dentro de un mismo bloque, puede haber viajes con distintos tipos de reservaciones definidas lo que hace necesario definirlos así. Las reservaciones para este tipo de transportación pueden ser personalizadas o estándar. Las personalizadas son aquellas en la que el usuario puede escoger tanto su destino de ida y lugar para el regreso. Las reservaciones estándares son aquellas en la que el sistema solo le da la opción al usuario de reservar para el municipio que aparece registrado en la base de datos de Akademos (Sistema de Gestión Académica de la UCI) ya que la aplicación lo toma del servicio web disponible. El sistema le envía un correo de confirmación como constancia de que reservó correctamente y una vez se procede a la distribución de las reservaciones en los transportes; al usuario se le es asignada una capacidad en un transporte especifico y se le envía otro correo con los datos del vehículo que lo transportará, la hora y el punto de salida tanto para el viaje de ida como para el de regreso.

2.4 Descripción del proceso de Distribución de la Transportación Masiva

La distribución consiste en asignar reservaciones de un viaje a transportes con capacidades disponibles según los destinos. Ya que esta modalidad de reservación abarca movimiento de personas hacia todos los municipios del país, primeramente se organizan los transportes según los destinos y paradas intermedias (municipios) que tengan. La interfaz consiste en una vista panorámica informativa de la cantidad de reservaciones, capacidades y distribuciones de personas para un viaje especificado organizada por provincias, lo cual funciona a la vez como reporte para posteriores consultas; a partir de aquí, el administrador tiene varias opciones o caminos para realizar las distribuciones.

2.4.1 Distribución automática

Consiste en seleccionar una provincia para internamente ejecutar procedimientos almacenados y funciones para asignar cada reservación hecha para cada municipio de dicha provincia a un asiento libre de cada transporte que tenga dicho destino. Se pueden especificar criterios para la prioridad del algoritmo como **orden de reservaciones** (el primero que reservó es el primero que se distribuye), **área** (se distribuyen por orden de áreas), **año** (se distribuyen por orden de los años de los estudiantes).

2.4.2 Distribución por municipios

Dada una provincia el sistema despliega un listado de todos los municipios con las cantidades de reservaciones, capacidades, distribuciones y las opciones de distribuir a los reservados para un municipio especificado en un transporte.

2.4.3 Distribución por transporte o puntuales

De una provincia especificada se puede acceder a la información de cada uno de sus transportes, permitiendo además buscar un usuario reservado no distribuido y “sentarlo” en un transporte seleccionado. El proceso de distribución para un viaje debe ser iniciado por un administrador luego de que pase el período de reservación. Los datos de la distribución de cada reservación de los usuarios serán mostrados por el sistema una vez concluida y estará siempre disponible.

2.5 Consulta de Historiales

Cada acción que implique interacción con la base de datos es registrada. Gracias a funcionalidades administrativas una persona designada puede tener acceso a ver reportes mediante parámetros de búsquedas que informen qué persona creó, modificó o eliminó registros junto a la fecha y hora de la

acción. Se puede consultar todas las veces que un determinado usuario reservó y/o canceló alguna reservación en cualquiera de las modalidades, así como las incidencias de bloqueo que ha tenido una persona desde su primera interacción con el sistema.

2.6 Generación de Reportes

Este conjunto de funcionalidades está muy vinculado con los procesos de distribución a la hora de realizar cálculos de las capacidades de transportes y reservaciones; más allá de esto, se pueden consultar viajes de cualquiera de las modalidades para obtener los detalles de las reservaciones y personas reservadas, distribuidas y limitadas según un criterio de agrupamiento. Los listados generados pueden ser llevados a formato PDF y también está garantizada la generación de boletines tanto para la Transportación Masiva como la de Trabajadores Internos.

2.7 Envíos de Correos

Con el objetivo de mantener al usuario informado acerca del estado de sus reservaciones, se ha desarrollado un módulo con funcionalidades que utilizan una encapsulación que hace Symfony sobre la función **mail()** que ofrece PHP. Desde distintos "lugares" de la aplicación se envían mensajes con contenido muy útil al viajero. Cuando una persona reserva, inmediatamente recibe una notificación con la afirmación de que la reservación se ha creado correctamente; lo mismo sucede cuando manualmente se cancela una solicitud de viaje. En la Transportación Masiva y en la de Trabajadores Internos existe un mecanismo activado por un administrador que automáticamente cada persona reservada y distribuida recibe los detalles de la reservación y del transporte. En esta última modalidad se cuenta además con envíos de mensajes recordatorios a personas que no han confirmado sus reservaciones.

El administrador también tiene disponible una interfaz en la cual puede redactar un mensaje y enviárselo a todas las personas reservadas distribuidas para las modalidades excepto la Transportación de Fin de Semana a modo de información; esta funcionalidad es muy útil cuando existen cambios de última hora en los horarios de salida o en los transportes.

2.8 Consultar Información

En el sistema, cualquier usuario puede introducir parámetros para obtener información acerca del estado de las reservaciones de cualquier persona en cualquier momento. Especificando valores como usuario,

carné de identidad o solapín, se obtienen las reservaciones de la persona consultada agrupadas por modalidad, además, en caso de las Transportaciones Nacionales se muestran también las distribuciones.

El Sistema Integrado de Transportación también dispone de un buscador de personas del dominio UCI aprovechando el consumo que hace la aplicación de los servicios web y el LDAP de la UCI que brindan toda la información de los usuarios.

2.9 Servicios Web Brindados

El Sistema Integrado de Transportación ofrece información acerca de las reservaciones manipuladas a través de la implementación estándar que hace PEAR de SOAP, es decir, mostrando como interfaz documentos WSDL, es posible hacer llamados de métodos publicados por este sistemas desde otras aplicaciones mediante las clases SoapClient y SoapServer de la extensión SOAP de PHP.

Para la Transportación Estudiantil de Fin de Semana es posible consultar información acerca de las reservaciones y familiares asociados a un usuario específico; conocer los viajes y rutas que hay disponibles y, teniendo un nivel de acceso específico, realizar y/o cancelar reservaciones.

Para la Transportación Masiva se ofrecen los datos específicos de los viajes que tiene disponible cada usuario y, una vez realizadas las reservaciones y distribuciones es posible además consultar los datos de los transportes en los cuales están asociadas cada una de las solicitudes.

2.10 Funcionamiento general del Symfony

2.10.1 Estructura de directorios del framework Symfony

Symfony utiliza una estructura jerárquica en sus directorios que, unido a las estructura del MVC, se ha implementado un subsistema de configuración en cascada, lo cual facilita el establecimiento de valores globales a nivel de proyecto, aplicación y módulos, es decir, un mismo valor puede ser redefinido en diferentes niveles y en diferentes entornos para luego ser utilizado de varias formas.

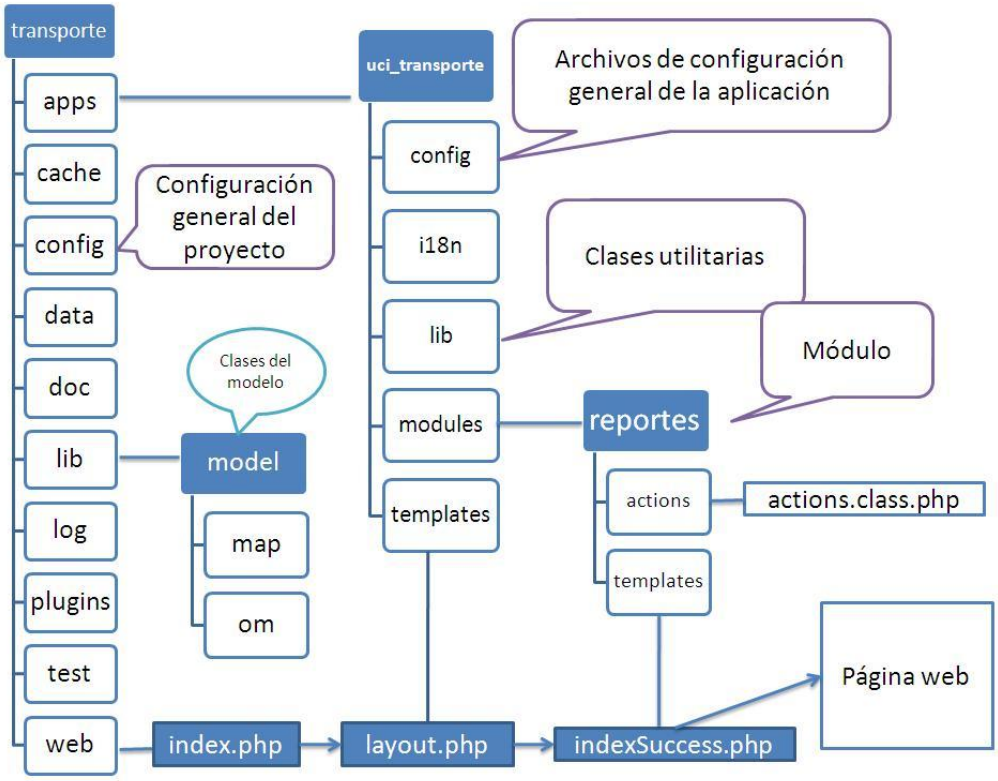


Figura 2 Estructura general de un proyecto Symfony

Un **proyecto** está conformado por las librerías del núcleo del framework y opciones de configuración, al cual se le pueden adicionar varias **aplicaciones** que heredan las mismas opciones configurables y a la vez personalizables (conexión a bases de datos, variables globales, constantes, etc.). Cada aplicación está compuesta por **módulos,** donde residen las clases controladoras y las vistas o plantillas que utilizan el patrón Decorator para ser mostradas junto al **layout** y mostrar así el resultado final de una petición. Los módulos también contienen, gracias al lenguaje YAML, opciones para configurar sus comportamientos. Unido a todo esto, existen clases en el núcleo del framework para cargar automáticamente librerías predefinidas por el usuario y así integrarlas al proyecto.

En un proyecto definido se establece la conexión a la base de datos y se generan las clases de la capa del modelo según el ORM a utilizar (en este caso **Propel**). Estas clases que representan una abstracción orientada a objetos de la base de datos son las encargadas de realizar todo tipo de consultas de forma

bien optimizada. A nivel de aplicación, las clases controladoras de cada módulo y las funciones definidas por el usuario son las que interactúan con el modelo. Cada **action** (executeIndex, executePortada, etc.) tiene asociada una o más vistas (indexSuccess.php, portadaSuccess.php), que son las que se muestran en el navegador decoradas por la plantilla general o **layout.**

2.11 Descripciones de clases y funcionalidades

2.11.1 Clases del modelo

Propel, según el esquema del modelo de la base de datos existente, genera e implementa clases con las funcionalidades necesarias que constituyen la abstracción del modelo relacional. Se generan 5 clases por cada tabla de la base de datos teniendo disponible 2 para ser personalizadas por el programador.

Nombre: TbDestudiante	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Permite cambiar el estado del objeto PersonaUci asociado a desactivado en la base de datos.
Nombre:	FamiliaresVisitados
Descripción:	Devuelve un arreglo con todos los familiares que tiene asociado el estudiante
Nombre:	ReservacionesTe
Descripción:	Devuelve las reservaciones de la Transportación de Fin de semana de un estudiante dado un viaje específico.
Nombre:	AgregarUnFamiliar
Descripción:	Permite agregarle un familiar a un estudiante y si el carnet de identidad ya existe entonces devuelve el identificador.

Tabla 1 Descripción de la clase entidad TbDestudiante

Nombre: TbDestudiantePeer
Entidad
Operaciones

Nombre:	CrearPersonaUci
Descripción:	De acuerdo al modelo de datos se crea la PersonaUci correspondiente al estudiante.

Tabla 2 Descripción de la clase entidad TbDestudiantePeer

Nombre: TbDfamiliar	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Desactiva el objeto persona correspondiente.

Tabla 3 Descripción de la clase entidad TbDfamiliar

Nombre: TbDpersona	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Desactiva la persona en la base de datos y desactiva todas sus reservaciones.
Nombre:	ObtenerUltimaReservacion
Descripción:	Devuelve la última reservación que hizo la persona en cualquiera de las modalidades.
Nombre:	EstaReservadoIda
Descripción:	Retorna true si tiene al menos una reservación dado una aplicación en un viaje de ida.
Nombre:	EstaReservadoRegreso
Descripción:	Retorna verdadero si tiene al menos una reservación dado una aplicación en un viaje de regreso.
Nombre:	MiViaje
Descripción:	Devuelve, dada una aplicación, el viaje de ida o regreso para el cual reservó.

Tabla 4 Descripción de la clase entidad TbDpersona

Nombre: TbDpersonaPeer	
Entidad	
Operaciones	
Nombre:	ExistePersona
Descripción:	Comprueba, según el carnet de identidad, si la persona está registrada en la base de

	datos.

Tabla 5 Descripción de la clase entidad TbDpersonaPeer

Nombre: TbDpersonaUci	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Desactiva en la base de datos el objeto persona correspondiente.
Nombre:	ExisteEstaPersona
Descripción:	Busca en la base de datos dado un parámetro de búsqueda y retorna true si está registrada.

Tabla 6 Descripción de la clase entidad TbDpersonaUci

Nombre: TbDpersonaUciPeer	
Entidad	
Operaciones	
Nombre:	ExistePersonaDadoIDEXP
Descripción:	Retorna true si la persona está registrada en la base de datos dado el id_expediente
Nombre:	ExistePersonaDadoUSUA
Descripción:	Retorna true si la persona está registrada en la base de datos dado el usuario.
Nombre:	CrearPersonaUci
Descripción:	Permite crear el objeto PersonaUci.

Tabla 7 Descripción de la clase entidad TbDpersonaUciPeer

Nombre: TbDreservacion	
Entidad	
Operaciones	
Nombre:	BloqueAQuePertenece
Descripción:	Si la reservación es de la Transportación Nacional entonces devuelve el bloque

Tabla 8 Descripción de la clase entidad TbDreservacion

Nombre: TbDreservacionTe
Entidad

Operaciones	
Nombre:	Desactivar
Descripción:	Permite cambiar el estado a desactivado la reservación.

Tabla 9 Descripción de la clase entidad TbDreservacionTe

Nombre: TbDreservacionTnl	
Entidad	
Operaciones	
Nombre:	EsDePersonaUCI
Descripción:	Permite saber si la reservación es de un familiar o de una persona UCI.
Nombre:	EstaDistribuido
Descripción:	Comprueba si la reservación ha sido asignada a un transporte.
Nombre:	Desactivar
Descripción:	Permite cancelar la reservación y además elimina la capacidad ocupada en un transporte.
Nombre:	EnEspera
Descripción:	Para la Transportación Semestral de Trabajadores verifica si el último estado está en espera.
Nombre:	EsConfirmada
Descripción:	Para la Transportación Semestral de Trabajadores verifica si el último estado está en confirmada.
Nombre:	EsReservada
Descripción:	Para la Transportación Semestral de Trabajadores verifica si el último estado está en habilitada.
Nombre:	DatosDistribucion
Descripción:	Devuelve los datos del transporte en el cual está distribuida la reservación

Tabla 10 Descripción de la clase entidad TbDreservacionTnl

Nombre: TbDviaje	
Entidad	
Operaciones	
Nombre:	DistribuirMunicipioEn
Descripción:	Distribuir todas las personas que reservaron para un municipio dentro de un transporte

	conociendo el id del municipio y el id del transporte.
Nombre:	DistribuirReservacionEn
Descripción:	Distribuir una persona en un transporte dado el id de la reservación y el id del transporte.
Nombre:	Desactivar
Descripción:	Permite cambiar el estado de un viaje y las reservaciones y transporte.
Nombre:	CancelarDistribucionAutomatica
Descripción:	Permite cancelar la distribución automática de una provincia.
Nombre:	CancelarDistribucionMunicipio
Descripción:	Permite cancelar la distribución hecha para un municipio.
Nombre:	CancelarPuntual
Descripción:	Permite cancelar la distribución de una reservación para un transporte.
Nombre:	EstaReservada
Descripción:	Verificar si una persona está reservada.

Tabla 11 Descripción de la clase entidad TbDviaje

Nombre: TbDviajePeer	
Entidad	
Operaciones	
Nombre:	CancelarReservacion
Descripción:	Permite cancelar una reservación hecha para un viaje.
Nombre:	DeshabilitarVarios
Descripción:	Permite deshabilitar varios viajes a la vez dado un arreglo de identificadores.

Tabla 12 Descripción de la clase entidad TbDviajePeer

Nombre: TbDviajeTe	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Permite desactivar un viaje de la Transportación Estudiantil de Fin de Semana
Nombre:	EstaReservada
Descripción:	Verifica si una persona está reservada para este tipo de viaje.
Nombre:	ModificarReservacion

Descripción:	Permite modificar una reservación de una persona de la Transportación de Fin de Semana.
Nombre:	ReservarFinSemana
Descripción:	Permite reservar una persona de la Transportación de Fin de Semana.

Tabla 13 Descripción de la clase entidad TbDviajeTe

Nombre: TbDviajeTePeer	
Entidad	
Operaciones	
Nombre:	ViajesProximo
Descripción:	Retorna el próximo viaje dado el tipo de viaje.

Tabla 14 Descripción de la clase entidad TbDviajeTePeer

Nombre: TbDviajeTn	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Permite desactivar un viaje de la Transportación Nacional.
Nombre:	cantReservacionesArea
Descripción:	Devuelve la cantidad de reservaciones que hay para un área UCI.
Nombre:	__toString
Descripción:	Devuelve los datos del viaje en forma de cadena.
Nombre:	PersonasSentadas
Descripción:	Devuelve las personas sentadas que hay en un transporte.
Nombre:	ProvinciasViaje
Descripción:	Devuelve las provincias que están en las reglas del viaje.
Nombre:	Boletines
Descripción:	Devuelve todas las distribuciones dado un transporte.
Nombre:	CapacidadProvincia
Descripción:	Devuelve la capacidad de una provincia dada.
Nombre:	DisponiblesProvincia
Descripción:	Devuelve los asientos disponibles para una provincia dada.

Nombre:	CantidadReservaciones
Descripción:	Devuelve la cantidad de reservaciones hechas para un municipio dado.
Nombre:	CantidadDistribuidos
Descripción:	Calcula las personas distribuidas en una provincia.
Nombre:	CantidadFamDistribuidos
Descripción:	Calcula los Familiares distribuidos en una provincia.
Nombre:	EstaReservada
Descripción:	Devuelve verdadero si la persona está reservada para la Transportación Nacional.
Nombre:	ReservarTNEstudiante
Descripción:	Permite reservarle un transporte nacional a un estudiante.
Nombre:	ReservarTnFamiliar
Descripción:	Permite reservarle un transporte nacional a un familiar.
Nombre:	ReservarTNProfesor
Descripción:	Permite reservarle un transporte nacional a un trabajador.
Nombre:	CantidadFamiliares
Descripción:	Devuelve la cantidad de reservaciones de familiares por provincia.
Nombre:	CantidadReservacionesMunicipio
Descripción:	Devuelve la cantidad de reservaciones que hay para un determinado municipio.
Nombre:	CantidadReservacionesDistribuidasMunicipio
Descripción:	Devuelve la cantidad de reservaciones que están distribuidas para un determinado municipio.
Nombre:	FamiliaresReservadosMunicipio
Descripción:	Devuelve las reservaciones de los familiares para un determinado municipio.
Nombre:	CantidadFamiliaresDistribuidasMunicipio
Descripción:	Devuelve las reservaciones de los familiares que están distribuidas para un determinado municipio.
Nombre:	FamiliaresReservados
Descripción:	Devuelve las personas que están reservados para un determinado municipio.
Nombre:	Distribuir
Descripción:	Ejecuta los procedimientos almacenados que hacen los procesos de distribución automática.
Nombre:	DistribuirAno

Descripción:	Ejecuta los procedimientos almacenados que hacen los procesos de distribución por año.
Nombre:	DistribuirArea
Descripción:	Ejecuta los procedimientos almacenados que hacen los procesos de distribución por área.
Nombre:	CancelarDistribucionTranporte
Descripción:	Cancela las distribuciones hechas para un transporte.
Nombre:	CancelarDistribucionMunicipio
Descripción:	Cancela las distribuciones hechas para un municipio.

Tabla 15 Descripción de la clase entidad TbDviajeTn

Nombre: TbDviajeTnl	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Permite desactivar un viaje de las transportaciones nacionales.
Nombre:	ObtenerTransportes
Descripción:	Devuelve todos los transportes dado una provincia.
Nombre:	EstaEnPeriodoReservacion
Descripción:	Saber si un viaje está en fecha de reservación.

Tabla 16 Descripción de la clase entidad TbDviajeTnl

Nombre: TbDviajeTnp	
Entidad	
Operaciones	
Nombre:	Desactivar
Descripción:	Permite desactivar un viaje de la Transportación de Trabajadores Internos.
Nombre:	ReservarTNProfesor
Descripción:	Permite reservar a un trabajador.
Nombre:	ListaEspera
Descripción:	Pone en lista de espera una reservación.
Nombre:	EsPrimerSemestre
Descripción:	Chequea a qué semestre pertenece el viaje.
Nombre:	ReservarTnFamiliar

Descripción:	Permite reservarle un familiar a una persona UCI.
Nombre:	PersonasSentadas
Descripción:	Devuelve las personas sentadas en un transporte.
Nombre:	PersonasSentadasSinConfirmar
Descripción:	Devuelve las personas sentadas en un transporte que no tienen la reservación confirmada.

Tabla 17 Descripción de la clase entidad TbDviajeTnp

Nombre: TbRdistribucionAsientoC	
Entidad	
Operaciones	
Nombre:	EstaPersonaEstaDistribuida
Descripción:	Permite saber si una persona está distribuida en un tren para obtener los datos.

Tabla 18 Descripción de la clase entidad TbRdistribucionAsientoC

Nombre: TbRdistribucionAsientoT	
Entidad	
Operaciones	
Nombre:	EstaPersonaEstaDistribuida
Descripción:	Permite saber si una persona está distribuida en un ómnibus para obtener los datos.

Tabla 19 Descripción de la clase entidad TbRdistribucionAsientoT

2.11.2 Clases controladoras o acciones

Estas clases, predefinidas y autogeneradas por el framework Symfony, proveen el esqueleto para construir las funcionalidades da la capa de control de la arquitectura utilizada, es decir, manipulan los datos que provienen de las Vistas y/o de las clases del modelo de datos o clases utilitarias. Cada una de las operaciones tiene asociada una o varias vistas.

2.11.2.1 Consultar información

Nombre: consultar_infActions
Controladora
Operaciones

Nombre:	executeIndex
Descripción:	Busca en la base de datos, según un criterio de búsqueda, una persona UCI y prepara sus datos para enviárselos a la plantilla IndexSuccess.php.
Nombre:	executePmasivo
Descripción:	Dado una reservación, prepara los datos de la persona junto a sus reservaciones y distribuciones de la Transportación Nacional para ser enviados a pmasivoSuccess.php.
Nombre:	executePprofesores
Descripción:	Dado una reservación, prepara los datos de la persona junto a sus reservaciones y distribuciones de la Transportación de Trabajadores Internos para ser enviados a pprofesoresSuccess.php.
Nombre:	executePestudiantil
Descripción:	Dado una reservación, prepara los datos de la persona junto a sus reservaciones y distribuciones de la Transportación Estudiantil de Fin de Semana para ser enviados a pestudiantilSuccess.php.
Nombre:	executePortada
Descripción:	Permite editar la información que se mostrará en la portada del sitio.

Tabla 20 Descripción de la clase controladora consultar_infActions

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra los datos de una persona y, depende el tipo de persona, se muestran vínculos para acceder a ver información sobre las reservaciones.

Tabla 21 Descripción de la clase interfaz indexSuccess.php

Archivo: pestudiantilSuccess.php	
Interfaz	
Descripción:	Muestra los datos de una persona y las reservaciones que tiene para la Transportación De Fin de Semana.

Tabla 22 Descripción de la clase interfaz pestudiantilSuccess.php

Archivo: pmasivoSuccess.php	
Interfaz	
Descripción:	Muestra los datos de una persona y las reservaciones que tiene para la Transportación

	Nacional.

Tabla 23 Descripción de la clase interfaz pmasivoSuccess.php

Archivo: pprofesoresSuccess.php	
Interfaz	
Descripción:	Muestra los datos de una persona y las reservaciones que tiene para la Transportación Semestral de Trabajadores Internos.

Tabla 24 Descripción de la clase interfaz pprofesoresSuccess.php

Archivo: portadaSuccess.php	
Interfaz	
Descripción:	Muestra un formulario para editar la información que se muestra en la portada de la aplicación.

Tabla 25 Descripción de la clase interfaz portadaSuccess.php

2.11.2.2 Correo

Nombre: correoActions	
Controladora	
Operaciones	
Nombre:	executeIndex
Descripción:	Extrae todos los viajes habilitados de la Transportación Nacional y de Trabajadores Internos para enviarlos a la plantilla IndexSuccess.php.
Nombre:	executeConfirma
Descripción:	Dado un viaje se consulta a la base de datos las personas distribuidas sin confirmar en la Transportación de Trabajadores Internos y se envía a cada una un correo.
Nombre:	executeDetalles
Descripción:	Enviar correos para las personas que ya confirmaron su reservación con los detalles de estas.
Nombre:	executePersonalizado
Descripción:	Dado una reservación, prepara los datos de la persona junto a sus reservaciones y distribuciones de la Transportación Estudiantil de Fin de Semana para ser enviados a

	pestudiantilSuccess.php.
Nombre:	executeQuejas
Descripción:	Obtiene las aplicaciones registradas en la base de datos para mostrarlas en la plantilla quejasSuccess.php.
Nombre:	handleErrorEnviarQueja()
Descripción:	Redirecciona la petición para el formulario de enviar quejas en caso de algún error.
Nombre:	executeEnviarQueja()
Descripción:	Construye y envía un correo con los datos entrados por el usuario.

Tabla 26 Descripción de la clase controladora correoActions

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra un formulario que contiene las aplicaciones o modalidades de reservación, los viajes a escoger y el campo de texto a llenar para enviar un correo con texto personalizado a las personas reservadas para un viaje.

Tabla 27 Descripción de la clase interfaz indexSuccess.php

Archivo: detallesError.php	
Interfaz	
Descripción:	Muestra un mensaje si no se han enviado correctamente los correos.

Tabla 28 Descripción de la clase interfaz detallesError.php

Archivo: detallesSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje si se han enviado correctamente los correos.

Tabla 29 Descripción de la clase interfaz detallesSuccess.php

Archivo: enviadoSuccess.php	
Interfaz	
Descripción:	Es una plantilla genérica que puede ser utilizada por varias acciones. Muestra un mensaje si se han enviado correctamente los correos.

Tabla 30 Descripción de la clase interfaz enviadoSuccess.php

Archivo: enviarQuejaSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje si se ha enviado correctamente el correo de las quejas.

Tabla 31 Descripción de la clase interfaz enviarQuejaSuccess.php

Archivo: quejaSuccess.php	
Interfaz	
Descripción:	Muestra un formulario con los campos usuario, aplicación y cuerpo del mensaje para ser enviados en modo de correo.

Tabla 32 Descripción de la clase interfaz quejaSuccess.php

Nombre: scCorreoMasivo	
Controladora	
Operaciones	
Nombre:	EnviarAConfirmados
Descripción:	Envia un correo masivo a todas las personas que tienen confirmada su reservación y en el caso de la Transportación Nacional a los que están distribuidos.
Nombre:	EnviarParaQueConfirmen
Descripción:	Permite enviar correos a todas las personas que no han confirmado su reservación en la Transportación de Trabajadores Internos.
Nombre:	EnviarAReservado
Descripción:	Permite enviar un correo a las personas que reservaron en un determinado momento.
Nombre:	EnviarACancelado
Descripción:	Permite enviar un correo a las personas que cancelaron su reservación en un determinado momento.
Nombre:	EnviarPersonalizado
Descripción:	Permite enviar un correo con texto personalizado.
Nombre:	EnviarQueja
Descripción:	Permite enviar un correo con texto personalizado.

Tabla 33 Descripción de la clase controladora scCorreoMasivo

Archivo: enviarQueja.yml

Utilidad	
Descripción:	Especifica en YAML cómo los campos del formulario de enviar quejas deben ser validados.

Tabla 34 Descripción de la clase utilidad enviarQueja.yml

2.11.2.3 Datos de usuarios

Nombre: datos_usuarioActions	
Controladora	
Operaciones	
Nombre:	executeIndex
Descripción:	Extrae de los servicios web disponibles un arreglo de provincias y de áreas UCI para ser enviados a su plantilla correspondiente.
Nombre:	executeBuscarBasico
Descripción:	Mediante parámetros de búsqueda, se consultan a los servicios web disponibles para obtener datos de una persona UCI y ser enviados a la plantilla correspondiente para visualizar.
Nombre:	executeMunicipios
Descripción:	Prepara, consultando los servicios web disponibles, un arreglo de municipios para ser enviados a la plantilla correspondiente.

Tabla 35 Descripción de la clase controladora datos_usuarioActions

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra un formulario con un campo para introducir un criterio de búsqueda de personas en el dominio UCI.

Tabla 36 Descripción de la clase interfaz indexSuccess.php

Archivo: buscarBasicoSuccess.php	
Interfaz	
Descripción:	Muestra, mediante una petición AJAX, los datos resultantes de una búsqueda de personas.

Tabla 37 Descripción de la clase interfaz buscarBasicoSuccess.php

Archivo: buscarBasicoError.php	
Interfaz	
Descripción:	Muestra, mediante una petición AJAX, un mensaje de error si no se efectuó correctamente la búsqueda de personas en **executeBuscarBasico()**.

Tabla 38 Descripción de la clase interfaz buscarBasicoError.php

Archivo: municipiosSuccess.php	
Interfaz	
Descripción:	Muestra, mediante una petición AJAX, un campo de tipo *select* que contiene los municipios a escoger dado una provincia.

Tabla 39 Descripción de la clase interfaz municipiosSuccess.php

2.11.2.4 Distribuir

Nombre: distribuirActions	
Controladora	
Operaciones	
Nombre:	executeIndex
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se crean arreglos que contienen información de las personas reservadas y distribuidas en un viaje en específico de la Transportación Nacional.
Nombre:	executeMunicipios
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se crean arreglos que contienen información de las personas reservadas y distribuidas en un viaje en específico de la Transportación Nacional para una provincia determinada.
Nombre:	executeTransportes
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se crean arreglos que contienen información de las personas reservadas y distribuidas en un viaje en específico de la Transportación Nacional para un transporte determinado.
Nombre:	executeDistribuciones

Descripción:	Envía a la plantilla correspondiente los datos que vienen de la petición.
Nombre:	executeOpcancelar
Descripción:	Envía a la plantilla correspondiente los datos que vienen de la petición.
Nombre:	executeTransportedisp
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se crean arreglos que contienen los transportes disponibles.
Nombre:	executeCancelar
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se llama a la funcionalidad que permite cancelar la distribución automática de las reservaciones.
Nombre:	executeAutomatica
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se llama a la funcionalidad que permite efectuar la distribución automática de las reservaciones.
Nombre:	executeFacultad
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se llama a la funcionalidad que permite efectuar la distribución por facultad de las reservaciones.
Nombre:	executeAno
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se llama a la funcionalidad que permite efectuar la distribución por año de las reservaciones.
Nombre:	executePersonalizada
Descripción:	Envía a la plantilla correspondiente los datos que vienen de la petición.
Nombre:	executePuntual
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se llama a la funcionalidad que permite efectuar la distribución puntual de una reservación.
Nombre:	executeBuscar
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se preparan los datos de una persona y sus reservaciones hechas para un viaje para ser mostrados en la plantilla correspondiente.

Nombre:	executeDistribuirmunicipio
Descripción:	Utilizando variables que provienen de una petición y consultando los servicios web disponibles, se llama a la funcionalidad que permite efectuar la distribución por municipios de las reservaciones.
Nombre:	executeCancelarpun
Descripción:	Utilizando variables que provienen de una petición se llama a la funcionalidad que permite cancelar la distribución puntual de una reservación.
Nombre:	executeCancelardistransporte
Descripción:	Utilizando variables que provienen de una petición se llama a la funcionalidad que permite cancelar la distribución puntual de un transporte.
Nombre:	executeCancelardismunicipio
Descripción:	Utilizando variables que provienen de una petición se llama a la funcionalidad que permite cancelar la distribución puntual de un municipio.

Tabla 40 Descripción de la clase controladora distribuirActions

Archivo: buscarSuccess.php	
Interfaz	
Descripción:	Muestra, mediante una petición AJAX, los datos de una persona y las opciones de distribuir y/o cancelar su distribución.

Tabla 41 Descripción de la clase interfaz buscarSuccess.php

Archivo: distribucionesSuccess.php	
Interfaz	
Descripción:	Muestra, mediante una petición AJAX, los datos de una persona y las opciones para efectuar las distribuciones generales.

Tabla 42 Descripción de la clase interfaz distribucionesSuccess.php

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra la interfaz principal de las distribuciones con las cantidades de reservaciones, capacidades y distribuciones de personas y familiares además de las distintas vías de realizar las distribuciones.

Tabla 43 Descripción de la clase interfaz indexSuccess.php

Archivo: municipiosSuccess.php	
Interfaz	
Descripción:	Muestra la interfaz que contiene las opciones e información de las distribuciones con las cantidades de reservaciones y capacidades para una provincia dada.

Tabla 44 Descripción de la clase interfaz municipiosSuccess.php

Archivo: opcancelarSuccess.php	
Interfaz	
Descripción:	Muestra la opción de cancelar una distribución automática.

Tabla 45 Descripción de la clase interfaz opcancelarSuccess.php

Archivo: personalizadaSuccess.php	
Interfaz	
Descripción:	Muestra un formulario asíncrono con los datos necesarios para buscar un usuario para su distribución.

Tabla 46 Descripción de la clase interfaz personalizadaSuccess.php

Archivo: transportedispSuccess.php	
Interfaz	
Descripción:	Muestra un formulario asíncrono con los datos necesarios para efectuar una distribución por transporte.

Tabla 47 Descripción de la clase interfaz transportedispSuccess.php

Archivo: transportesSuccess.php	
Interfaz	
Descripción:	Muestra un formulario asíncrono con los datos de los transportes en cuanto a las distribuciones (asientos disponibles, capacidad, asientos ocupados) y la opción de cancelar distribuciones.

Tabla 48 Descripción de la clase interfaz transportesSuccess.php

2.11.2.5 Error

Nombre: errorActions	
Controladora	
Operaciones	
Nombre:	executeIndex
Descripción:	Ejecuta la vista correspondiente cuando ocurre un error general.
Nombre:	executeBd
Descripción:	Ejecuta la vista correspondiente cuando ocurre un error al solicitar datos de la base de datos.
Nombre:	executeDeshabilitado
Descripción:	Ejecuta la vista correspondiente cuando ocurre una solicitud a un módulo que no está habilitado.
Nombre:	executeError404
Descripción:	Ejecuta la vista correspondiente cuando ocurre una solicitud a una página que no existe.
Nombre:	executeWebService
Descripción:	Ejecuta la vista correspondiente cuando ocurre un error al solicitar datos de los servicios web.

Tabla 49 Descripción de la clase controladora errorActions

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje cuando ocurre un error general.

Tabla 50 Descripción de la clase interfaz indexSuccess.php

Archivo: bdSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje cuando ocurre un error al solicitar datos de la base de datos.

Tabla 51 Descripción de la clase interfaz bdSuccess.php

Archivo: deshabilitadoSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje cuando ocurre una solicitud a un módulo que no está habilitado.

Tabla 52 Descripción de la clase interfaz deshabilitadoSuccess.php

Archivo: error404Success.php	
Interfaz	
Descripción:	Muestra un mensaje cuando ocurre una solicitud a una página que no existe.

Tabla 53 Descripción de la clase interfaz error404Success.php

Archivo: webServiceSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje cuando ocurre un error al solicitar datos de los servicios web.

Tabla 54 Descripción de la clase webServiceSuccess.php

2.11.2.6 Gestionar viajero

Nombre: gest_viajeroActions	
Controladora	
Operaciones	
Nombre:	preExecute
Descripción:	Chequea que esté habilitada la opción de reservar antes de ejecutar cualquier opción.
Nombre:	executeTrabajadores
Descripción:	Prepara un arreglo de transportes disponibles para ser mostrados en la plantilla correspondiente.
Nombre:	executeReservartnp
Descripción:	Permite hacerle una reservación al trabajador para un viaje.
Nombre:	executeListaespera
Descripción:	Permite poner una reservación en modo de espera.
Nombre:	executeBuscar
Descripción:	Dado un parámetro de entrada permite consultar los servicios web disponibles para obtener datos de una persona y, además, se consulta a la base de datos para obtener los viajes y/o reservaciones de la persona buscada.
Nombre:	executeReservar
Descripción:	Permite realizar una reservación a una persona UCI para un viaje.

Nombre:	executeCancelar
Descripción:	Permite cancelar una reservación de una persona UCI.
Nombre:	executeMunicipios
Descripción:	Muestra, en su plantilla correspondiente, los municipios de una provincia mediante una llamada Ajax.
Nombre:	executeBuscartn
Descripción:	Consulta a la base de datos para obtener los viajes habilitados de la Transportación Nacional.
Nombre:	executeFamiliar
Descripción:	Prepara los datos obtenidos de los servicios web para ser utilizados en su plantilla correspondiente.
Nombre:	handleErrorReservarfamiliar
Descripción:	Permite validar formulario de reservar familiar.
Nombre:	handleErrorReservar
Descripción:	Permite validar formulario de reservar.
Nombre:	executeReservarfamiliar
Descripción:	Permite crearle una reservación a una persona que no está registrada en el dominio UCI.

Tabla 55 Descripción de la clase controladora gest_viajeroActions

Archivo: _enEspera.php	
Interfaz	
Descripción:	Elemento parcial para mostrar reservaciones en modo de espera.

Tabla 56 Descripción de la clase interfaz enEspera.php

Archivo: _familiarReservado.php	
Interfaz	
Descripción:	Elemento parcial para mostrar las reservaciones de familiares de una persona UCI.

Tabla 57 Descripción de la clase interfaz familiarReservado.php

Archivo: _reservadosIda.php	
Interfaz	
Descripción:	Muestra las reservaciones realizadas para un viaje de ida.

Tabla 58 Descripción de la clase interfaz reservadosIda.php

Archivo: _reservadosRegreso.php	
Interfaz	
Descripción:	Muestra las reservaciones realizadas para un viaje de regreso.

Tabla 59 Descripción de la clase reservadosRegreso.php

Archivo: _reservarIda.php	
Interfaz	
Descripción:	Muestra los viajes disponibles de ida con las opciones de reservar y reservar familiar.

Tabla 60 Descripción de la clase interfaz reservarIda.php

Archivo: _reservarRegreso.php	
Interfaz	
Descripción:	Muestra los viajes disponibles de regreso con las opciones de reservar y reservar familiar.

Tabla 61 Descripción de la clase interfaz reservarRegreso.php

Archivo: buscarSuccess.php	
Interfaz	
Descripción:	Muestra los datos de la persona buscada, los viajes disponibles y/o reservaciones realizadas.

Tabla 62 Descripción de la clase interfaz buscarSuccess.php

Archivo: buscartnSuccess.php	
Interfaz	
Descripción:	Muestra un contenedor HTML para mostrar, de forma AJAX, otras plantillas.

Tabla 63 Descripción de la clase interfaz buscartnSuccess.php

Archivo: familiarSuccess.php	
Interfaz	
Descripción:	Muestra un formulario para llenar los datos del familiar que se va a reservar.

Tabla 64 Descripción de la clase interfaz familiarSuccess.php

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra un contenedor HTML para mostrar, de forma AJAX, otras plantillas.

Tabla 65 Descripción de la clase interfaz indexSuccess.php

Archivo: listaesperaSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje al final de una llamada AJAX.

Tabla 66 Descripción de la clase interfaz listaesperaSuccess.php

Archivo: municipiosSuccess.php	
Interfaz	
Descripción:	Muestra un *selec*t con los municipios de una provincia dada mediante una petición AJAX.

Tabla 67 Descripción de la clase interfaz municipiosSuccess.php

Archivo: trabajadoresSuccess.php	
Interfaz	
Descripción:	Para la Transportación Semestral de Trabajadores se muestran los transportes disponibles para reservar.

Tabla 68 Descripción de la clase interfaz trabajadoresSuccess.php

2.11.2.7 Historiales

Nombre: historialesActions	
Controladora	
Operaciones	
Nombre:	executeResultadoTraza
Descripción:	Dado un parámetro de búsqueda se consulta a la base de datos para obtener las acciones realizadas sobre las tablas.
Nombre:	executeResultadoReservaciones
Descripción:	Dado un parámetro de búsqueda que puede ser el usuario de una persona se consulta a la base de datos para obtener las reservaciones realizadas.

Nombre:	executeResultadoUsuarioBloqueado
Descripción:	Dado un parámetro de búsqueda que puede ser el solapín de una persona se consulta a la base de datos para obtener las incidencias y bloqueos que ha tenido.

Tabla 69 Descripción de la clase controladora historialesActions

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Esta plantilla sirve de contenedor para mostrar resultados de acciones de tipo asíncronas que contienen campos para introducir distintos parámetros de búsqueda.

Tabla 70 Descripción de la clase interfaz indexSuccess.php

Archivo: resultadoReservacionesSuccess.php	
Interfaz	
Descripción:	Muestra todas las reservaciones que ha tenido un usuario.

Tabla 71 Descripción de la clase interfaz resultadoReservacionesSuccess.php

Archivo: resultadoTrazaSuccess.php	
Interfaz	
Descripción:	Muestra todas las acciones que ha hecho un usuario en el sistema.

Tabla 72 Descripción de la clase interfaz resultadoTrazaSuccess.php

Archivo: resultadoUsuarioBloqueadoSuccess.php	
Interfaz	
Descripción:	Muestra todas las incidencias de bloqueo que ha hecho un usuario en el sistema.

Tabla 73 Descripción de la clase interfaz resultadoUsuarioBloqueadoSuccess.php

2.11.2.8 Listados

Nombre: listadosActions
Controladora
Operaciones

Nombre:	executeIndex
Descripción:	Se consulta a la base de datos para obtener reservaciones.
Nombre:	executeListaEsperaOmnibus
Descripción:	Dado un transporte se obtienen todas las reservaciones que están en espera.
Nombre:	executePdf
Descripción:	Prepara datos obtenidos de los servicios web para ser mostrados en su plantilla correspondiente.
Nombre:	executeShowmunicipios
Descripción:	Prepara datos obtenidos de los servicios web para ser mostrados en su plantilla correspondiente.
Nombre:	executeShowtransportes
Descripción:	Se consulta a la base de datos para obtener los transportes de un municipio dado.
Nombre:	executeListado
Descripción:	Construye el documento PDF con los listados solicitados por distintos parámetros.

Tabla 74 Descripción de la clase controladora listadosActions

Nombre: Listado	
Utilidad	
Operaciones	
Nombre:	Reservados
Descripción:	Obtiene todas las reservaciones que hay para un transporte específico.
Nombre:	ListaEspera
Descripción:	Obtiene todas las reservaciones en espera para un viaje específico.
Nombre:	ListaEsperaPorOmnibus
Descripción:	Obtiene todas las reservaciones en espera para un transporte específico.
Nombre:	ReservadosParaUnMunicipio
Descripción:	Obtiene todas las reservaciones para un municipio.
Nombre:	ReservadosParaUnaProvincia
Descripción:	Obtiene todas las reservaciones para un municipio.
Nombre:	PDF
Descripción:	Construye el documento PDF con los listados para un transporte específico.

Tabla 75 Descripción de la clase utilidad Listado

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra un listado de usuarios reservados.

Tabla 76 Descripción de la clase interfaz indexSuccess.php

Archivo: listaEsperaOmnibusSuccess.php	
Interfaz	
Descripción:	Muestra un listado de usuarios que tienen reservación en espera para un ómnibus.

Tabla 77 Descripción de la clase interfaz listaEsperaOmnibusSuccess.php

Archivo: pdfSuccess.php	
Interfaz	
Descripción:	Muestra un formulario con campos que son criterios para generar pdfs con listados de usuarios reservados.

Tabla 78 Descripción de la clase interfaz pdfSuccess.php

Archivo: showMunicipiosSuccess.php	
Interfaz	
Descripción:	Muestra un *select* con los municipios del país dado una provincia.

Tabla 79 Descripción de la clase interfaz showMunicipiosSuccess.php

Archivo: showtransportesSuccess.php	
Interfaz	
Descripción:	Muestra un *select* con nombres de transportes.

Tabla 80 Descripción de la clase interfaz showtransportesSuccess.php

2.11.2.9 Reportes

Nombre: reportesActions	
Controladora	
Atributo	**Tipo**
facultades	array

Operaciones	
Nombre:	executePrincipal
Descripción:	Obtiene todos los viajes habilitados en la base de datos para ser mostrados en la vista correspondiente.
Nombre:	executeIndex
Descripción:	Envía a su plantilla correspondiente un identificador de un viaje.
Nombre:	executeRouter
Descripción:	Chequea los identificadores de un viaje y aplicación y los envía a la vista correspondiente.
Nombre:	executeEstudiantes_anno
Descripción:	Dado un viaje y una aplicación, obtiene de la base de datos todas las reservaciones de estudiantes organizadas por año.
Nombre:	executeEstudiantes
Descripción:	Dado un año, se obtiene de la base de datos todas las reservaciones de estudiantes.
Nombre:	executeEstudiantes_facultad
Descripción:	Dado un viaje, obtiene de la base de datos todas las reservaciones de estudiantes organizadas por área.
Nombre:	executeEstudiantes_por_facultad
Descripción:	Dado un viaje, obtiene de la base de datos todas las reservaciones de estudiantes organizadas por área.
Nombre:	executeEstudiantes_por_facultad
Descripción:	Dado un viaje, obtiene de la base de datos todas las reservaciones de estudiantes organizadas por área.
Nombre:	executeEstudiantes_provincia
Descripción:	Dado un viaje, obtiene de la base de datos todas las reservaciones de estudiantes organizadas por provincia.
Nombre:	executeEstudiantes_limitados
Descripción:	Obtiene de la base de datos todas las personas bloqueadas.
Nombre:	executeRutas
Descripción:	Obtiene de la base de datos todas las rutas habilitadas junto a las reservaciones hechas para cada una de ellas.
Nombre:	executeEstudiantes_ruta
Descripción:	Dado un viaje, obtiene de la base de datos todas las reservaciones de estudiantes para

	una ruta.
Nombre:	executeFacultad
Descripción:	Dado un viaje, obtiene de la base de datos todas las reservaciones de estudiantes para una facultad.
Nombre:	executeArea
Descripción:	Dado un viaje, obtiene de la base de datos todas las reservaciones para un área UCI.
Nombre:	executeUsuarios_bloqueados
Descripción:	Obtiene de la base de datos todas las personas bloqueadas en una de las modalidades de reservación dada.
Nombre:	executeFamiliar
Descripción:	Envía identificadores a la plantilla correspondiente.
Nombre:	executeFamiliar_general
Descripción:	Dado un viaje extrae de la base de datos la cantidad de reservaciones de familiares organizadas por provincia.
Nombre:	executeFamiliar_destino
Descripción:	Dada una aplicación extrae de la base de datos todas las reservaciones de familiares organizados por municipios.
Nombre:	executeTrabajadores
Descripción:	Dado un viaje y un área UCI se obtienen los datos de una persona.
Nombre:	executeMunicipios
Descripción:	Dada una aplicación y un viaje, se selecciona la cantidad de reservaciones hechas organizadas por municipio.
Nombre:	executeReservados
Descripción:	Dada una aplicación, un viaje y un municipio, se seleccionan todas las reservaciones hechas para el municipio dado.
Nombre:	executeRfamiliares
Descripción:	Dado un municipio se seleccionan todas las reservaciones hechas para el municipio dado por familiares.
Nombre:	executeArea_usuario
Descripción:	Se filtran los usuarios reservados por área.
Nombre:	executeMunicipio_familiar
Descripción:	Dada una provincia y un viaje se seleccionan todas las reservaciones hechas por

	familiares.
Nombre:	executeOmnibus
Descripción:	Dado una aplicación y un viaje se seleccionan todos los transportes junto a la cantidad de reservaciones y distribuciones de cada uno.
Nombre:	executeDistribuidos
Descripción:	Dado un transporte, se obtienen los datos de todas las personas sentadas.
Nombre:	executeListado
Descripción:	Construye un documento PDF con el listado de personas distribuidas en un transporte.

Tabla 81 Descripción de la clase controladora reportesActions

Nombre: Listado1	
Utilidad	
Operaciones	
Nombre:	Reservados
Descripción:	Obtiene todas las reservaciones que hay para un transporte específico.
Nombre:	PDF
Descripción:	Construye un documento PDF con el listado de personas distribuidas en un transporte.

Tabla 82 Descripción de la clase utilidad Listado1

Archivo: areaSuccess.php	
Interfaz	
Descripción:	Muestra la cantidad de reservaciones para cada área UCI.

Tabla 83 Descripción de la clase utilidad areaSuccess.php

Archivo: area_usuarioSuccess.php	
Interfaz	
Descripción:	Muestra los datos de una persona reservada.

Tabla 84 Descripción de la clase interfaz area_usuarioSuccess.php

Archivo: estudiantesSuccess.php	
Interfaz	
Descripción:	Muestra los datos de un estudiante reservado.

Tabla 85interfaz estudiantesSuccess.php

Archivo: estudiantes_annoSuccess.php	
Interfaz	
Descripción:	Muestra la cantidad de reservaciones de estudiantes organizadas por año.

Tabla 86 Descripción de la clase interfaz estudiantes_annoSuccess.php

Archivo: estudiantes_facultadSuccess.php	
Interfaz	
Descripción:	Muestra la cantidad de reservaciones de estudiantes organizadas por área.

Tabla 87 Descripción de la clase interfaz estudiantes_facultadSuccess.php

Archivo: estudiantes_limitadosSuccess.php	
Interfaz	
Descripción:	Muestra los datos de un estudiante que está bloqueado.

Tabla 88 Descripción de la clase interfaz estudiantes_limitadosSuccess.php

Archivo: estudiantes_por_facultadSuccess.php	
Interfaz	
Descripción:	Muestra los datos de un estudiante.

Tabla 89 Descripción de la clase interfaz estudiantes_por_facultadSuccess.php

Archivo: estudiantes_rutaSuccess.php	
Interfaz	
Descripción:	Muestra los datos de un estudiante.

Tabla 90 Descripción de la clase interfaz estudiantes_rutaSuccess.php

Archivo: facultadSuccess.php	
Interfaz	
Descripción:	Muestra la cantidad de reservaciones por cada facultad.

Tabla 90 Descripción de la clase interfaz facultadSuccess.php

Archivo: facultad_areaSuccess.php
Interfaz

Descripción:	Muestra la cantidad de reservaciones por cada área UCI.

Tabla 91 Descripción de la clase interfaz facultad_areaSuccess.php

Archivo: familiarSuccess.php	
Interfaz	
Descripción:	Muestra la un contenedor para exponer los reportes de familiares reservados y distribuidos.

Tabla 92 Descripción de la clase interfaz facultad_areaSuccess.php

Archivo: familiar_destinoSuccess.php	
Interfaz	
Descripción:	Muestra los datos de familiares reservados y distribuidos con los destinos.

Tabla 93 Descripción de la clase interfaz familiar_destinoSuccess.php

Archivo: familiar_generalSuccess.php	
Interfaz	
Descripción:	Muestra la cantidad de reservaciones de familiares organizadas por provincia.

Tabla 94 Descripción de la clase interfaz familiar_generalSuccess.php

Archivo: municipio_familiarSuccess.php	
Interfaz	
Descripción:	Muestra la cantidad de reservaciones de familiares organizadas por municipio.

Tabla 95 Descripción de la clase interfaz municipio_familiarSuccess.php

Archivo: municipiosSuccess.php	
Interfaz	
Descripción:	Muestra la cantidad de reservaciones organizadas por municipio.

Tabla 96 Descripción de la clase interfaz municipiosSuccess.php

Archivo: omnibusSuccess.php	
Interfaz	
Descripción:	Muestra por cada transporte del sistema, la cantidad de reservaciones, distribuciones y opciones para generar listados.

Tabla 97 Descripción de la clase interfaz omnibusSuccess.php

Archivo: principalSuccess.php	
Interfaz	
Descripción:	Muestra la pantalla principal para acceder a los reportes para una aplicación y un viaje específico.

Tabla 98 Descripción de la clase interfaz principalSuccess.php

Archivo: reservadosSuccess.php	
Interfaz	
Descripción:	Muestra los datos personales de un usuario reservado.

Tabla 100 Descripción de la clase interfaz reservadosSuccess.php

Archivo: rfamiliaresSuccess.php	
Interfaz	
Descripción:	Muestra los datos personales de un familiar reservado o distribuido.

Tabla 101 Descripción de la clase interfaz rfamiliaresSuccess.php

Archivo: routerSuccess.php	
Interfaz	
Descripción:	Muestra la interfaz principal con las opciones de un reporte de un viaje específico.

Tabla 99 Descripción de la clase interfaz routerSuccess.php

Archivo: trabajadoresSuccess.php	
Interfaz	
Descripción:	Muestra los datos personales de un trabajador reservado o distribuido.

Tabla 100 Descripción de la clase interfaz trabajadoresSuccess.php

Archivo: usuarioSuccess.php	
Interfaz	
Descripción:	Muestra los datos personales de una persona UCI reservada o distribuida.

Tabla 101 Descripción de la clase interfaz usuarioSuccess.php

Archivo: usuarios_bloqueadosSuccess.php	
Interfaz	
Descripción:	Muestra los datos personales de una persona UCI limitada para reservar.

Tabla 102 Descripción de la clase interfaz usuarios_bloqueadosSuccess.php

2.11.2.10 Reservaciones Estudiantiles de Fin de Semana

Nombre: res_teActions	
Controladora	
Operaciones	
Nombre:	preExecute
Descripción:	Antes que ocurra cualquier acción, verifica que las reservaciones estén activadas y que el usuario autenticado no esté bloqueado en esta modalidad de reservación.
Nombre:	executeIndex
Descripción:	Chequea que la provincia del usuario autenticado no esté bloqueada. Extrae los viajes disponibles y las reservaciones y los envía a la plantilla correspondiente.
Nombre:	executeRutas
Descripción:	Extrae de la base de datos las rutas habilitadas para ser mostradas en la plantilla correspondiente.
Nombre:	executePuntos
Descripción:	Extrae de la base de datos los puntos habilitados para ser mostrados en la plantilla correspondiente.
Nombre:	handleErrorReservarTe
Descripción:	Se ejecuta cuando los datos del formulario de reservar no están correctos.
Nombre:	executeReservarTe
Descripción:	Para la Transportación de Fin de Semana construye una nueva reservación registrando los datos del usuario y del familiar a visitar; permite modificar el destino de una reservación, cancelarla y envía correos en dependencia de la acción realizada.
Nombre:	executeCancelar
Descripción:	Permite cancelar una reservación.

Tabla 103 Descripción de la clase controladora res_teActions

Nombre: scCorreoFinDeSemana	
Utilidad	
Operaciones	
Nombre:	EnviarAReservado
Descripción:	Construye un mensaje que se enviará a una persona en el momento en que reserva.
Nombre:	EnviarACancelado
Descripción:	Construye un mensaje que se enviará a una persona en el momento en que cancela una reservación.

Tabla 104 Descripción de la clase utilidad scCorreoFinDeSemana

Archivo: _datosBloqueo.php	
Interfaz	
Descripción:	Muestra un texto que contiene los motivos del bloqueo de una provincia.

Tabla 105 Descripción de la clase interfaz datosBloqueo.php

Archivo: _familiaresVisitados.php	
Interfaz	
Descripción:	Muestra todos los datos de los familiares que un estudiante tiene registrado para visitar y un formulario para entrar los datos de uno nuevo.

Tabla 106 f Descripción de la clase interfaz amiliaresVisitados.php

Archivo: _reservadosIdaTe.php	
Interfaz	
Descripción:	Muestra los detalles de la reservación hecha para un viaje de ida.

Tabla 107 Descripción de la clase interfaz reservadosIdaTe.php

Archivo: _reservadosRegresoTe.php	
Interfaz	
Descripción:	Muestra los detalles de la reservación hecha para un viaje de regreso.

Tabla 108 Descripción de la clase interfaz reservadosRegresoTe.php

Archivo: _reservarIdaTe.php

Interfaz	
Descripción:	Muestra los campos para reservar en un viaje de ida.

Tabla 109 Descripción de la clase interfaz reservarIdaTe.php

Archivo: _reservarRegresoTe.php	
Interfaz	
Descripción:	Muestra los campos para reservar en un viaje de regreso.

Tabla 110 Descripción de la clase interfaz reservarRegresoTe.php

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra la interfaz principal de las reservaciones de la Transportación Estudiantil de Fin de Semana.

Tabla 111 Descripción de la clase interfaz indexSuccess.php

Archivo: puntosSuccess.php	
Interfaz	
Descripción:	Muestra los puntos disponibles de una ruta dada.

Tabla 112 Descripción de la clase interfaz puntosSuccess.php

Archivo: rutasSuccess.php	
Interfaz	
Descripción:	Muestra las rutas disponibles de un municipio dado.

Tabla 113 Descripción de la clase interfaz rutasSuccess.php

2.11.2.11 Reservación de Trabajadores Internos

Nombre: res_tnpActions	
Controladora	
Operaciones	
Nombre:	preExecute
Descripción:	Antes que ocurra cualquier acción, verifica que las reservaciones estén activadas y que el

	usuario autenticado no esté bloqueado en esta modalidad de reservación.
Nombre:	executeIndex
Descripción:	Obtiene todas las reservaciones de familiares que tiene el usuario autenticado.
Nombre:	executeMeses
Descripción:	Obtiene todas las reservaciones estándar y en espera que tenga el usuario.
Nombre:	executeViajes
Descripción:	Obtiene todos los viajes disponibles para un mes.
Nombre:	executeTransportes
Descripción:	Obtiene todos los transportes disponibles para un viaje.
Nombre:	executeReservartnp
Descripción:	Permite reservar un transporte al usuario.
Nombre:	executeListaespera
Descripción:	Permite poner en lista de espera a un usuario para un viaje.
Nombre:	executeCancelar
Descripción:	Permite cancelar una reservación y a la vez subir una persona de la lista de espera.
Nombre:	executeConfirmar
Descripción:	Permite poner el estado de una reservación en confirmada.
Nombre:	executeFamiliar
Descripción:	Permite reservarle a un familiar un viaje.
Nombre:	handleErrorFamiliar
Descripción:	Se ejecuta cuando el formulario de reservación del familiar no está correcto.

Tabla 114 Descripción de la clase controladora res_tnpActions

Nombre: res_tnpComponents	
Controladora	
Operaciones	
Nombre:	executeInfoTransporte
Descripción:	Obtiene todos los datos de un transporte.
Nombre:	executeLink
Descripción:	Envía a su plantilla correspondiente datos para construir vínculos dinámicos.

Tabla 115 Descripción de la clase controladora res_tnpComponents

Nombre: scDatosReservacionTnp	
Utilidad	
Operaciones	
Nombre:	BuscarFamiliaresReservados
Descripción:	Devuelve todas las reservaciones activas de familiares asignados a una persona UCI.
Nombre:	MunicipiosDelWebService
Descripción:	Se consulta a los servicios web para obtener los municipios del país.
Nombre:	MunicipioDadoIdMunicipioTransporte
Descripción:	Se consulta a los servicios web para obtener los datos del municipio de un transporte
Nombre:	FiltrarTransportesPorProvincia
Descripción:	Dado un arreglo de transportes pasados por parámetro se filtran por el criterio del municipio que tienen.
Nombre:	BuscarReservacionesDelTrabajador
Descripción:	Dada una persona, se obtienen todas las reservaciones que tiene activas.
Nombre:	ReservacionesEsperaTrabajador
Descripción:	Dada una persona, se obtienen todas las reservaciones en espera para un semestre.
Nombre:	MostrarMes
Descripción:	Devuelve los meses en los cuales la persona tiene al menos una reservación.
Nombre:	ReservacionDadoUnMes
Descripción:	Devuelve la reservación que tiene en un mes de un viaje específico.
Nombre:	BorrateListaEspera
Descripción:	Permite eliminar una reservación en espera de una persona.
Nombre:	DatosDeUnViajeTnp
Descripción:	Devuelve un viaje de la Transportación Semestral de Trabajadores Internos.
Nombre:	EstaBloqueado
Descripción:	Chequea si una persona está limitada a viajar en un semestre determinado.

Tabla 116 Descripción de la clase utilidad scDatosReservacionTnp

Nombre: scListaEspera	
Utilidad	
Operaciones	
Nombre:	SubirListaEspera

Descripción:	Para un transporte determinado con capacidad libre permite subir de la lista de espera una reservación y distribuirla en dicho transporte.
Nombre:	ParadasIntermedias
Descripción:	Devuelve todas las paradas intermedias que tiene definido un transporte.
Nombre:	EnviarAConfirmados
Descripción:	Permite enviar un mensaje a una persona que ha pasado de lista de espera a reservación oficial.
Nombre:	CancelarReservacion
Descripción:	Permite cancelar una reservación y a la vez subir una persona de la lista de espera.

Tabla 117 Descripción de la clase utilidad scListaEspera

Archivo: _enEspera.php	
Interfaz	
Descripción:	Muestra todas las reservaciones en espera.

Tabla 118 Descripción de la clase interfaz enEspera.php

Archivo: _familiarReservado.php	
Interfaz	
Descripción:	Muestra todas las reservaciones de familiares.

Tabla 119 Descripción de la clase interfaz familiarReservado.php

Archivo: _infoTransporte.php	
Interfaz	
Descripción:	Muestra las paradas intermedias de un transporte.

Tabla 120 Descripción de la clase interfaz infoTransporte.php

Archivo: _link.php	
Interfaz	
Descripción:	Muestra un vínculo que se crea dinámicamente en dependencia de parámetros pasados.

Tabla 121 Descripción de la clase interfaz link.php

Archivo: _reservados.php
Interfaz

Descripción:	Muestra todos los datos de una reservación para un viaje.

Tabla 122 Descripción de la clase interfaz reservados.php

Archivo: _reservar.php	
Interfaz	
Descripción:	Muestra todos los datos de un viaje a reservar.

Tabla 123 Descripción de la clase interfaz reservar.php

Archivo: familiarError.php	
Interfaz	
Descripción:	Muestra mensajes de error cuando el formulario de reservar familiar tiene valores incorrectos.

Tabla 124 Descripción de la clase interfaz familiarError.php

Archivo: familiarSuccess.php	
Interfaz	
Descripción:	Muestra el formulario de reservar familiar.

Tabla 125 Descripción de la clase interfaz familiarSuccess.php

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra la interfaz principal de las reservaciones de Trabajadores Internos con contenedores para mostrar otras vistas de forma asíncrona.

Tabla 126 Descripción de la clase interfaz indexSuccess.php

Archivo: listaesperaSuccess.php	
Interfaz	
Descripción:	Muestra un mensaje cuando se ha agregado una reservación en espera.

Tabla 130 Descripción de la clase interfaz listaesperaSuccess.php

Archivo: mesesSuccess.php	
Interfaz	
Descripción:	Muestra contenedores para mostrar las reservaciones de un semestre específico.

Tabla 127 Descripción de la clase interfaz mesesSuccess.php

Archivo: transportesSuccess.php	
Interfaz	
Descripción:	Muestra todos los transportes disponibles de un viaje específico.

Tabla 128 Descripción de la clase interfaz transportesSuccess.php

Archivo: viajesSuccess.php	
Interfaz	
Descripción:	Muestra todos los viajes disponibles de un mes.

Tabla 129 Descripción de la clase interfaz viajesSuccess.php

2.11.2.12 Reservaciones de la Transportación Masiva

Nombre: reservacionesActions	
Controladora	
Operaciones	
Nombre:	preExecute
Descripción:	Antes que ocurra cualquier acción, verifica que las reservaciones estén activadas.
Nombre:	executeIndex
Descripción:	Aqui se filtran los viajes que tiene el usuario autenticado, si está reservado y/o si tiene familiares reservados.
Nombre:	executeReservar
Descripción:	Permite reservarle un viaje a un estudiante y/o a un trabajador.
Nombre:	executeFamiliar
Descripción:	Permite reservarle un viaje a un familiar.
Nombre:	handleErrorReservarfamiliar
Descripción:	Se ejecuta cuando el formulario de reservar familiar tiene campos no válidos.
Nombre:	executeMunicipios
Descripción:	Consulta los servicios web para obtener los municipios del país.
Nombre:	executeReservarfamiliar
Descripción:	Permite reservarle a un familiar un viaje.

Nombre:	executeCancelar
Descripción:	Permite cancelar una reservación.

Tabla 130 Descripción de la clase controladora reservacionesActions

Nombre: reservacionesComponents	
Controladora	
Operaciones	
Nombre:	executeInformacionTnp
Descripción:	Dada una reservación de Trabajadores Internos, se obtiene información adicional sobre un transporte.
Nombre:	executeInformacionTn
Descripción:	Dada una reservación de Transportación Masiva, se obtiene información adicional sobre un transporte.
Nombre:	executeInformacionFamiliar
Descripción:	Devuelve datos de un familiar.
Nombre:	executeInformacionFamiliar
Descripción:	Devuelve datos de un familiar.

Tabla 131 Descripción de la clase controladora reservacionesComponents

Nombre: scDatosReservacion	
Utilidad	
Operaciones	
Nombre:	BuscarFamiliaresReservados
Descripción:	Devuelve todas las reservaciones de familiares asociados a una persona UCI.
Nombre:	MunicipiosDelWebService
Descripción:	Se consulta a los servicios web para obtener los municipios del país.
Nombre:	MunicipioDadoIdMunicipioTransporte
Descripción:	Se consulta a los servicios web para obtener los datos del municipio de un transporte
Nombre:	BuscarReservacionesDelTrabajador
Descripción:	Obtiene todas las reservaciones de un trabajador.
Nombre:	ReservacionesEsperaTrabajador
Descripción:	Obtiene todas las reservaciones en espera de un trabajador.

Nombre:	CancelarReservacion
Descripción:	Permite cancelar una reservación y a la vez subir una persona de la lista de espera.

Tabla 132 Descripción de la clase utilidad scDatosReservacion

Archivo: _familiarReservado.php	
Interfaz	
Descripción:	Muestra todas las reservaciones de familiares.

Tabla 133 Descripción de la clase interfaz familiarReservado.php

Archivo: _informaciónFamiliar.php	
Interfaz	
Descripción:	Muestra datos personales de un familiar reservado.

Tabla 134 Descripción de la clase interfaz informaciónFamiliar.php

Archivo: _informaciónTn.php	
Interfaz	
Descripción:	Muestra los datos de la distribución de una reservación.

Tabla 135 Descripción de la clase interfaz informaciónTn.php

Archivo: _informaciónTnp.php	
Interfaz	
Descripción:	Muestra los datos de la distribución de una reservación.

Tabla 136 Descripción de la clase interfaz informaciónTnp.php

Archivo: _reservadosIda.php	
Interfaz	
Descripción:	Muestra la reservación para un viaje de ida.

Tabla 141Descripción de la clase interfaz reservadosIda.php

Archivo: _reservadosRegreso.php	
Interfaz	
Descripción:	Muestra la reservación para un viaje de regreso.

Tabla 137 Descripción de la clase interfaz reservadosRegreso.php

Archivo: _reservarIda.php	
Interfaz	
Descripción:	Muestra los datos de un viaje de ida para reservar.

Tabla 138 Descripción de la clase interfaz reservarIda.php

Archivo: _reservarRegreso.php	
Interfaz	
Descripción:	Muestra los datos de un viaje de regreso para reservar.

Tabla 139 Descripción de la clase interfaz reservarRegreso.php

Archivo: bloqueoSuccess.php	
Interfaz	
Descripción:	Se muestra cuando el usuario autenticado está limitado para reservar.

Tabla 140 Descripción de la clase interfaz bloqueoSuccess.php

Archivo: familiarSuccess.php	
Interfaz	
Descripción:	Muestra el formulario para reservar un familiar.

Tabla 141 Descripción de la clase interfaz familiarSuccess.php

Archivo: indexSuccess.php	
Interfaz	
Descripción:	Muestra la interfaz principal de la reservación de la Transportación Masiva con contenedores para mostrar otras páginas de forma asíncrona.

Tabla 142 Descripción de la clase interfaz indexSuccess.php

Archivo: municipiosSuccess.php	
Interfaz	
Descripción:	Muestra un *select* con los municipios de una provincia dada.

Tabla 143 Descripción de la clase interfaz municipiosSuccess.php

Archivo: principalSuccess.php

Interfaz	
Descripción:	Muestra la interfaz principal de todas las modalidades de reservaciones con contenedores para mostrar otras páginas de forma asíncrona.

Tabla 144 Descripción de la clase interfaz principalSuccess.php

2.11.2.13 Reservaciones puntuales de Trabajadores Internos

Nombre: reservaciones_puntualesActions	
Controladora	
Operaciones	
Nombre:	executeBuscar
Descripción:	Permite preparar los datos de una persona buscada para que el administrador pueda gestionar sus reservaciones.
Nombre:	executeMeses
Descripción:	Obtiene todas las reservaciones estándar y en espera que tenga el usuario buscado.
Nombre:	executeViajes
Descripción:	Obtiene todos los viajes disponibles para un mes.
Nombre:	executeTransportes
Descripción:	Obtiene todos los transportes disponibles para un viaje.
Nombre:	executeReservartnp
Descripción:	Permite reservar un transporte al usuario.
Nombre:	executeListaespera
Descripción:	Permite poner en lista de espera a un usuario para un viaje.
Nombre:	executeCancelar
Descripción:	Permite cancelar una reservación y a la vez subir una persona de la lista de espera.
Nombre:	executeConfirmar
Descripción:	Permite poner el estado de una reservación en confirmada.
Nombre:	executeFamiliar
Descripción:	Permite reservarle a un familiar un viaje.
Nombre:	handleErrorFamiliar

Descripción:	Se ejecuta cuando el formulario de reservación del familiar no está correcto.

Tabla 145 Descripción de la clase controladora reservaciones_puntualesActions

Nombre: scDatosReservacionTnpGest	
Utilidad	
Operaciones	
Nombre:	BuscarFamiliaresReservados
Descripción:	Devuelve todas las reservaciones activas de familiares asignados a una persona UCI.
Nombre:	MunicipiosDelWebService
Descripción:	Se consulta a los servicios web para obtener los municipios del país.
Nombre:	MunicipioDadoIdMunicipioTransporte
Descripción:	Se consulta a los servicios web para obtener los datos del municipio de un transporte
Nombre:	FiltrarTransportesPorProvincia
Descripción:	Dado un arreglo de transportes pasados por parámetro se filtran por el criterio del municipio que tienen.
Nombre:	BuscarReservacionesDelTrabajador
Descripción:	Dada una persona, se obtienen todas las reservaciones que tiene activas.
Nombre:	ReservacionesEsperaTrabajador
Descripción:	Dada una persona, se obtienen todas las reservaciones en espera para un semestre.
Nombre:	MostrarMes
Descripción:	Devuelve los meses en los cuales la persona tiene al menos una reservación.
Nombre:	ReservacionDadoUnMes
Descripción:	Devuelve la reservación que tiene en un mes de un viaje específico.
Nombre:	BorrateListaEspera
Descripción:	Permite eliminar una reservación en espera de una persona.
Nombre:	DatosDeUnViajeTnp
Descripción:	Devuelve un viaje de la Transportación Semestral de Trabajadores Internos.

Tabla 146 Descripción de la clase controladora scDatosReservacionTnpGest

Nombre: scListaEspera
Utilidad
Operaciones

Nombre:	SubirListaEspera
Descripción:	Para un transporte determinado con capacidad libre permite subir de la lista de espera una reservación y distribuirla en dicho transporte.
Nombre:	ParadasIntermedias
Descripción:	Devuelve todas las paradas intermedias que tiene definido un transporte.
Nombre:	EnviarAConfirmados
Descripción:	Permite enviar un mensaje a una persona que ha pasado de lista de espera a reservación oficial.
Nombre:	CancelarReservacion
Descripción:	Permite cancelar una reservación y a la vez subir una persona de la lista de espera.

Tabla 147 Descripción de la clase utilidad scListaEspera

Archivo: buscarSuccess.php	
Interfaz	
Descripción:	Muestra los datos de una persona buscada para acceder a las reservaciones de Trabajadores Internos.

Tabla 148 Descripción de la clase interfaz buscarSuccess.php

Archivo: buscartnSuccess.php	
Interfaz	
Descripción:	Muestra un contenedor para mostrar **buscarSuccess.php** de forma asíncrona.

Tabla 149 Descripción de la clase interfaz buscartnSuccess.php

2.11.2.14 Clases utilitarias extras

Nombre: Reportes	
Utilidad	
Operaciones	
Nombre:	Reservados
Descripción:	Devuelve todos los datos de las personas reservadas para un transporte.
Nombre:	ReporteOmnibus

Descripción:	Devuelve la cantidad de personas distribuidas en un ómnibus.
Nombre:	ReservadosMes
Descripción:	Devuelve todos los datos de las personas reservadas para un viaje en un mes específico.
Nombre:	ReservadosAreas
Descripción:	Devuelve todos los datos de las personas reservadas para un área UCI.

Tabla 150 Descripción de la clase utilidad Reportes

Nombre: scFechas	
Utilidad	
Operaciones	
Nombre:	EsMasAntigua
Descripción:	Compara dos fechas.
Nombre:	ActivarConfirmacion
Descripción:	Dado una reservación de Trabajadores Internos permite activar o no la opción de confirmar reservación.
Nombre:	DesactivarOpciones
Descripción:	Para un mes, devuelve verdadero si la fecha actual está entre la fecha del viaje y (fecha_viaje – diasparaconfirmar)
Nombre:	YaPasoElViaje
Descripción:	Retorna verdadero si ya se efectuó el viaje.
Nombre:	Informacion
Descripción:	Convierte una fecha a texto y “dice” cuánto tiempo queda para el viaje.

Tabla 151 Descripción de la clase utilidad scFechas

Nombre: scHerramientas	
Utilidad	
Operaciones	
Nombre:	CorreoSegunAplicacion
Descripción:	Chequea en la configuración de la aplicación y devuelve el destinatario para las quejas para una modalidad.

Tabla 152 Descripción de la clase utilidad scHerramientas

Archivo: _datosUsuario.php	
Interfaz	
Descripción:	Muestra todos los datos obtenidos de los servicios web de un usuario

Tabla 153 Descripción de la clase interfaz datosUsuario.php

Archivo: _info.php	
Interfaz	
Descripción:	Muestra la información editable que aparece en la pantalla de bienvenida del sitio.

Tabla 154 Descripción de la clase interfaz info.php

Archivo: _infoTrabajadores.php	
Interfaz	
Descripción:	Muestra la información que aparece en la reservación de Trabajadores Internos.

Tabla 155 Descripción de la clase interfaz infoTrabajadores.php

Archivo: imprimir.php	
Interfaz	
Descripción:	Muestra la página con formato de impresión.

Tabla 156 Descripción de la clase interfaz imprimir.php

Archivo: layout.php	
Interfaz	
Descripción:	Muestra la plantilla general sobre la cual salen todas las vistas de cada módulo.

Tabla 157 Descripción de la clase interfaz layout.php

Archivo: main.php	
Interfaz	
Descripción:	Muestra la plantilla general alternativa sobre la cual salen todas las vistas de cada módulo.

Tabla 158 Descripción de la clase interfaz main.php

2.11.3 Descripción de los Servicios Web brindados

2.11.3.1 Transportación Estudiantil de Fin de Semana

Nombre: Reservaciones	
Utilidad	
Operaciones	
Nombre:	_obtener_viaje
Descripción:	Dado un id_expediente y el tipo de viaje se devuelve el viaje para un usuario si no está bloqueado
Nombre:	_obtener_reservacion
Descripción:	Devuelve una reservación para un id_expediente
Nombre:	_acceso
Descripción:	Chequea la clave que será utilizada en los métodos de inserción.
Nombre:	ObtenerPuntos
Descripción:	Devuelve todos los puntos habilitados de la modalidad.
Nombre:	ObtenerIdRelacionPuntoRuta
Descripción:	Obtiene la relación entre ruta y puntos.
Nombre:	ObtenerRutas
Descripción:	Devuelve todas las rutas habilitadas.
Nombre:	ObtenerPuntosDadoIdRuta
Descripción:	Devuelve todos los puntos de paradas de una ruta
Nombre:	ObtenerFamiliaresVisitadosDadoIdExpediente
Descripción:	Devuelve todos los familiares que ha visitado el estudiante
Nombre:	ObtenerViajeFinDeSemanaIdaDadoIdExpediente
Descripción	Devuelve el viaje del próximo fin de semana de ida.
Nombre:	ObtenerViajeFinDeSemanaRegresoDadoIdExpediente
Descripción:	Devuelve el viaje del próximo fin de semana de regreso.
Nombre:	ReservarEstudiante
Descripción:	Permite reservar a un estudiante para un viaje de fin de semana, esto incluye agregar un familiar si no tiene ninguno registrado.
Nombre:	ModificarReservacionEstudiante
Descripción:	Permite modificar el punto destino de una reservación.

Nombre:	CancelarReservacionEstudiante
Descripción:	Permite cancelar una reservación.

Tabla 159 Descripción de la clase utilidad Reservaciones

2.11.3.2 Transportación Masiva de Estudiantes y Trabajadores Internos

Nombre: Masiva	
Utilidad	
Operaciones	
Nombre:	_obtener_reservacion
Descripción:	Devuelve una reservación para un id_expediente
Nombre:	_acceso
Descripción:	Chequea la clave que será utilizada en los métodos de inserción.
Nombre:	_obtener_viajes
Descripción:	Devuelve todos los viajes disponibles para una persona.
Nombre:	_obtener_distribucion
Descripción:	Obtiene la ubicación y los datos del transporte de una reservación.
Nombre:	ObtenerViajesIdaDisponiblesDadoIdExpediente
Descripción:	Devuelve, dado el id_expediente, un arreglo de viajes disponibles de tipo ida.
Nombre:	ObtenerViajesRegresoDisponiblesDadoIdExpediente
Descripción:	Devuelve, dado el id_expediente, un arreglo de viajes disponibles de tipo regreso.
Nombre:	ObtenerReservacionIdaDadoIdExpediente
Descripción:	Devuelve la reservación de ida dado id_expediente
Nombre:	ObtenerReservacionRegresoDadoIdExpediente
Descripción	Devuelve la reservación de ida dado id_expediente.
Nombre:	ObtenerTransporte
Descripción:	Devuelve los datos del transporte en el cual la persona está sentada.
Nombre:	CancelarReservacion
Descripción:	Cancela una reservación.
Nombre:	ReservarPersona

Descripción:	Permite hacer una reservación.

Tabla 160 Descripción de la clase utilidad Masiva

2.11.3.3 Servicios web generales

Nombre: scTransportesWebServices	
Utilidad	
Operaciones	
Nombre:	ObtenerReservacionesDeIda
Descripción:	Devuelve todas las reservaciones de ida de las tres modalidades de reservaciones.
Nombre:	ObtenerReservacionDadoIdExpediente
Descripción:	Devuelve todas las reservaciones dado el id_expediente de una persona
Nombre:	ObtenerReservacionesDeRegreso
Descripción:	Devuelve todas las reservaciones de regreso de las tres modalidades de reservaciones.

Tabla 161 Descripción de la clase utilidad scTransportesWebServices

2.12 Conclusiones

La programación de un software no siempre es concluida con la entrega del mismo, muchas veces es necesario agregarle ciertas funcionalidades o corregir algunas que no hayan sido bien programadas. Con la ayuda de estas descripciones, cualquier programador verá un útil manual que le permitirá comprender el funcionamiento de sistema fácilmente aunque nunca antes haya visto programa, lo que le permitiría darle soporte sin ser necesariamente el auténtico desarrollador.

VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

Dentro de los procesos más importantes en el desarrollo de una aplicación se encuentran las pruebas de código, ya que es la mejor forma de detectar errores tempranamente en el desarrollo. En este capítulo se describen los procesos de pruebas **unitarias** de los módulos Reportes, Reservación y Gestión de la Información pues de todo el conjunto de pruebas (aceptación, integración, carga y estrés), es considerada la más importante para garantizar un proyecto exitoso. Se pretende utilizar las herramientas y utilidades que proporciona Symfony para automatizar las pruebas asegurándose de esta forma la compatibilidad, fiabilidad y agilidad del antes mencionado proceso.

3.2 Framework de pruebas

El framework Symfony incluye su propio framework de pruebas llamado **lime** que se basa en la librería Test::More de Perl y es compatible con TAP(*Test Anything Protocol*), protocolo creado para facilitar la lectura de los resultados de las pruebas. Está basado en la metodología TDD (desarrollo basado en pruebas) el cual establece que las pruebas se escriben antes que el código de la aplicación. Esto ayuda a pensar y centrarse en el funcionamiento de un método antes de programarlo. Se trata de una buena práctica que también recomiendan otras metodologías como XP (*Extreme Programming*). Además, es un hecho innegable que si no se escriben las pruebas antes, se acaba sin escribirlas nunca. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto.

Las pruebas unitarias de Symfony son archivos PHP cuyo nombre termina en Test.php y que se encuentran en el directorio **test** del proyecto. Su sintaxis es sencilla y fácil de leer. Cada prueba unitaria

consiste en una llamada a un método de la instancia de **lime_test**, clase principal de la librería. El objeto lime_test dispone de un gran número de métodos para las pruebas, como se muestra a continuación:

Método	Descripción
diag($mensaje)	Muestra un comentario, pero no ejecuta ninguna prueba
ok($prueba, $mensaje)	Si la condición que se indica es true, la prueba tiene éxito
is($valor1, $valor2, $mensaje)	Compara 2 valores y la prueba pasa si los 2 son iguales (==)
isnt($valor1, $valor2,$mensaje)	Compara 2 valores y la prueba pasa si no son iguales
like($cadena,$expresionRegular,$mensaje)	Prueba que una cadena cumpla con el patrón de una expresión regular
unlike($cadena,$expresionRegular,$mensaje)	Prueba que una cadena no cumpla con el patrón de una expresión regular
cmp_ok($valor1,$operador,$valor2,$mensaje)	Compara 2 valores mediante el operador que se indica
isa_ok($variable, $tipo,$mensaje)	Comprueba si la variable que se le pasa es del tipo que se indica
isa_ok($objeto, $clase,$mensaje)	Comprueba si el objeto que se le pasa es de la clase que se indica
can_ok($objeto, $metodo,$mensaje)	Comprueba si el objeto que se le pasa dispone del método que se indica
is_deeply($array1, $array2,$mensaje)	Comprueba que 2 arreglos tengan los mismos valores
include_ok($archivo, $mensaje)	Valida que un archivo existe y que ha sido incluido correctamente
fail()	Provoca que la prueba siempre falle (es útil para las excepciones)
pass()	Provoca que la prueba siempre se pase (es

	útil para las excepciones)
skip($mensaje, $numeroPruebas)	Cuenta como si fueran $numeroPruebas pruebas (es útil para las pruebas condicionales)
todo()	Cuenta como si fuera 1 prueba (es útil para las pruebas que todavía no se han escrito)

Tabla 162 Métodos de la clase lime_test para realizar pruebas al código

Casi todos los métodos permiten indicar un mensaje como último parámetro. Este mensaje es el que se muestra como resultado esperado de la prueba cuando esta tiene éxito, así se compara con el que devuelve la prueba. Para ejecutar el conjunto de pruebas, se utiliza la tarea **test-unit** desde la línea de comandos. El resultado de esta tarea es muy explícito, lo que permite localizar fácilmente las pruebas que han fallado y las que se han ejecutado correctamente.

3.3 Objetivos

El objetivo principal del test seleccionado es ofrecer un método ágil de probar eficientemente el funcionamiento interno del sistema a los desarrolladores para lograr el avance seguro del desarrollo de la aplicación y evitando inconvenientes con los cambios de requisitos. Es por ello que se pensó en utilizar las pruebas automáticas como vía de revisar las funcionalidades ya que los demás métodos tomarían mucho más tiempo y no se podrían realizar todas las pruebas pertinentes. Este tipo de metodología de pruebas ayuda mucho a los programadores a la hora de implementar y probar lo que está realizando ya que cuenta con una herramienta confiable para comprobar el buen funcionamiento de lo que se programa, acción que ocurre muy frecuentemente.

3.4 Alcance

Como se menciona anteriormente el objetivo del test es facilitarle al programador una vía de comprobar si lo que está programando esta correcto o no cada vez que termina un método, por lo tanto esta herramienta lo estará acompañando durante todo el proceso de desarrollo desde que escribe el primer método de la primer clase hasta la implementación en la etapa de soporte donde se continúa iterando y modificando el código, para ir probando la funcionalidad de los mismos.

3.5 Descripción de los valores utilizados para los test

Las tablas que se presentan a continuación representan las pruebas realizadas a las principales funcionalidades implementadas agrupadas por clases. En las mismas se recogen los resultados obtenidos después de aplicar aquellos métodos que ofrece **lime** para comprobar el correcto funcionamiento de las funcionalidades a comprobar. En el primer campo (Funcionalidad) corresponde al nombre de la funcionalidad que será comprobada, el segundo (Método de Prueba) es el método de la clase **lime_test** utilizado, el tercer campo (Recibe) contiene los parámetros que recibe la funcionalidad al ser comprobada, el cuarto campo (Esperado) es el resultado que se espera que devuelva el método de prueba y el quinto campo (Regresó) es el resultado real obtenido. El resultado es satisfactorio si coinciden los resultados de las pruebas con los esperados o sea, los campos Esperado y Regresó. En el caso del campo Recibe cuando algún elemento va acompañado de (NV) representa un parámetro no válido donde el comportamiento del método será distinto.

3.6 Evaluación de la ejecución del test y de los resultados

3.6.1 Pruebas del módulo Reportes

Reportes.class.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
Reservados	isa_ok	20	Ok	Ok
Reservados	is	20	Not ok	Not ok
Reservados	is	304928(NV)	Ok	Ok
ReporteOmnibus	is	20	Ok	Ok
ReporteOmnibus	is	304928(NV)	Ok	Ok
Metodo ReservadosMes	isa_ok	19,1	Ok	Ok

ReservadosMes	is	19,1	Not ok	Not ok
ReservadosMes	is	304928,0(NV)	Ok	Ok
ReservadosAreas	isa_ok	L0000,90	Ok	Ok
ReservadosAreas	is	L0000,90	Not ok	Not ok
ReservadosAreas	is	A,0(NV)	Ok	Ok

Tabla 163 Resultados de pruebas. Clase Reportes: módulo Reportes

Listado.class.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
ListaEspera	isa_ok	90	Ok	Ok
ListaEspera	is	90	Not ok	Not ok
ListaEspera	is	304928(NV)	Ok	Ok
Reservados	isa_ok	20	Ok	Ok
Reservados	is	20	Not ok	Not ok
Reservados	is	304928(NV)	Ok	Ok
ListaEsperaPorOmnibus	isa_ok	90, 34	Ok	Ok
ListaEsperaPorOmnibus	is	90, 34	Not ok	Not ok
ListaEsperaPorOmnibus	is	304928 304928(NV)	Ok	Ok

Tabla 164 Resultados de pruebas. Clase Listado: módulo Reportes

Listado1.class.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
Reservados	isa_ok	20	Ok	Ok
Reservados	is	20	Not ok	Not ok

Reservados	is	304928(NV)	Ok	Ok

Tabla 165 Resultados de pruebas. Clase Listado1: módulo Reportes

TbDviajeTn.class.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
cantReservacionesArea	isa_ok	L0000	Ok	Ok
Reservados	is	L0000	Not ok	Not ok

Tabla 166 Resultados de pruebas. Clase TbDviajeTn: módulo Reportes

```
14  $database = new sfDatabaseManager();
15  $database->initialize();
16  ///************************************************************************
17
18
19  $probador = new lime_test(30, new lime_output_color());
20
21
22  $probador->diag('////////////////////////////Reportes.class.php/////////////////////////////////////');
23  $probador->diag('------------------Reservados($id_transporte)------------------------------');
24  $probador->isa_ok(Reportes::Reservados(34),'array','Parametro vÁlido');
25  $probador->is(Reportes::Reservados(34),array(),'Parametro vÁlido');
26  $probador->is(Reportes::Reservados(304928),array(),'Parametro no vÁlido');
27
28
29  $probador->diag('------------------ReporteOmnibus($id_transporte)------------------------------');
30  $probador->is(Reportes::ReporteOmnibus(34),45,'Parametro vÁlido');
31  $probador->is(Reportes::ReporteOmnibus(304928),null,'Parametro no vÁlido');
32
33
34  $probador->diag('------------------Metodo ReservadosMes($id_viaje, $id_mes)------------------------------');
35  $probador->isa_ok(Reportes::ReservadosMes(19,1),'array','Parametro vÁlido');
36  $probador->is(Reportes::Reservados(19,1),array(),'Parametro vÁlido');
37  $probador->is(Reportes::Reservados(304928,0),array(),'Parametro no vÁlido');
38
39
40  $probador->diag('------------------ReservadosAreas($id_area,$id_viaje, $id_aplicacion=null)----------');
41  $probador->isa_ok(Reportes::ReservadosAreas('L0000',90),'array','Parametro vÁlido');
42  $probador->is(Reportes::ReservadosAreas('L0000',90),array(),'Parametro vÁlido');
43  $probador->is(Reportes::ReservadosAreas('A',0),array(),'Parametro no vÁlido');
```

Figura 3 Vista de las pruebas realizadas a métodos del módulo Reportes

3.6.2 Pruebas del módulo Reservaciones

scDatosReservacionTnp.class.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
BuscarFamiliaresReservados	isa_ok	12786	Ok	Ok
BuscarFamiliaresReservados	is	12786	Ok	Ok
BuscarFamiliaresReservados	is	50019	Ok	Ok
BuscarReservacionesDelTrabajador	is	Un objeto de tipo TbDpersona	Ok	Ok
BuscarReservacionesDelTrabajador	isnt	Un objeto de tipo TbDpersona	Ok	Ok
ReservacionesEsperaTrabajador	isa_ok	12786	Ok	Ok
ReservacionesEsperaTrabajador	isnt	12786	Ok	Ok

Tabla 167 Resultados de pruebas. Clase scDatosReservacionTnp: módulo Reservaciones

scListaEspera.class.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
SubirListaEspera	is	20,45	Ok	Ok

Tabla 168 Resultados de pruebas. Clase scListaEspera: módulo Reservaciones

TbDpersona.class.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
ObtenerUltimaReservacion	isa_ok	Tipo de datos: TbDpersona	Ok	Ok

Tabla 169 Resultados de pruebas. Clase TbDpersona: módulo Reservaciones

TbDreservacionTnl.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
EstaDistribuido	is	true	Ok	Ok
DatosDistribucion	isa_ok	array	Ok	Ok
DatosDistribucion	is	array()	Ok	Ok

Tabla 170 Resultados de pruebas. Clase TbDreservacionTnl: módulo Reservaciones

TbDviajeTnl.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
EstaEnPeriodoReservacion	Ok		Ok	Ok

Tabla 171 Resultados de pruebas. Clase TbDviajeTnl: módulo Reservaciones

TbDviajeTe.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
ViajesProximo	isa_ok	TbDviajeTe	Ok	Ok

Tabla 172 Resultados de pruebas. Clase TbDviajeTe: módulo Reservaciones

```
21  $probador= new lime_test(30, new lime_output_color());
22
23  $probador->diag('//////////////////scDatosReservacionTnp.class.php////////////////');
24
25  $probador->diag('----------BuscarFamiliaresReservados($id_expediente)--------');
26  $probador->isa_ok(scDatosReservacionTnp::BuscarFamiliaresReservados(12786), 'array', 'Tipo de datos');
27  $probador->is(scDatosReservacionTnp::BuscarFamiliaresReservados(12786), array(), 'Parametros validos');
28  $probador->is(scDatosReservacionTnp::BuscarFamiliaresReservados(50019), array(), 'Parametros invalidos');
29
30
31  $probador->diag('----------BuscarReservacionesDelTrabajador($persona)--------');
32  $persona = TbDpersonaPeer::retrieveByPK(37);
33  $probador->isa_ok(scDatosReservacionTnp::BuscarReservacionesDelTrabajador($persona),'array','Devuelve un arreglo'
34  $probador->isnt(scDatosReservacionTnp::BuscarReservacionesDelTrabajador($persona),array(),'Parametro valido');
35
36  $probador->diag('----------ReservacionesEsperaTrabajador($id_expediente, $id_semestre)--------');
37  $probador->isa_ok(scDatosReservacionTnp::ReservacionesEsperaTrabajador(12786, 1),'array','Devuelve un arreglo');
38  $probador->isnt(scDatosReservacionTnp::ReservacionesEsperaTrabajador(12786, 1),array(),'Valor real');
39
40  $probador->diag('//////////////////scListaEspera:.class.php////////////////');
41
42  $probador->diag('-----------------SubirListaEspera($id_viaje,$id_transporte)----------------------');
43  $probador->is(scListaEspera::SubirListaEspera(20, 45),null,'Subir lista de espera para un transporte, este metodo
44
45
46  ////////////////////////////////////Modelo/////////////////////////////////////////
47  $probador->diag('//////////////////TbDpersona.php////////////////');
48  $criterio = new Criteria();
49  $criterio->add(TbDpersonaUciPeer::ID_EXPEDIENTE,50019);
50  $criterio->addJoin(TbDpersonaUciPeer::ID_PERSONA, TbDpersonaPeer::ID_PERSONA);
```

Figura 4 Vista de las pruebas realizadas a métodos del módulo Reservaciones

3.6.3 Pruebas del módulo Gestión de la información

TbDviaje.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
DistribuirMunicipioEn	is	null	Ok	Ok
DistribuirReservacionEn	is	null	Ok	Ok
CancelarDistribucionAutomatica	is	null	Ok	Ok
CancelarDistribucionMunicipio	is	null	Ok	Ok
CancelarPuntual	is	null	Ok	Ok
EstaReservada	is	false	Ok	Ok

Tabla 173 Resultados de pruebas. Clase TbDviaje: módulo Gestión de la Información

TbDviajeTn.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
Boletines	isa_ok	array	Ok	Ok
Boletines	is	Array()	Ok	Ok
CapacidadProvincia	isa_ok	integer	Ok	Ok
CapacidadProvincia	is	0	Ok	Ok
DisponiblesProvincia	isa_ok	integer	Ok	Ok
DisponiblesProvincia	is	0	Ok	Ok
CantidadReservaciones	isa_ok	integer	Ok	Ok
CantidadReservaciones	is	0	Ok	Ok
CantidadDistribuidos	isa_ok	integer	Ok	Ok
CantidadDistribuidos	is	0	Ok	Ok
CantidadFamiliares	isa_ok	integer	Ok	Ok
CantidadFamiliares	is	0	Ok	Ok
CantidadReservacionesMunicipio	isa_ok	integer	Ok	Ok
CantidadReservacionesMunicipio	is	0	Ok	Ok
CantidadReservacionesDistribuidasMunicipio	isa_ok	integer	Ok	Ok
CantidadReservacionesDistribuidasMunicipio	is	0	Ok	Ok
FamiliaresReservadosMunicipio	isa_ok	integer	Ok	Ok
FamiliaresReservadosMunicipio	is	0	Ok	Ok

Tabla 174 Resultados de pruebas. Clase TbDviajeTn: módulo Gestión de la Información

TbDviajeTnp.php				
Funcionalidad	**Método de Prueba**	**Recibe**	**Esperado**	**Regresó**
PersonasSentadas	isa_ok	array	Ok	Ok
PersonasSentadasSinConfirmr	isa_ok	array	Ok	Ok

Tabla 175 Resultados de pruebas. Clase TbDviajeTnp: módulo Gestión de la Información

```
21  $test = new lime_test(30, new lime_output_color());
22
23  $test->diag('////////////////////////TbDviaje.php/////////////////////////////////////////////');
24  $test->diag('----------------------DistribuirMunicipioEn($id_municipio,$id_transporte)---------------');
25  $viaje = TbDviajePeer::doSelectOne(new Criteria());
26  $test->is($viaje->DistribuirMunicipioEn(01111, 40),null,'Este metodo es void');
27
28  $test->diag('----------------------DistribuirReservacionEn($id_reservacion,$id_transporte)---------------');
29  $test->is($viaje->DistribuirReservacionEn(42401, 40),null,'Este metodo es void');
30
31
32  $test->diag('----------------------CancelarDistribucionAutomatica($ids_municipio)---------------');
33  $test->is($viaje->CancelarDistribucionAutomatica(array(1109)),null,'Este metodo es void');
34
35
36  $test->diag('----------------------CancelarDistribucionMunicipio($id_municipio)---------------');
37  $test->is($viaje->CancelarDistribucionMunicipio(1109),null,'Este metodo es void');
38
39
40  $test->diag('----------------------CancelarPuntual($id_reservacion,$id_transporte)---------------');
41  $test->is($viaje->CancelarPuntual(42301,13),null,'Este metodo es void');
42
43  $test->diag('----------------------EstaReservada($id_persona)---------------');
44  $test->is($viaje->EstaReservada(1),false,'Saber si una persona esta reservada');
45  $test->is($viaje->EstaReservada('fsadf'),false,'Con parametros invalidos');
46
47
48
49  $test->diag('////////////////////////TbDviajeTn.php/////////////////////////////////////////////');
50  $test->diag('----------------------Boletines($id_transporte)---------------');
```

Figura 5 Vista de las pruebas realizadas a métodos del módulo Gestión de la Información

3.7 Conclusiones

De mucha ayuda resulta la utilización del framework de pruebas **lime** de Symfony, ya que muchas veces se obvian estos procesos sistemáticos por lo tedioso que resulta probar correctamente una aplicación cada vez que varían los requisitos, caso que resulta muy frecuente en este tipo de desarrollo. De esta manera se logran realizar todas las pruebas necesarias correctamente, con el mínimo de esfuerzo y en un tiempo relativamente corto. Se ha probado la integridad de los métodos que forman parte del núcleo del sistema y que representan el pilar del funcionamiento de las tres modalidades de reservaciones.

CONCLUSIONES

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

- ✓ Se logró concebir el módulo Reservaciones con la capacidad de procesar miles de solicitudes en cuestión de pocos minutos desde los servidores asignados donde se demostró la eficiencia y la adaptabilidad a las condiciones de la UCI.
- ✓ Se logró implementar el módulo Gestión de la Información con las facilidades de proporcionar reportes e información necesaria para el control de las reservaciones y del personal que solicitan servicios; evitando la utilización de recursos, tiempo y esfuerzo innecesarios en los complejos procesos de la reservaciones.
- ✓ Se logró la implementación un Servicio Web que no solo es capaz de ofrecer los requeridos previamente, sino que es capaz de mediante estos servicios de realizar las reservaciones a través de cualquier aplicación futura de la universidad.
- ✓ Se realizó un estudio de las herramientas informáticas que posibilitarían la implementación de la solución propuesta.
- ✓ Se documentó todo en cuanto a implementación de funcionalidades de los módulos.

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

Con el presente trabajo se demuestra la solución del problema a resolver relacionado con los procesos de gestión de las reservaciones del Departamento de Transportaciones Nacionales UCI en la Universidad de las Ciencias Informáticas.

RECOMENDACIONES

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

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

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

REFERENCIAS BIBLIOGRÁFICAS

1. [En línea] http://www.internetworldstats.com/stats.htm
2. [En línea] http://www.cgisecurity.com/xss-faq.html.
3. [En línea] http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html.
4. **Sensio Labs.** *The symfony Cookbook.* 2008.
5. [En línea] http://c2.com/cgi/wiki?DontRepeatYourself .
6. **Beck, Kent.** *Implementation Patterns.* s.l. : Addison-Wesley, 2007. ISBN 978-0321413093.
7. **Avgeriou, Paris y Uwe, Zdun.** *Architectural patterns revisited:a pattern language.* Irsee, Germany : s.n., 2005.

BIBLIOGRAFÍA

1. **Booch, Grady.** *OBJECT-ORIENTED ANALYSIS AND DESIGN.* s.l. : Addison Wesley Longman, Inc., 1994.
2. **Cerami, Ethan.** *Web Services Essentials. Distributed Applications with XML-RPC, SOAP, UDDI & WSDL.* s.l. : O'Reilly, 2002. ISBN: 0-596-00224-6.
3. **Hernández Sampieri, Roberto. Fernández Collado, Carlos. Baptista Lucio, Pilar.** *Metodología de la Investigación.* México D.F. : McGRAW-HILL INTERAMERICANA EDITORES, S. A. de C. V., 1998. ISBN 970-10-1899-0.
4. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El proceso unificado de desarrollo de software.* 2000.
5. **Jim Conallen***, Building Web Applications with UML, Object Technology Series by Addison Wesley Longman,* 1999
6. **Jim Conallen,** *Modeling Web Application Architectures with UML, Rational*
7. **Jim Conallen,** *UML Extension for Web Applications 0.91,* 1999**.**
8. *Software,*1999
9. **Oren Ben-Kiki**, *YAML Ain't Markup Language (YAML™).* C. E. 2004.
10. **Pérez, Javier Eguíluz.** *Introducción a AJAX.* 2008.
11. **Potencier, Fabien .** symfony, Web PHP framework. *Symfony* [En línea] Sensio-Labs, 2005. www.symfony-project.org.
12. **Pressman, Roger S..** *Ingeniería del Software. Un enfoque práctico*.(traducido). 2001.
13. **Racciatti, Hernán Marcelo.** *Seguridad en Aplicaciones Web***.** 2005, Vol. @RROBA # 96.
14. **Rosario, Jimmy**, *"La Tecnología de la Información y la Comunicación (TIC). Su uso como Herramienta para el Fortalecimiento y el Desarrollo de la Educación Virtual". Disponible en el ARCHIVO del Observatorio para la CiberSociedad en http://www.cibersociedad.net/archivo/articulo.php?art=218.* 2005.
15. **Stan Ward, Per Kroll,** *Building Web Solutions with the Rational Unified Process: Unifying the Creative Design Process and the Software Engineering Process*

GLOSARIO DE TÉRMINOS

1. **BOM: (*Browser Object Model).*** Permite acceder y modificar las propiedades de las ventanas del propio navegador. Mediante BOM, es posible redimensionar y mover ventanas, modificar el texto que se muestra en la barra de estado y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML.
2. **COOKIES:** Pequeños textos que determinados sitios web guardan en el ordenador para almacenar información referente a las preferencias del usuario. Ellas permiten recoger información acerca del uso del sitio.
3. **CORBA*: (Common Object Request Broker Architecture)***. Arquitectura e infraestructura independiente que las aplicaciones pueden usar para interactuar a través de las redes.
4. **DOM**: (***Document Object Model)*.** Conjunto de utilidades específicamente diseñadas para manipular documentos XML. Por extensión, DOM también se puede utilizar para manipular documentos XHTML y HTML.
5. **FTP: *(File Transfer Protocol)*** Protocolo de transferencia de archivos.
6. **JSON: *(JavaScript Object Notation***) Formato para intercambio de datos, fácil de leer y escribir para los humanos y ligero para las máquinas para procesar. Está basado en la sintaxis de representación de arreglos en Javascript.
7. **HTTP: *(Hypertext Transfer Protocol)*** Protocolo de transmisión de hipertexto.
8. **Internet**: Red de computadoras alrededor de todo el mundo que comparten información unas con otras por medio de páginas o sitios.
9. **PHP: *(Hypertext Pre-processor)*** Es un lenguaje de programación usado generalmente para la creación de contenido para sitios web. PHP es un acrónimo recurrente que significa Preprocesador de Hypetexto PHP.
10. **RSS:** Documento con sintaxis XML con contenido web que puede ser publicado de nuevo por otros sitios o descargados periódicamente y presentado a los usuarios.
11. **SCRIPT:** Cada uno de los programas, aplicaciones o trozos de código creados con el lenguaje de programación JavaScript. A veces se traduce al español directamente como "guión", aunque script es una palabra más adecuada y comúnmente aceptada.

12. **SQL:** ***(Structured Query Language)*** Lenguaje de Consulta Estructurado es un lenguaje declarativo de acceso a bases de datos relacionales
13. **SSL:** ***(Secure Sockets Layer)***. Protocolo diseñado por la empresa Netscape para proveer comunicaciones encriptadas.
14. **Servicio Web:** ***(Web Service)*** Es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.
15. **TICs:** (***Tecnologías de la Informática y las Comunicaciones)***. Conjunto de servicios y tecnologías integradas a un sistema con el objetivo de mejorar los procesos y la calidad de vida.
16. **XML**: ***(Extensible Markup Language)*** un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium.
17. **YAML:** ***(Ain't Markup Language)*** Lenguaje de serialización diseñado a partir de las estructura de datos más comunes de los lenguajes de programación ágiles. Es muy útil para las necesidades de configuración.

ANEXOS

Anexo 1

Figura 6 Interfaz de la distribución de reservaciones para la Transportación Masiva.

Anexo2

Figura 7 Interfaz de la búsqueda de usuarios del dominio UCI.

Anexo 3

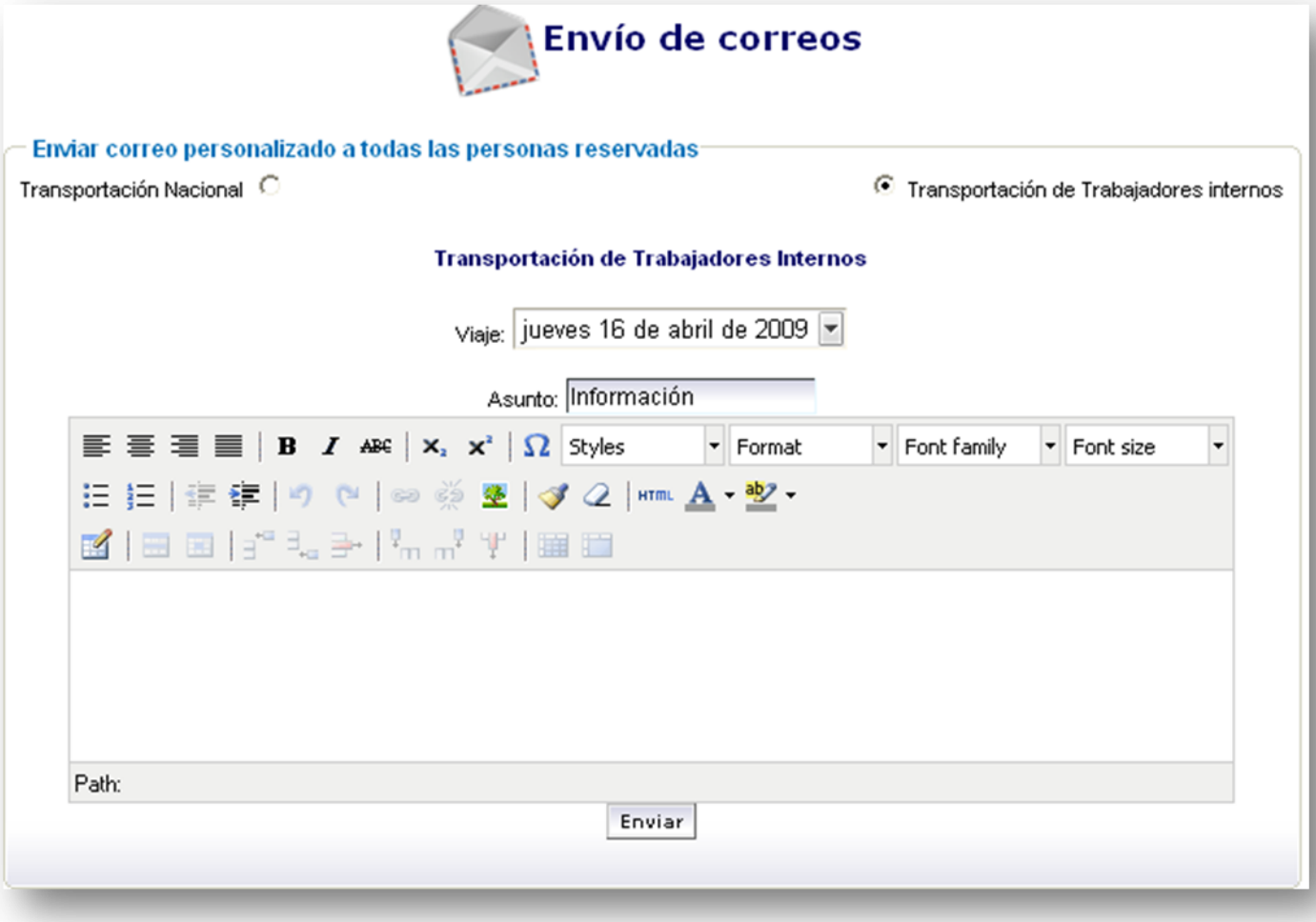


Figura 8 Interfaz para el envío de correos personalizados a personas reservadas y distribuidas en transportes.

Anexo 4

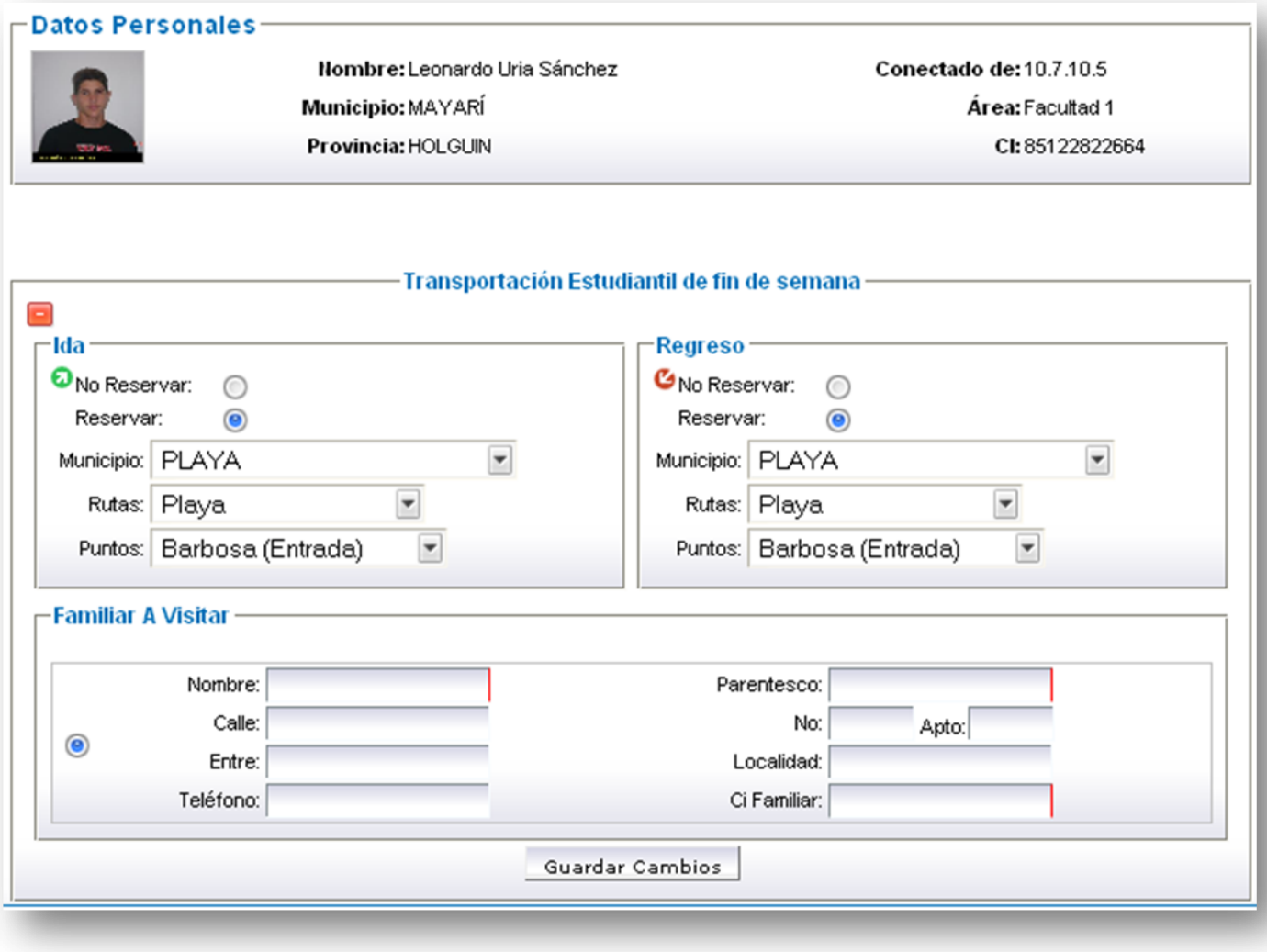
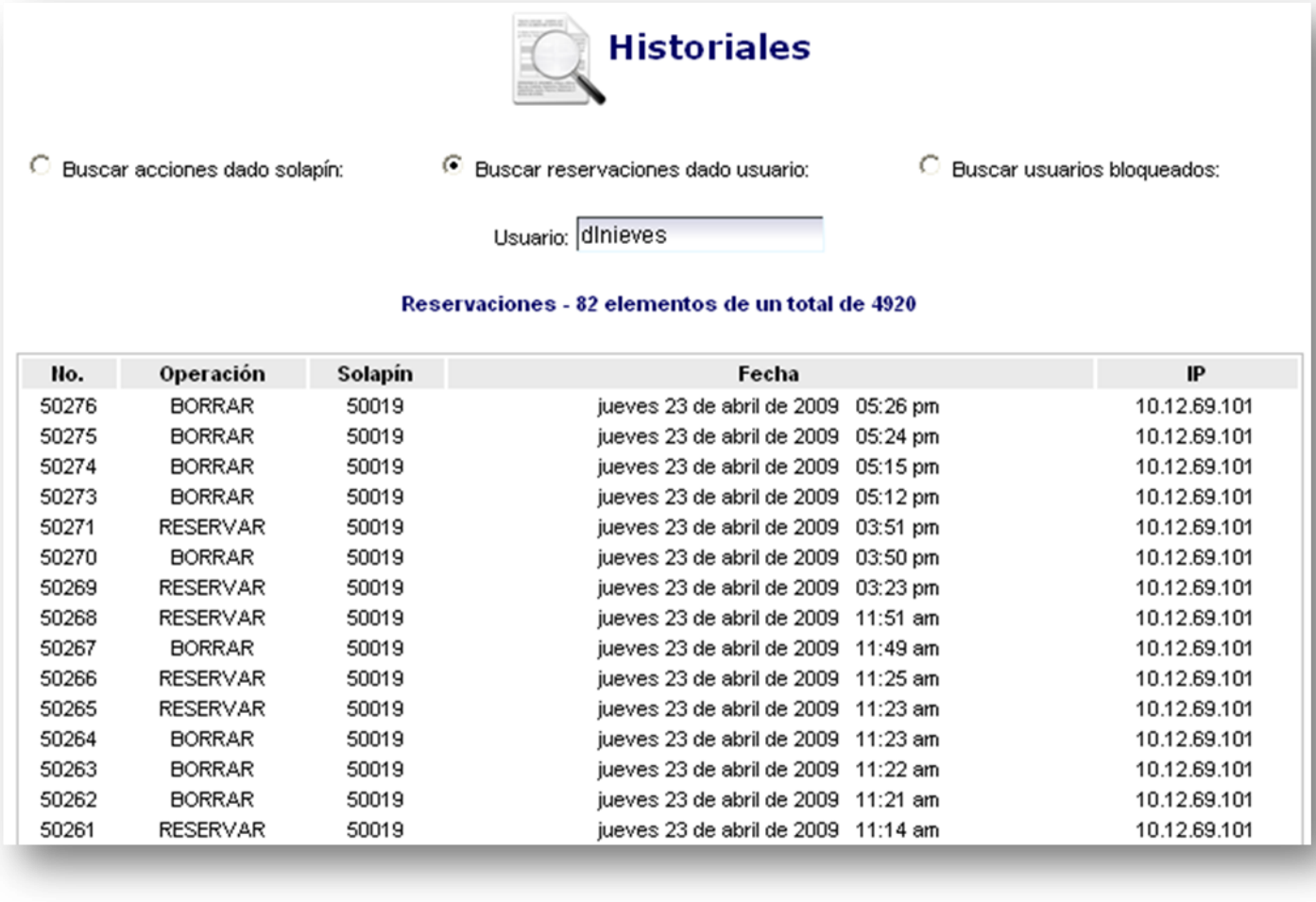


Figura 9 Interfaz de la reservación de la Transportación Estudiantil de Fin de Semana

Anexo 5



No.	Operación	Solapín	Fecha	IP
50276	BORRAR	50019	jueves 23 de abril de 2009 05:26 pm	10.12.69.101
50275	BORRAR	50019	jueves 23 de abril de 2009 05:24 pm	10.12.69.101
50274	BORRAR	50019	jueves 23 de abril de 2009 05:15 pm	10.12.69.101
50273	BORRAR	50019	jueves 23 de abril de 2009 05:12 pm	10.12.69.101
50271	RESERVAR	50019	jueves 23 de abril de 2009 03:51 pm	10.12.69.101
50270	BORRAR	50019	jueves 23 de abril de 2009 03:50 pm	10.12.69.101
50269	RESERVAR	50019	jueves 23 de abril de 2009 03:23 pm	10.12.69.101
50268	RESERVAR	50019	jueves 23 de abril de 2009 11:51 am	10.12.69.101
50267	BORRAR	50019	jueves 23 de abril de 2009 11:49 am	10.12.69.101
50266	RESERVAR	50019	jueves 23 de abril de 2009 11:25 am	10.12.69.101
50265	RESERVAR	50019	jueves 23 de abril de 2009 11:23 am	10.12.69.101
50264	BORRAR	50019	jueves 23 de abril de 2009 11:23 am	10.12.69.101
50263	BORRAR	50019	jueves 23 de abril de 2009 11:22 am	10.12.69.101
50262	BORRAR	50019	jueves 23 de abril de 2009 11:21 am	10.12.69.101
50261	RESERVAR	50019	jueves 23 de abril de 2009 11:14 am	10.12.69.101

Figura 10 Interfaz para la consulta de historiales de la base de datos

Anexo 6

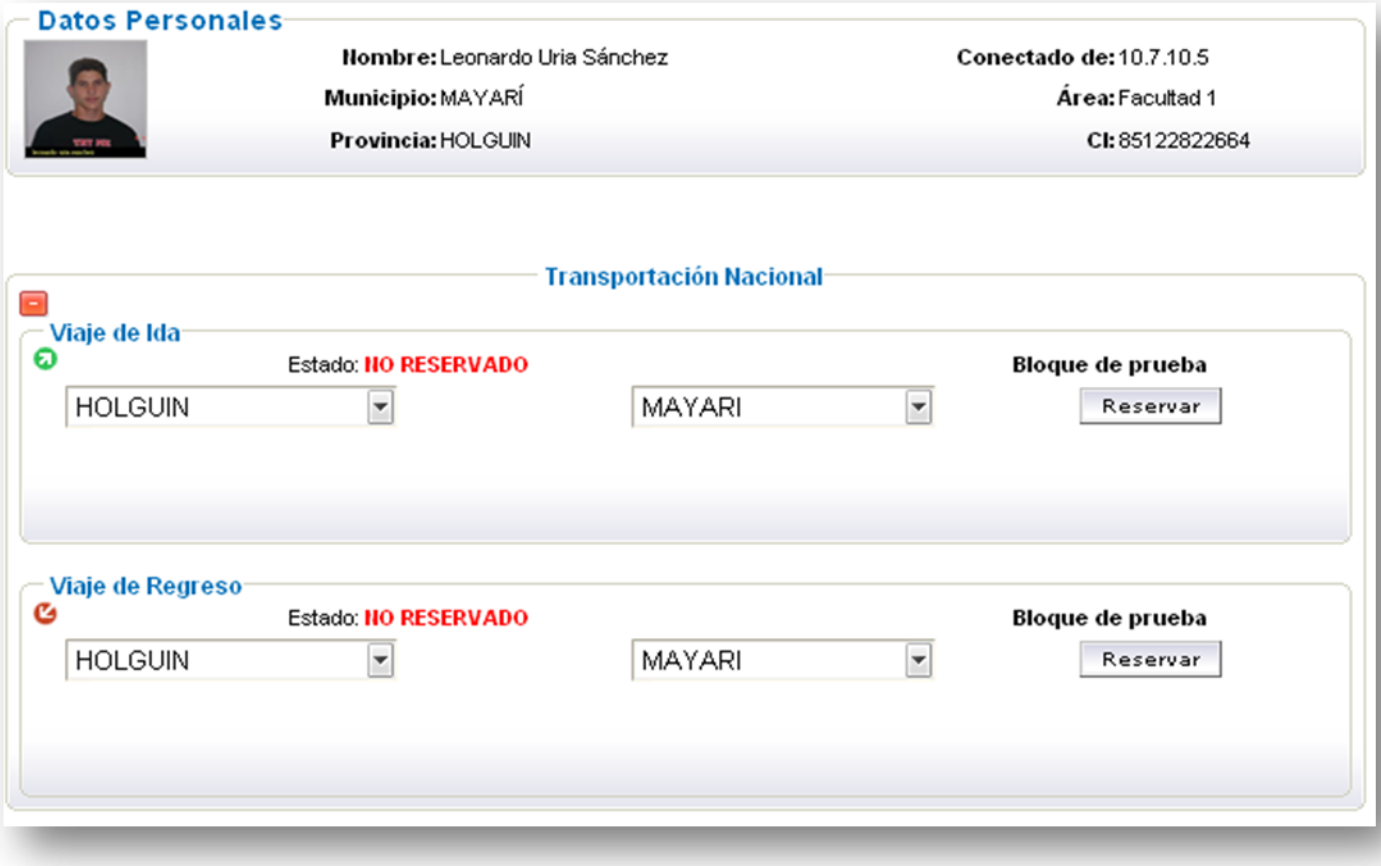


Figura 11 Interfaz de la reservación de la Transportación Masiva

Anexo 7

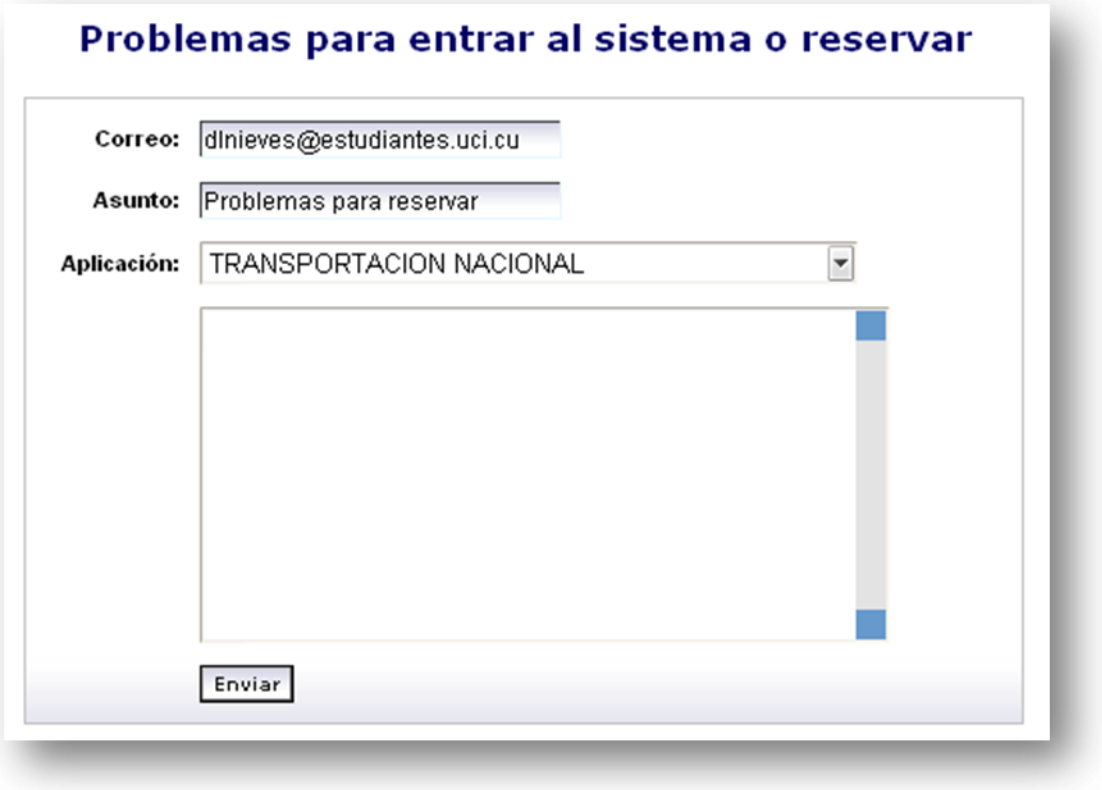


Figura 12 Interfaz para el envío de quejas y sugerencias para cada una de las modalidades de reservaciones.