Universidad de las Ciencias Informáticas

FACULTAD 6

SUBSISTEMA DE VISUALIZACIÓN DE INFORMACIÓN AL VIAJERO

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores

Andrés Mario Hidalgo Brito.

Ramón Antonio Morales Jiménez.

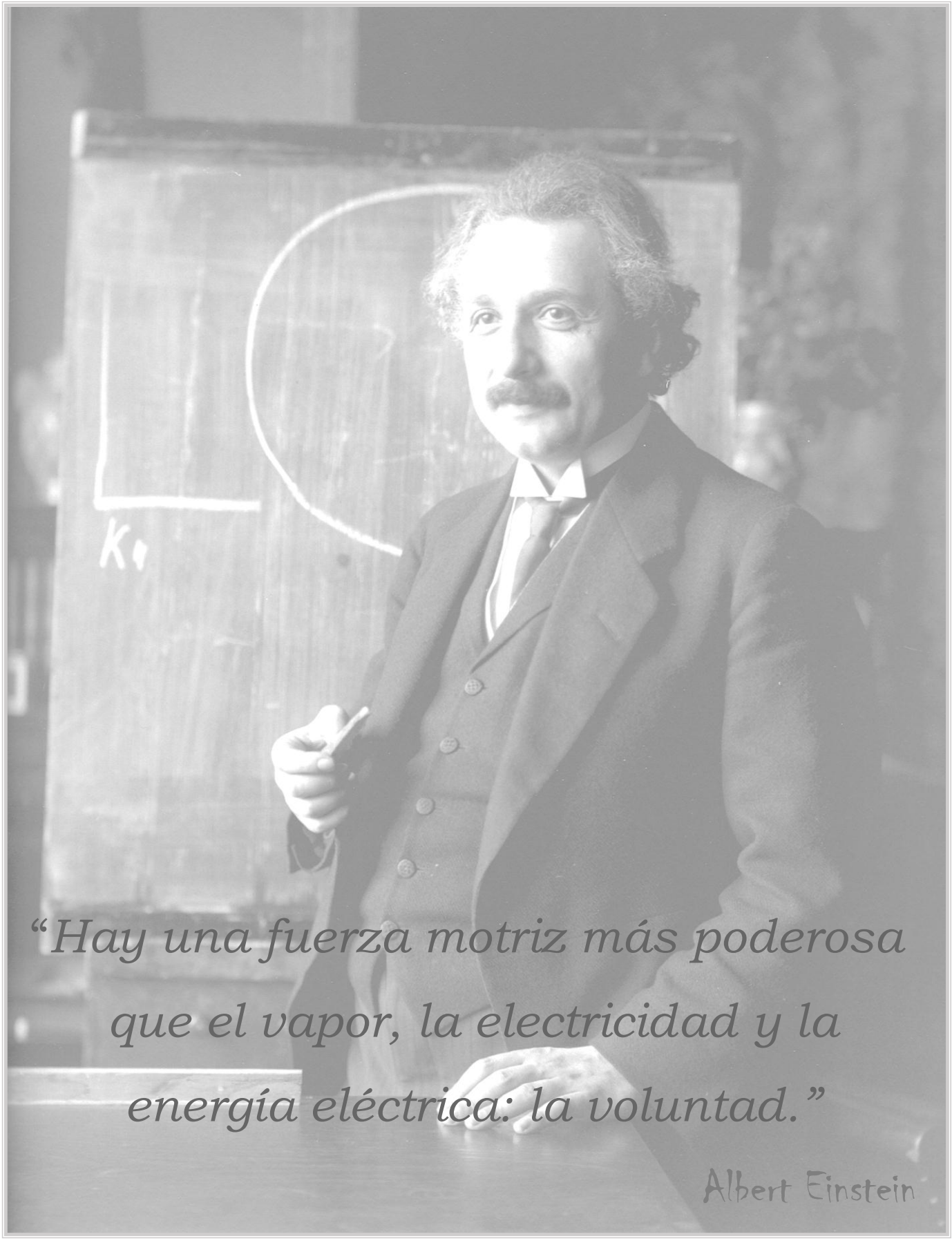
Tutores: MCs. Alexis René Rodríguez León.

Ing. Susana Yaque Rivera.

Ing. Aramis Romero Carballea.

Junio de 2014

"Año 56 de la Revolución"


“Hay una fuerza motriz más poderosa
que el vapor, la electricidad y la
energía eléctrica: la voluntad.”
Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Andrés Mario Hidalgo Brito
[Autor]

Ramón Antonio Morales Jiménez
[Autor]

Ing. Susana Yaque Rivera.
[Tutor]

Ing. Aramis Romero Carballea.
[Tutor]

MCs. Alexis René Rodríguez León.
[Tutor]

DATOS DE CONTACTO

Tutores:

Ing. Susana Yaque Rivera: Graduada de Ingeniero en Ciencias Informáticas en el año 2013. Trabaja en el proyecto Plataforma de Televisión Informativa, PRIMICIA, del centro Geoinformática y Señales Digitales (GEYSED) de la Facultad 6 en la Universidad de las Ciencias Informáticas (UCI).

Correo electrónico: susanay@uci.cu

Teléfono: 837-2244

Ing. Aramis Romero Carballea: Graduado de Ingeniero en Ciencias Informáticas en el año 2012. Trabaja en el proyecto Plataforma de Televisión Informativa, PRIMICIA, del centro Geoinformática y Señales Digitales (GEYSED) de la Facultad 6 en la Universidad de las Ciencias Informáticas (UCI).

Correo electrónico: aramis@uci.cu

Teléfono: 837-2244

MCs. Alexis René Rodríguez León: Graduado de Ingeniero en Ciencias Informáticas en el año 2007. Graduado de Máster en Bioinformática en 2010. Trabaja en el Departamento de Técnicas de Programación y Sistemas Digitales de la Facultad 6. Miembro del grupo de investigación de Bioinformática.

Correo electrónico: arodriguezl@uci.cu

Teléfono: 837-2560

AGRADECIMIENTOS

Andrés:

Quiero agradecer primeramente a toda mi familia por ser mi guía y el apoyo incondicional que siempre necesité los quiero.

También agradecer a mis tutores Susana, Aramis y al master ya que si no fuera por ellos, ahora mismo no estuviera aquí gracias por todo su apoyo.

Bueno sino agradezco a la gente del grupo me matan a ellos gracias por todo Carlitos, el Benny, Javier, Pedro y el resto del grupo, ya que no los puedo mencionar todos son muchos gracias todos.

Agradecer a mis amigos Flavia y Yasmani a los cuales le debo muchísimo desde que entre a la UCI son lo mejor.

También agradecer al personal del proyecto y a los que de alguna u otra forma me han ayudado.

AGRADECIMIENTOS

Ramón:

A los magníficos padres que tengo, por la confianza que han depositado en mí, por educarme, guiarme y ponerme siempre en el camino correcto. Por el esfuerzo dedicado, su apoyo incondicional, sacrificio, preocupación constante, por ser esa parte de mi vida donde siempre encuentro amor y comprensión.

A mi familia, por hacerme un hombre de bien, por siempre estar atenta a cada paso que doy por la vida. Gracias Abuelita Mango, tía Nacy, tío Felix, tío Roly, tío Joseito, Olguita, Jito, Alexis. A mi Abuelo Jiménez, tía Marta y tía Berta, que aunque no se encuentren físicamente entre nosotros, sé que siempre estarán orgullos de su nieto y sobrino. Los amo a todos.

A mi otra mitad (Alienis), por esa forma tan especial de quererme, por su amor incondicional de toda la vida, por siempre estar ahí tanto en las buenas como en las malas, por estos 5 largos años de espera, por soportarme, aguantarme y por toda su comprensión. Te amo.

A mis amigos Ricardo y William, que más que amigos son mis hermanos, por estar siempre ahí para las que sea, por no escatimar tiempo para resolver los problemas que se presentan, ni para fiestar tampoco, por juntos formar un equipo inseparable en el que prima la hermandad.

A mis tutores, no tengo palabras para expresarle mi gratitud, por su apoyo, tiempo, ayuda, dedicación y confianza, que sin eso no hubiera sido posible la realización de esta tesis.

A Aramis, por siempre contar con él para cualquier problema de códigos, y siempre atenderme y auxiliarme ante cualquier duda.

A Susana por orientarme desde los cimientos de la investigación, por sus señalamientos, observaciones, recomendaciones, consejos, por su disposición siempre ante cualquier necesidad en la investigación y por defendernos siempre. Gracias a los dos.

A mis compañeros de grupo y apartamento por compartir su espacio y convivencia: Yoilan, Alexander, Ángel, Aleko, Yoandy, Javier, Denis, Reynaldo, Laffita, Andrés mi compañero de tesis, en fin a todos.

A todos aquellos que desde 1ro hasta 5to año me han acompañado,(El chinito Feria, Ailín, Daniel) los que están aún y aquellos que ya no veo tan seguido, pero que formaron parte importante de mi vida durante estos años de universidad.

A los profesores que de alguna manera contribuyeron a lo largo de este período de enseñanza a mi formación como profesional y a los profes de PRIMICIA que estuvieron siempre dispuestos a ayudar.

DEDICATORIA

A mis padres, mi hermano mis abuelos y toda mi familia por siempre apoyarme en todo. A todo los que estuvieron y están a mi lado.

Andrés

A toda mi familia en especial a mis padres. A todos los que hicieron posible que este sueño se haya hecho realidad. A mi compañera de la vida Alienys.

Ramón

RESUMEN

En la actualidad, el manejo de la información es fundamental para cualquier empresa. Los sistemas informativos son una herramienta necesaria para manejar el flujo de información en terminales de transporte de pasajeros. Estos proporcionan a los clientes los aspectos básicos de la entrada y salida de los medios de transporte a la terminal. En nuestro país la información al viajero que se muestra a través de estos sistemas no se estructura de forma correcta, no siempre está disponible y muchas veces es escasa. El presente trabajo consiste en el desarrollo del Subsistema de Visualización de Información al Viajero, que permite visualizar la información de forma cíclica y constante. El objetivo de este subsistema es lograr la informatización de los procesos de visualización de información en las terminales de pasajeros del país.

Para realizar el Subsistema de Visualización de Información al Viajero se siguieron los pasos que propone el Proceso Unificado de Desarrollo de Software (RUP). Fue utilizando C++ como lenguaje de programación, como lenguaje de modelado UML y una arquitectura de n capas para el diseño de su estructura. Con la realización de este subsistema se contribuye a la mejora de los actuales procesos de visualización de información al viajero, permitiendo mostrar mayor cantidad de información como: las rutas de los viajes en un mapa, información adicional a través de cintillos informativos y la transmisión de varias señales televisivas.

Palabras clave: información; sistema; subsistema; televisión, visualización.

ABSTRACT

Nowadays, the information management is essential for whichever enterprise. The information systems are considered a necessary tool to manage the information flow at Passenger Transport Stations .These systems provide the clients with the basic aspects of the arrival and departure of the transport means toward or from the Station. In our country, the information shown to the travelers through these systems is not correctly structured, it is not always available, and the most of the time it is scarce.

This paper consists in the development of the Subsystem for the Visualization of the Information to the Traveler, which allows visualize the information in a continual and cyclic way. The main target is to computerize the processes for the visualization of the information in the Passengers Station of the country.

Steps proposed by Rational Unified Process (RUP) were followed to carry out this subsystem, C++ was used as programming language, UML as modeling language, and architecture of ¨n¨ layers was used for the design of the structure. The fulfillment of this Subsystem contributes to the improvement of the existing processes for the visualization of the information to the traveler, it allows showing more information such as: the trip routes pointed in a map, additional information through the information stripes, as well as the broadcasting of several TV signals.

Keywords: information; system; subsystem; TV(television); visualization.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Desde el surgimiento del hombre la comunicación ha sido un elemento clave para su evolución y desarrollo social. La necesidad de la actividad comunicativa ha propiciado que se empleen grandes esfuerzos, recursos humanos y materiales al desarrollo de los medios de comunicación. Estos medios desde sus orígenes, han ido avanzando de manera acelerada en búsqueda de su perfeccionamiento, los cuales van desde la prensa, la radio, la televisión hasta Internet. A pesar del auge y desarrollo que ha venido cobrando este último, la televisión sigue siendo un medio masivo y de gran aceptación a nivel mundial.

Cuba no ha pasado por alto la relevancia que presentan los avances tecnológicos y el desarrollo de la televisión. Utiliza estos adelantos para llevar a cabo un proceso de informatización de la sociedad, centrándose en sectores importantes de la economía y los servicios. Con este fin se han creado soluciones informáticas que utilizan la televisión para la transmisión de noticias, audiovisuales y otros tipos de información. En este sentido se han concebido centros de desarrollo de software que se vinculan directamente al trabajo de este tema. La Universidad de las Ciencias Informáticas (UCI) como parte de estos centros, desempeña un papel protagónico en el desarrollo de este tipo de soluciones. Las plataformas de televisión informativas forman partes de los software creados por esta universidad para la transmisión de información, obteniendo productos como la Plataforma de Televisión Informativa PRIMICIA.

Esta plataforma es capaz de transmitir información en distintos formatos mediante un canal de televisión y puede ser personalizada para entidades que tengan la necesidad de mantener informada de forma constante a determinada cantidad de personas. Aunque actualmente se cuente con este tipo de solución, existen instituciones en Cuba como las terminales de ómnibus, trenes y aviones en las que dadas las características del negocio que presentan, no pueden utilizar este tipo de plataforma para brindar información acerca de los procesos y eventos que en ellas ocurren.

En estas terminales el sistema de visualización de información existente no satisface todas las necesidades de los clientes, pues la información que brinda generalmente se encuentra disponible de manera estática, ejemplo: en murales o pizarras informativas, medios en los cuales su ciclo de actualización depende del personal encargado de esta tarea. Además el volumen de información que se muestra en estos medios informativos, provoca que al cliente se le dificulte en ocasiones la obtención de

la información que necesita. Por otro lado, las notificaciones o alertas de los eventos que suceden a diario, no se realizan de forma visual, solo tienen una recepción auditiva. También, en el momento de actualización de la información, se detiene el proceso de visualización, lo que trae como consecuencia que el proceso no cuente con la inmediatez que se necesita.

A partir de la situación problemática planteada anteriormente se formula como **problema a resolver**: la información que se brinda a los clientes en las terminales de pasajeros de Cuba no tiene una estructura que le permita visualizarla de forma cíclica y constante.

Teniendo en cuanta el problema planteado se define como **objeto de estudio**: el proceso de desarrollo de sistemas de información al viajero. Enmarcado en el **campo de acción**: procesos de visualización en los sistemas de información al viajero.

Para dar solución al problema se establece como **objetivo general**: Desarrollar el Subsistema de Visualización de Información al Viajero que permita que la información se visualice de forma cíclica y constante para las terminales de pasajeros en Cuba.

Para guiar la investigación se establecen las siguientes **preguntas de investigación**:

¿Cuáles son los elementos principales de los procesos de visualización en los sistemas de información al viajero para terminales de pasajeros?

¿Qué características debe tener el Subsistema de Visualización de Información al Viajero que permita que la información se visualice de forma cíclica y constante?

¿Cómo organizar el proceso de desarrollo del Subsistema de Visualización de Información al Viajero?

¿El Subsistema de Visualización de Información al Viajero desarrollado permite visualizar la información de forma cíclica y constante?

Para dar cumplimiento al objetivo general se definieron las siguientes **tareas de la investigación**:

- Análisis de los sistemas existentes dedicados a la visualización de información al viajero a nivel nacional e internacional para obtener de ellos las características y funcionalidades comunes.

- Selección y caracterización de la metodología y las herramientas a utilizar en el proceso de desarrollo del Subsistema de Visualización de Información al Viajero.
- Definición de los requisitos funcionales y no funcionales para establecer las capacidades y restricciones con que debe cumplir del Subsistema de Visualización de Información al Viajero.
- Realización del diseño ingenieril de los procesos para representar las funcionalidades presentes en el Subsistema de Visualización de Información al Viajero.
- Implementación de las funcionalidades diseñadas para el Subsistema de Visualización de Información al Viajero.
- Diseño y aplicación de las pruebas al subsistema desarrollado para identificar la mayor cantidad de errores y que puedan ser corregidos en el menor tiempo posible.

Durante el desarrollo esta investigación, se utilizaron un conjunto de métodos científicos, para la recopilación y el análisis de la información.

Métodos teóricos:

Análisis Histórico-Lógico: Utilizado para realizar un análisis sobre la evolución que han tenido los sistemas de visualización de información al viajero a nivel nacional e internacional.

Analítico-Sintético: Utilizado para extraer y sintetizar los aspectos más significativos de los sistemas de visualización de información al viajero a partir del análisis de los procesos de visualización de información.

Modelación: Utilizado para crear abstracciones de forma que se explique lo que en realidad se desea representar a través de los modelos. Se hace visible en el trabajo al crear modelos como el de implementación.

Capítulo 1: Procesos de visualización de información al viajero.

1.1. Introducción

En este capítulo se abordan términos fundamentales asociados al tema de investigación, los cuales permiten un mejor entendimiento del dominio del problema. De forma general se describe el proceso de visualización en los sistemas de información al viajero, atendiendo al objeto de estudio planteado. Para finalizar se realiza el análisis de algunas soluciones existentes a nivel nacional e internacional y se selecciona la metodología y las herramientas a utilizar en el desarrollo de la aplicación.

1.2. Términos asociados al dominio del problema

Visualización de Información

La visualización de información es la disciplina que se encarga de la representación visual de contenidos mediante el uso de diagramas, gráficos y esquemas. Estas representaciones facilitan la asimilación, interpretación, transformación y comunicación de los contenidos (Pérez-Montoro, 2009).

Los autores Stuart K. Card, Jock Mackinlay y Ben Schneiderman, definen la visualización de la información en el considerado primer libro sobre visualización de información aterrizado al campo de la informática como: *"El uso de representaciones visuales e interactivas de datos abstractos, soportadas por un computador para amplificar el conocimiento"* (Card, y otros, 1999).

Se puede puntualizar que la visualización de información aplicada al campo de la informática, es el proceso de transformar información en representaciones visuales como diagramas, gráficos y esquemas, soportadas por una computadora, de modo tal que le permita al usuario la asimilación e interpretación de los contenidos.

Sistema Informático

Un sistema informático es el conjunto de componentes de software y hardware interrelacionados que permiten la interacción con los usuarios para lograr un objetivo común. Durante la concepción de un sistema informático es importante tener en cuenta que las partes interconectadas funcionan como un todo,

cualquier modificación de los componentes afecta el correcto funcionamiento del sistema propiamente dicho, por lo que también hay que tener claro la disposición de los nuevos elementos y sus funciones (Berenguer, 2009).

Puede definirse que sistema informático es un conjunto de software y hardware con elementos, componentes o funcionalidades soportados por un computador que trabajan como un todo para realizar un objetivo común.

Subsistema

Parte importante de un sistema que a su vez tiene las características de un sistema. Es un conjunto de elementos ordenados o funciones relacionadas para cumplir un propósito o fin determinado y cuyas partes deben reunir ciertas condiciones de tal manera que se complementen formando el sistema. Por lo general consta de varios componentes. Por ejemplo, puede considerarse como módulo, componente o subrutina que conforman un sistema (Princelton University, 2003).

También puede especificarse como subsistema a un sistema que realiza una función específica para complementar el conjunto de funcionalidades del sistema general al que pertenece. Ejemplo: el subsistema de visualización, destinado sólo a mostrar información gestionada por el subsistema de administración que en conjunto conforman el sistema de información al viajero.

Sistema de visualización de información

Un sistema de visualización de información se puede definir técnicamente como un conjunto de componentes relacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información (Laudon, 2009)

Se puede establecer sistema de visualización de información como: sistema informático (que puede estar compuesto por subsistemas) capaz de mostrar la información a través de un proceso de trasformación de los datos, en representaciones visuales soportadas por un computador para facilitar la interpretación y asimilación por parte del usuario de los contenidos.

1.3. Proceso de visualización de información en terminales de pasajeros

Para comenzar con el análisis de los procesos de visualización de información en terminales de pasajeros se debe partir primero del término Infografía, que tiene como objetivo la transmisión de datos informativos de diferentes maneras, ya sea por medio de dibujos, gráficos, esquemas, estadísticas o representaciones. La infografía ha sido considerada por muchos como la raíz de las artes y el inicio de los procesos de visualización de información (González, 2011).

Con el desarrollo de las tecnologías, esta forma de comunicación se ha apropiado de nuevas herramientas para explicar la realidad informativa existente en el entorno donde se desarrolla el hombre. Es aquí donde juega un papel fundamental la Arquitectura de la Información (AI), considerada como la disciplina encargada del estudio y análisis de contenidos informativos, así como su organización y estructuración en espacios que lo requieran (León, 2008).

Estos espacios van desde lugares abiertos como parques, avenidas y centros urbanos, hasta locales cerrados como las terminales ferroviarias, aeroportuarias y de ómnibus, en las que los procesos de visualización de información juegan un papel fundamental para su funcionamiento. En estas instalaciones la información llega a las personas mediante dos vías, la auditiva y la visual, pero principalmente se apoyan en la utilización de los medios visuales.

La información que se proporciona hace referencias tanto a ubicaciones importantes, dígase departamentos, taquillas de información o puertas de salidas y entrada, como a la información referente a los viajes programados: fecha, origen, destino, hora de salida, hora de llegada, entre otros. Estos datos pueden manejarse de forma manual, representando la información en pizarras o murales llenados y actualizados por el personal encargado de esta tarea. También pueden controlarse por sistemas informáticos capaces de procesarlos, gestionarlos y estructurarlos para su posterior visualización al cliente.

La visualización en estos sistemas generalmente se lleva a cabo por un subsistema, es decir, otro sistema dedicado específicamente a visualizar la información que ya se encuentra registrada en el sistema del cual forma parte. En este subsistema la información se organiza por secciones en la que los datos contenidos en ellas comparten características comunes. Esta agrupación la realiza el sistema con el objetivo de facilitar la comprensión y asimilación de los contenidos por parte de los clientes. Las secciones

comúnmente se dividen en: listado de próximas salidas, próximos arribos y viajes programados para el día; y en cada una de ellas puede mostrarse las horas de salidas, horas de llegadas, fecha, entre otros. Estas secciones cambian su contenido en consecuencia del tiempo de actualización que tengan definido y de si existe algún cambio en la información.

Otra forma de visualizar la información en los sistemas desplegados en las terminales es sin utilizar un subsistema dedicado a este propósito, es decir, que el propio sistema es capaz de, a través de funciones y procedimientos determinados, mostrar la información en una interfaz de cara al usuario. De esta dos formas vistas anteriormente es que se realiza la visualización en los sistemas informáticos encargados de mostrar la información en terminales de pasajeros.

1.4. Análisis de soluciones existentes

Dentro del gran grupo de sistemas para la visualización de información, se encuentran los desplegados en terminales de pasajeros. En la presente investigación fueron analizados algunos de estos sistemas, con el objetivo de obtener las características principales y los elementos más generales de su funcionamiento.

Ámbito internacional

INFO-pass

INFO-pass es un sistema de información al pasajero perteneciente a un grupo tecnológico de capital privado y con presencia internacional que tiene su sede en Madrid, España. Este sistema está destinado al transporte ferroviario, y combina texto, video y audio. Proporciona información en pantallas o en monitores interiores situados a lo largo del tren donde se encuentre desplegado, como son: la estación en que se encuentra el tren, la siguiente estación, hora de llegada, entre otras. Cuenta con un módulo que controla todas las pantallas presentes en cada coche y reproduce contenido de video a través de los monitores. También tiene un módulo de localización y comunicación que provee funciones de localización del tren por GPS y comunicación multimedia inalámbrica con una aplicación de escritorio para realizar descargas remotas de contenidos y transmisión electrónica de información. Brinda información relacionada con la estación y la línea, como conexiones con otros medios de transporte e información para personas discapacitadas. Pueden programarse a través de una aplicación de escritorio que provee el

sistema, contenidos de texto, vídeo y audio para su reproducción durante el servicio. Además pueden reproducirse contenidos multimedia para eventos concretos (GMV, 2013).

LINARIA

Linaria es un sistema para la información a pasajeros compuesto por un software estándar de servicio que proporciona la gestión de la información, ejecución de comandos, monitoreo del sistema y administración de archivos. También está compuesto por el software de aplicaciones, que provee: aplicación para la visualización de información, simulador de pantallas, documentación de aplicación y comandos. Linaria es capaz de brindar información en pantallas de televisión situadas en terminales y paradas de ómnibus, a fin de informar a los viajeros sobre las horas de llegada, de salida, retrasos y otras informaciones de tráfico. A diario ofrece a los viajeros información confiable y actualizada sobre sus rutas y los cambios que pueden ocurrir en los viajes que se han planificado. Se encuentra montado sobre la plataforma Windows. (SWARCO, 2012).

Ámbito nacional

Sistema de la Estación Central de Ferrocarriles de Cuba

En el año 1999, fue implementado por el Servicio Informático de Ferrocarriles un sistema para la Estación Central de Ferrocarriles de Cuba, con el objetivo de brindar información, principalmente, sobre la transportación ferroviaria del día y las salidas y llegadas de los trenes a la estación. A continuación se explican algunas de sus características (Oliva, 2007):

-En su implementación se utilizó como lenguaje de programación *Visual Basic* con una programación orientada a eventos.

-La aplicación utiliza *Microsoft Access* como gestor de base de datos, el cual es un software propietario y no es multiplataforma.

-Como interfaz gráfica de salida, la aplicación utiliza una presentación de *Power Point*, los cuales son ajustados según la cantidad de información que se desee mostrar a los clientes. Esto no permite visualizar los contenidos de forma cíclica.

Sistema de la Empresa Cubana de Aeropuertos y Servicios Aeronáuticos

Otro de los sistemas informativos existentes es el que se tiene desplegado en la Empresa Cubana de Aeropuertos y Servicios Aeronáuticos (ECASA) para brindar información a sus usuarios en las diferentes terminales de vuelos existentes en la capital. A continuación se explican algunas de sus características (Oliva, 2007):

-Cuenta con un servidor de base de datos central que contiene la información de los vuelos tanto nacionales como internacionales. Este servidor está desarrollado con SQL Server2000.

-La información que finalmente sale por pantalla es independiente para cada una de las terminales, en dependencia de la programación del día.

-Las pantallas brindan a los clientes la relación de los vuelos que partirán o arribarán a la terminal, además para cada salida o arribo se expone: el número del vuelo, hora de salida y de llegada.

Resultados del análisis

Los sistemas encontrados a nivel nacional e internacional, tienen características que se aproximan a lo que se desea desarrollar como son: transmisión de información en formato texto y/o video en pantallas. Sin embargo no pueden ser utilizados para dar solución al problema de la investigación definido pues estos utilizan software privativo, los cuales por su condición no permiten modificar su código para adaptarse a las exigencias del problema presentado. Por otra parte tampoco permiten mostrar información adicional, lo que dificulta que se brinde toda la información necesaria al viajero. Además sólo se encuentran concebidos para un tipo determinado de terminal de pasajeros, por lo que no resuelven completamente el problema presentado.

A pesar de no utilizar estos sistemas, se consideran y toman en cuenta las funcionalidades y características comunes que poseen como son: mostrar fecha, hora, precio para niño, precio para adultos, origen y destino de los itinerarios. Esto sienta las bases para el desarrollo de las funcionalidades principales del Subsistema de Visualización de Información al Viajero.

1.5. Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y soporte documental que guían el proceso de fabricación de una aplicación informática, arrojando un producto de calidad y en el tiempo requerido (Ivar Jacobson, 2000). De acuerdo a sus características y a los objetivos que persiguen se pueden dividir en dos grupos: ágiles y robustas.

Las metodologías ágiles se basan en una fuerte y constante interacción entre clientes y desarrolladores. Además están orientadas al resultado del producto y no a la documentación (Ivar Jacobson, 2000). Estas características hacen que no se ajusten a las necesidades del equipo de desarrollo, el cual no posee experiencia en el desarrollo de software y no se encuentra en constante interacción con el cliente. Además pueden ocurrir cambios de roles en el equipo de desarrollo por lo que se necesita una buena documentación (Roberth G. Figueroa, 2014).

Las metodologías robustas centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo y en cumplir con un plan de proyecto. Una de las más conocidas y usadas es *Rational Unified Process*[1] (RUP). Esta permite el desarrollo de proyectos a largo plazo y pueden ser utilizadas en proyectos donde las personas no cuentan con experiencia en la producción de software. Además no depende de la presencia constante del cliente y genera una documentación detallada (Roberth G. Figueroa, 2014).

Los autores de RUP destacan que el proceso de software tiene tres características esenciales: está dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental (Ivar Jacobson, 2000). A continuación se describe cómo se reflejarán estas características a lo largo del proceso de desarrollo del subsistema que se propone.

Dirigido por casos de uso: los casos de uso no son sólo una herramienta para especificar los requisitos del sistema, también guían su diseño, implementación y prueba (Ivar Jacobson, 2000). Estos constituyen un elemento integrador y una guía para el trabajo, no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo del producto que se desea, en este

[1] Traducido al español: Proceso Unificado de Desarrollo de Software

caso, del Subsistema de Visualización al Viajero. Basándose en estos casos de uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada caso de uso. De esta forma queda organizado el proceso de desarrollo del Subsistema de Visualización de Información al Viajero.

Proceso centrado en la arquitectura: La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de rendimiento del sistema, su reutilización y capacidad de evolución, por lo que debe ser flexible durante todo el proceso de desarrollo. También la arquitectura se ve influenciada por el sistema operativo y gestor de bases de datos, que en ocasiones estos elementos constituyen requisitos no funcionales del sistema (Ivar Jacobson, 2000). En materia de RUP, además de utilizar los casos de uso para guiar el proceso, se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento del producto.

Proceso iterativo e incremental: mediante este proceso, el trabajo se divide en partes más pequeñas o mini proyectos, logrando un equilibrio entre los casos de uso y la arquitectura durante cada uno de ellos. Cada mini proyecto se puede ver como una iteración de la cual se obtiene un incremento que produce un crecimiento en el producto que se está desarrollando (Ivar Jacobson, 2000). Esto permite alcanzar evoluciones, en las que se van corrigiendo los errores para obtener el producto final.

RUP es la metodología escogida para la generación de los artefactos del proceso de desarrollo del Subsistema de Visualización de Información al Viajero, porque puede ser empleada en proyectos donde las personas no cuentan con experiencia en la producción de software; no depende de la presencia constante del cliente y genera una documentación detallada. Además permite organizar y guiar el proceso de desarrollo y a través de su proceso iterativo e incremental se van obteniendo crecimientos, logrando en cada iteración obtener un producto más completo.

1.6. Herramientas, Técnicas y Tecnologías

Lenguaje Unificado de Modelado (UML)

UML es un lenguaje de modelado visual que se usa para especificar, construir y documentar los artefactos de un sistema de software. Para modelar, UML utiliza diagramas que se pueden clasificar en tres tipos: diagramas de estructura estática que comprenden los diagramas de clases, de objetos y de casos de uso; diagramas de comportamiento entre los que se encuentran los diagramas de actividades, de estados y de interacción y por último los diagramas de implementación que lo integran los de componentes y despliegue (Ivar Jacobson, 2000).

En el desarrollo del Subsistema de Visualización de Información al Viajero se utiliza UML pues permite modelar artefactos conceptuales como son: procesos de negocio y funciones del sistema. A través del uso de este lenguaje de modelado es posible obtener una abstracción del subsistema a desarrollar, facilitándole a los integrantes del equipo de desarrollo participar e intercomunicarse fácilmente.

Herramienta de modelado de software

Visual Paradigm for UML (VP-UML) es una herramienta CASE[2] multiplataforma que utiliza el lenguaje de modelado gráfico UML. Una de las licencias bajo la que se encuentra publicada es gratuita, lo que facilita su utilización sin gastos adicionales. Permite la realización de ingeniería directa e inversa, el modelado de procesos de negocio, de requisitos y de base de datos. Es una herramienta colaborativa que soporta múltiples usuarios trabajando sobre el mismo proyecto y a la vez permite realizar el control de versiones (Paradigm, 2007). Además soporta todo el ciclo de vida del software y permite el desarrollo de aplicaciones con rapidez, facilidad y calidad. También el uso de esta herramienta implica un ahorro de tiempo en el diseño de los artefactos generados por la metodología seleccionada y el equipo de desarrollo se encuentra familiarizado con esta herramienta. Las características antes mencionadas sustentan la elección de VP-UML en su versión 8.0 como herramienta CASE.

---

2 Ingeniería de Software Asistida por Computadora.

Lenguajes de programación

Los lenguajes de programación son estructuras, diseñados para expresar cálculos y procesos que serán llevados a cabo por ordenadores. Están formados por un conjunto de palabras reservadas, símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. El proceso de programación consiste en la escritura, compilación y verificación del código fuente de un programa (Guevara, 2013).

C++

El lenguaje C++ está formado por instrucciones muy explícitas, cuya duración de ejecución puede preverse con antelación, en el momento de escribir el programa. Es un lenguaje rápido en cuanto a tiempo de ejecución, está estandarizado y un mismo código fuente se puede compilar en diversas plataformas. Además es de propósito general, por lo que se puede emplear para resolver cualquier tipo de problema (Luján, 2010).

Por las potencialidades que provee este lenguaje, se utilizará para la implementación del Subsistema de Visualización de Información al Viajero. Otra de las razones que contribuyen a su selección es que el equipo de desarrollo se encuentre familiarizado con este lenguaje de programación.

QML

QML es un lenguaje declarativo basado en Java Script que permite diseñar interfaces de usuario. Se basa en declarar elementos QML en un árbol de objetos que permiten visualizar: textos, gráficos, imágenes y videos; además de poder interactuar con ellos mediante estados, animaciones y transiciones. Estos elementos son un conjunto de bloques gráficos que pueden ser combinados para construir componentes más complejos. Los elementos QML son parte del framework Qt, por lo que pueden ser integrados en software desarrollados con este framework en C++ (Qt-proyect, 2013). Por las características mencionadas anteriormente se selecciona el lenguaje QML para el diseño de las interfaces de usuario del subsistema que se propone, pues permite crear interfaces animadas, transiciones entre ellas y visualizar contenidos en forma de texto imágenes, videos o gráficos.

Framework de desarrollo

Un framework, es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto (Alegsa, 2014).

Qt es un framework de desarrollo de aplicaciones multiplataforma ampliamente usado para el desarrollo de programas con interfaz gráfica, aunque también es utilizado para desarrollar programas no gráficos tales como herramientas de consola y servidores. Utiliza el lenguaje de programación C++ de forma nativa. Presenta varios componentes que facilitan el trabajo a los desarrolladores, como son (Nokia, 2004):

-Las bibliotecas Qt (clases escritas en C++).

-Qt Designer: para diseñar formularios visualmente.

-Qt Assistant: acceso rápido a la documentación.

-Qt Linguist: traducción rápida de programas.

-Qmake: simplifica el proceso de construcción de proyectos en las diferentes plataformas soportadas.

Para la implementación del Subsistema de Visualización de Información al Viajero se seleccionó Qt 5.2.0 como framework de desarrollo. Este fue elegido no solo por todas las funcionalidades que brinda, sino también porque soporta los lenguajes de programación anteriormente establecidos. Además es el framework con el cual el equipo de desarrollo se encuentra familiarizado.

Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado es un programa compuesto por una serie de herramientas que utilizan los programadores para escribir códigos. Las herramientas que normalmente lo componen son las siguientes: un editor de texto, un compilador, un intérprete, herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones (Ovideo, 2009).

Qt Creator es un IDE multiplataforma que proporciona herramientas para diseñar y desarrollar aplicaciones con el framework Qt. Este brinda muchas funcionalidades que pueden acelerar y mejorar el proceso de desarrollo de software. Algunas de las funcionalidades que brinda son (Qt-proyect, 2013):

-Genera todos los archivos necesarios independientemente de si se importa un proyecto existente o se crea uno desde cero.

-El editor de código avanzado ofrece compatibilidad con la edición de C++.

-Ofrece dos editores visuales integrados: *Qt Designer* para la creación de interfaces visuales y Qt *Quick Designer* para el desarrollo de interfaces animadas con el lenguaje QML.

Para la implementación del Subsistema de Visualización de Información al Viajero se eligió Qt Creator 3.0.0 como IDE, por su integración con el framework, los lenguajes de programación seleccionados y por las funcionalidades que posee, lo cual se traduce en facilidad de trabajo para el equipo de desarrollo.

Sistema Gestor de Base de Datos:

PostgreSQL es un sistema de gestión de base de datos de código abierto. Para asegurar la estabilidad del sistema utiliza multiprocesos, permitiendo que si un proceso falla, el sistema continúe funcionando debido a que los demás procesos no se afectan. Garantiza la realización completa de operaciones, el mantenimiento de la integridad de los datos, la independencia entre transacciones y la persistencia de las operaciones. Es adaptable a las necesidades del usuario y posee documentación en varios idiomas gracias a sus numerosas comunidades de desarrollo. Es un gestor de base de datos que aporta estabilidad al sistema y posibilita la consistencia de los datos almacenados. Su código fuente está disponible gratis y es multiplataforma (Guerrero, 2010). Por estas razones se emplea PostgreSQL 9.1 como Sistema Gestor de Base de Datos para el Subsistema de Visualización de Información al Viajero.

1.7. Conclusiones parciales

Una vez realizado el estudio de las soluciones existentes dedicadas a la visualización de información en las terminales de pasajeros, se logró identificar que los datos referentes a las próximas salidas, arribos, origen y destino no pueden obviarse en los sistemas que tengan como objetivo mantener informados a los pasajeros.

La metodología y las herramientas seleccionadas para el desarrollo de la solución propuesta posibilitarán la obtención de un producto bien documentado en un lenguaje común, representando una garantía para la continuidad del proceso de desarrollo del Subsistema de Visualización de Información al Viajero.

De las soluciones existentes a nivel nacional e internacional analizadas se determinó que ninguna cuenta con las características necesarias para resolver la problemática que hoy enfrentan las terminales de pasajeros en Cuba. Por tanto y a favor del objetivo general propuesto para la investigación, se decidió llevar a cabo el desarrollo del Subsistema de Visualización de Información al Viajero.

Capítulo 2: Presentación de la solución propuesta.

2.1 Introducción

En este capítulo se presenta el modelo del dominio con el objetivo de proporcionar una perspectiva visual de los conceptos del mundo real que resultan importantes para el desarrollo del subsistema, sus características y relaciones. Se realiza el levantamiento de requisitos para determinar las funcionalidades y características del sistema que se desea desarrollar. Además se presenta el diagrama de casos de uso del sistema, las descripciones de sus actores y Casos de Uso (CU).

2.2 Modelo de dominio

Un modelo de dominio es una representación visual de clases conceptuales o de objetos reales en un dominio de interés. Utilizando la notación UML, un modelo de dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Los objetos de dominio representan los eventos en el entorno que trabaja el sistema (Ivar Jacobson, 2000).

Como no se encuentran bien definidos los procesos de negocio que tienen que ver con la problemática, se emplea un modelo conceptual o modelo de dominio, el cual permite mostrar los principales conceptos que se manejan en el entorno donde va a estar ubicado el sistema y de esta manera contribuir a la comprensión del negocio.

2.2.1 Conceptos asociados al dominio del problema

Taquillas de información: lugar en el cual se maneja y se brinda la información referente a los eventos realizados en la terminal.

Notificaciones: información que se comunica en cierto momento a los viajeros referente a los eventos de la terminal, ejemplo: atrasos, arribos, estado de los medios de trasporte, información sobre los viajes y otros.

Altavoces: medio sonoro mediante el cual se le hace llegar información a los viajeros.

Itinerario: contiene los datos de los viajes y rutas, ejemplo: destino, origen, precio, tipo y nombre del medio de transporte.

Medios visuales: medios mediante los cuales se muestra información en las terminales.

Audiovisuales: audiovisuales que son trasmitidos para el entretenimiento de los clientes en las terminales, ejemplo: canales televisivos, videos y música.

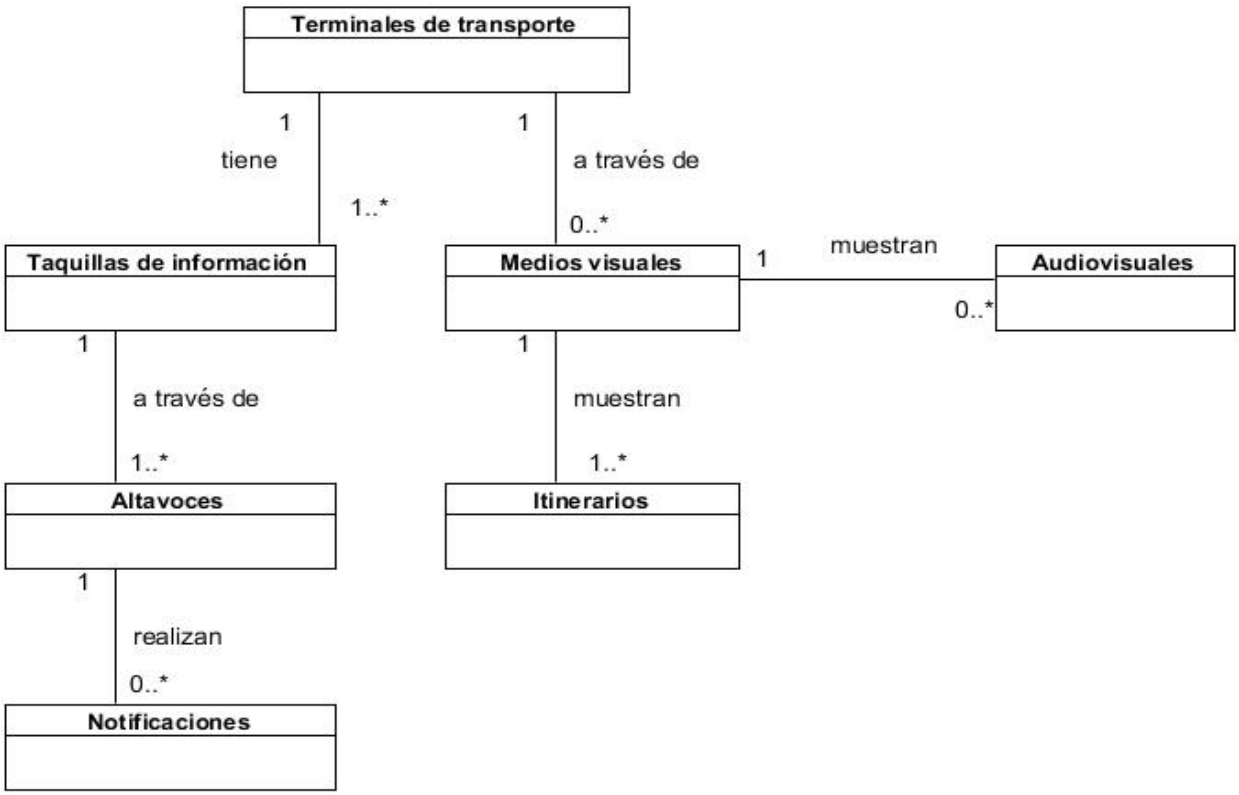


Fig. 1: Modelo de dominio para el Subsistema de Visualización de Información al Viajero.

Las terminales de transporte de pasajeros tienen taquillas informativas en las que se maneja la información de los eventos que suceden en ellas, los cuales se notifican a través de altavoces. También cuentan con medios visuales mediante los que se muestran los itinerarios y audiovisuales.

2.3 Captura de Requisitos

A través del proceso de captura de requisitos se llegan a definir los requisitos de software. Estos requisitos son características que el sistema debe tener o restricciones que debe satisfacer para ser aceptado por el cliente (Wiegers, 2005).

2.3.1 Técnicas de captura de requisitos

Para llevar a cabo el proceso de captura de requisitos existen varias técnicas de recopilación de información. La combinación de varias técnicas es una opción factible para lograr un buen proceso, pues cuando se aplica una sola, pudiera obviarse información importante. A continuación se enuncian las aplicadas en la presente investigación:

Entrevista

La entrevista es una técnica muy efectiva que permite conocer los problemas de los clientes y encontrar requisitos generales. Para aplicar esta técnica es necesario conocer la forma en que se debe realizar una entrevista para lograr una buena comunicación entre el entrevistador y el entrevistado (Toro, 2000). El tipo de entrevista realizada fue no estructurada, la cual fue aplicada a especialistas del centro productivo GEYSED[3] y a cierta cantidad de personas que trabajan en terminales de pasajeros. Con la misma se logró aclarar y resaltar los aspectos fundamentales que no pueden ser omitidos en el producto que se desea desarrollar.

Tormenta de ideas

Esta técnica es usada en grupo para generar nuevas ideas, que puede ayudar a componer una gran variedad de vistas del problema y a formularlo de diferentes formas. Es muy común en los comienzos del proceso de ingeniería de requisitos (Durán Toro, 2000). La aplicación de esta técnica le permitió al equipo de desarrollo definir como llevar a cabo la visualización de la información y de qué forma mostrar los contenidos.

---

[3] Centro de Geoinformática y Señales Digitales.

2.3.2 Requisitos funcionales

RF 1 Visualizar dos señales: el sistema debe permitir visualizar toda la información referente a los viajes por un medio televisivo y por otro reproducir varios tipos de medias y señales externas.

Variable de entrada: no existen

Variable de salida: señales trasmitidas por dos salidas diferentes.

RF 2 Mostrar los itinerarios de viajes: el sistema debe permitir mostrar los itinerarios de viajes con todos sus datos.

Variable de entrada: no existen.

Variable de salida: origen, destino, hora de salida, hora de llegada, precio para niños, precio para adultos, medio de transporte y estado del medio de transporte.

RF 3 Mostrar hora y fecha: el sistema debe permitir mostrar en pantalla la hora y la fecha actual.

Variable de entrada: no existen

Variable de salida: fecha y hora.

RF 4 Mostrar infocinta: el sistema debe permitir mostrar cintillos informativos inmediatos o programados con anterioridad en ambas transmisiones, sin necesidad de afectar las informaciones que se estén transmitiendo.

Variable de entrada: no existen.

Variable de salida: infocintas.

RF 5 Mostrar alertas: el sistema debe permitir mostrar un mensaje de alerta con un tiempo de antelación definido, relacionada con un itinerario.

Variable de entrada: no existen.

Variable de salida: alertas.

RF 6 Reproducir medias: el sistema debe permitir mostrar en pantalla contenidos audiovisuales ejemplo: video, audio y señales externas.

Variable de entrada: no existen.

Variable de salida: medias.

RF 7 Establecer parámetros de la conexión a la base de datos: el sistema debe permitir establecer los parámetros de conexión a la base de datos.

Variable de entrada: nombre, puerto, host, usuario, contraseña y driver.

Variable de salida: parámetros almacenados en el archivo de configuración.

RF 8 Establecer tamaño de pantalla de las transmisiones: el sistema debe permitir establecer el alto y ancho de la pantalla de transmisión de medias e itinerarios a mostrar.

Variable de entrada: dimensiones alto y ancho de ambas transmisiones.

Variable de salida: parámetros almacenados en el archivo de configuración.

RF 9 Establecer efecto de entrada y salida de medias e itinerarios: el sistema debe permitir seleccionar los efectos de entrada y salida de la trasmisión de medias e itinerarios.

Variable de entrada: efecto de entrada y salida de las transmisiones.

Variable de salida: parámetros almacenados en el archivo de configuración.

RF 10 Seleccionar patrón imagen: el sistema debe permitir seleccionar la imagen de patrón para la trasmisión de medias y señales, la cual aparecerá cuando no existan medias o señales que mostrar.

Variable de entrada: imagen seleccionada.

Variable de salida: dirección de la imagen almacenada en el archivo de configuración.

RF 11 Establecer intervalo de tiempo entre los itinerarios: el sistema debe permitir establecer el intervalo de tiempo entre los itinerarios.

Variable de entrada: intervalo.

Variable de salida: parámetro almacenado en el archivo de configuración.

Restricciones: el valor de entrada del campo es en milisegundos con un límite inferior de 5000 milisegundos y uno superior de 600000 milisegundos.

RF 12 Establecer intervalo de tiempo de antelación a las próximas salidas y arribos: el sistema debe permitir establecer el intervalo de tiempo de antelación a las próximas salidas y arribos.

Variable de entrada: intervalo.

Variable de salida: parámetro almacenado en el archivo de configuración.

Restricciones: valor de entrada del campo es en minutos con límite inferior de 1 minuto y superior de 720 minutos.

RF 13 Mostrar rutas: el sistema debe permitir mostrar un mapa con las rutas correspondientes a las dos próximas salidas que existan según el tiempo de antelación definido.

Variable de entrada: tiempo de antelación.

Variable de salida: mapa con las rutas ubicadas.

2.3.3 Requisitos no funcionales

RNF 1 Requisitos de software: el sistema operativo que soportará la solución será Ubuntu/Linux 12.04 o superior.

RNF 2 Requisitos de hardware del Servidor de Transmisión:

- ✓ *Intel Pentium Dual Core* G630 2.7 GHz.

- ✓ 2 GB de RAM[4].
- ✓ Tarjeta de red.
- ✓ Tarjeta exportadora de video: ATI Radeon 5700 o superior.

RNF 3 Restricciones en el diseño y la implementación:

- ✓ Para la modelación del subsistema se utilizará el lenguaje UML y *Visual Paradigm* como herramienta de modelado.
- ✓ El subsistema estará implementado en lenguaje C++.

RNF 4 Interfaces de Comunicación: Se utilizará la red interna de televisión para la transmisión de la señal del canal.

RNF 5 Interfaces de Software: Se reutilizan las funcionalidades brindadas por el *Qt Multimedia Kit* para la transmisión de los audiovisuales.

2.4 Especificación de casos de uso

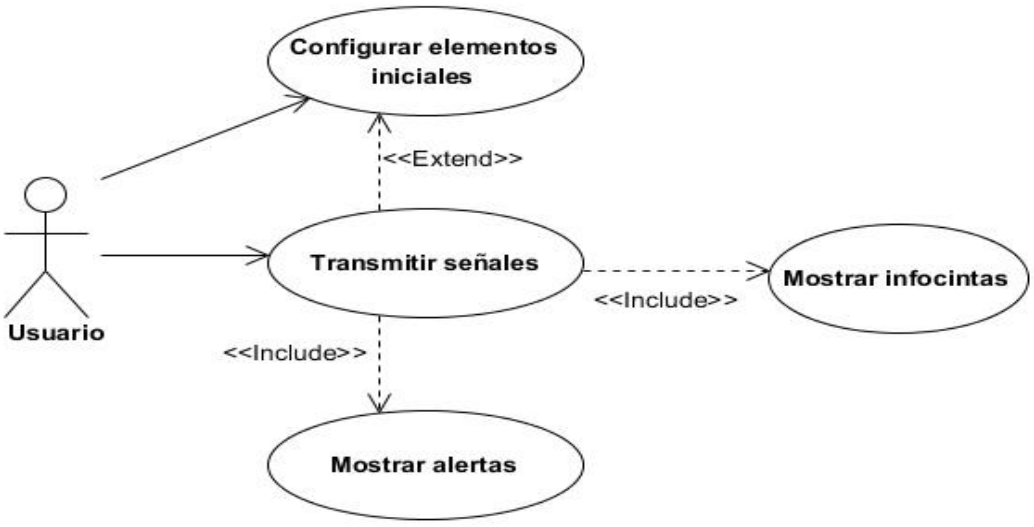


Fig. 2: Diagrama de caso de uso del sistema.

[4] Memoria de acceso aleatorio

Los requisitos funcionales identificados anteriormente, atendiendo a los casos de uso definidos se agruparon en:

Tabla. 1: Agrupación de requisitos funcionales por casos de uso.

Casos de uso	Requisitos funcionales
Configurar elementos iniciales	RF 7: Establecer parámetros de la conexión a la base de datos.
	RF 8: Establecer tamaño de pantalla de las transmisiones.
	RF 9: Establecer efecto de entrada y salida de medias e itinerarios.
	RF 10: Seleccionar patrón imagen.
	RF 11: Establecer intervalo de tiempo entre los itinerarios.
	RF 12: Establecer intervalo de tiempo de antelación a las próximas salidas y arribos.
Transmitir señales	RF 1: Visualizar dos señales.
	RF 2: Mostrar los itinerarios de viajes.
	RF 3: Mostrar hora y fecha.
	RF 6: Reproducir medias.
	RF 13: Mostrar rutas.
Mostrar infocintas	RF 4: Mostrar infocinta.
Mostrar alertas	RF 5: Mostrar alertas.

Tabla. 2: Descripción de caso de uso Configurar elementos iniciales.

Objetivo	Ajustar los elementos a tener en cuenta en las transmisiones, tales como: el idioma, los parámetros de la conexión a la base de datos, el tamaño de pantalla de las transmisiones, los efectos de entrada y salida de medias e itinerarios.	
Actores	Usuario: (Inicia) Configurar elementos iniciales.	
Resumen	El caso de uso se inicializa cuando el usuario ejecuta la aplicación de configuración inicial y el sistema le muestra la .ventana de configuración inicial del subsistema.	
Complejidad	Media.	
Prioridad	Secundario.	
Precondiciones	No existen.	
Postcondiciones	Elementos ajustados.	
Flujo de eventos		
Flujo básico Configurar elementos iniciales		
	Actor	**Sistema**
1.	Ejecuta la aplicación.	
2.		Muestra una ventana con los parámetros a configurar. Los campos inicialmente se muestran con los datos del archivo XML[5].
3.	Establece las propiedades de los elementos tales como: tamaño de las ventanas de ambas transmisiones, patrón de imagen, parámetros de la conexión a la base de datos, anfitrión y otros.	
4.	Selecciona la opción Aplicar.	

[5] Lenguaje de Marcas Extensible del inglés *Extensible Markup Language*.

5.		Guarda en un archivo XML la configuración establecida.
6.		Termina el CU.
Flujos alternos		
4a **Evento** Selecciona la opción Iniciar.		
	Actor	**Sistema**
5a		Guarda la configuración en un fichero XML y comienza a transmitir las señales. Ver descripción de CU **Transmitir señales**.
6a		Termina el CU.

Relaciones	**CU Incluidos**	No existen.
	CU Extendidos	Transmitir señales. Ver descripción de CU **Transmitir señales.**

Tabla. 3: Descripción de caso de uso Transmitir señales.

Objetivo	Mostrar toda la información referente a los viajes por un medio televisivo y por otro reproducir varios tipos de medias y señales externas.
Actores	Usuario: (Inicia) Transmitir señales.
Resumen	Este CU puede ejecutarse desde el CU Ajustar elementos cuando el usuario así lo decida, sin embargo puede ser invocado por el mismo usuario sin pasar por el CU base.
Complejidad	Alta.
Prioridad	Crítico.
Precondiciones	XML con los parámetros de configuración de cada una de las pantallas establecidos.

Postcondiciones	Señales transmitidas.	
Flujo de eventos		
Flujo básico Transmitir señales		
	Actor	**Sistema**
1.	Manda a transmitir las señales.	
2.		Muestra una imagen de presentación en cada medio televisivo.
3.		Comienza a transmitir las siguientes señales por los diferentes medios televisivos: -señal de los itinerarios. -señal de medias, ver sección 1: Medias.
4.		Lee de un archivo XML la configuración de la pantalla (ancho, largo, intervalo entre itinerarios y la cantidad de itinerario a mostrar).
5.		Lee de la base de datos los itinerarios.
6.		Muestran en pantalla la cantidad de itinerarios que fueron definidos en el archivo XML. Muestra las próximas salidas según tiempo definido. Muestra los próximos arribos según tiempo definido. Muestra las rutas de las dos próximas salidas. **Observación**: Este paso se repite hasta que

		no existan itinerarios que mostrar.
7.		Termina el CU.
Sección 1: Medias		
	Actor	**Sistema**
4.		Lee de un archivo XML la configuración de la pantalla (ancho, largo y efecto de transición entre videos).
5.		Lee de la base de datos todas las URL[6] de las medias según la programación para el día.
6.		Comienza a mostrar la primera media o señal según el rango de tiempo programado (hora inicio, hora fin), una vez que termine con esta pasa a la otra y así sucesivamente, mientras existan medias para mostrar.
Relaciones	**CU Incluidos**	Mostrar Infocintas. Ver descripción de **CU Mostrar infocintas.** Mostrar Alertas. Ver descripción de **CU Mostrar alertas.**
	CU Extendidos	No existen.

Tabla. 4: Descripción de caso de uso Mostrar infocintas.

Objetivo	Mostrar todas las infocintas activas e inmediatas.
Actores	Usuario: (inicia) Mostrar Infocintas.

[6] Localizador uniforme de recursos del inglés *Uniform Resource Locator*.

Resumen	El caso de uso se inicializa cuando el usuario ejecuta la aplicación y esta comienza a transmitir la señal.	
Complejidad	Media.	
Prioridad	Crítico.	
Precondiciones	Canal en transmisión.	
Postcondiciones	Infocintas mostradas.	
Flujo de eventos		
Flujo básico Mostrar infocintas		
	Actor	**Sistema**
1.		Lee de la base de datos las infocintas cuyo intervalo de publicación contenga la hora actual.
2.		Comienza a mostrar la primera infocinta, una vez que termine con esta pasa a la otra y así sucesivamente, creando un ciclo mientras existan infocintas para mostrar.
3.		Termina el CU.
Flujos alternos		
2a. **Evento:** Detecta infocinta inmediata.		
4.		Si se está mostrando una infocinta, detiene la visualización de esta. Luego muestra la infocinta inmediata. Regresa al evento 2 del flujo básico Mostrar infocintas.
Relaciones	**CU Incluidos**	No existen.
	CU Extendidos	No existen.

Tabla. 5: Descripción de caso de uso Mostrar alertas.

Objetivo	Mostrar todas las alertas.	
Actores	Usuario (inicia) Mostrar alertas.	
Resumen	El caso de uso se inicializa cuando el usuario ejecuta la aplicación y esta comienza a trasmitir la señal, el sistema lee de la base de datos las alertas.	
Complejidad	Media.	
Prioridad	Crítico.	
Precondiciones	Canal en transmisión.	
Postcondiciones	Alertas mostradas.	
Flujo de eventos		
Flujo básico Mostrar alertas		
	Actor	**Sistema**
1.		Lee de la base de datos todas las alertas y verifica que el itinerario asociado a ella su fecha de salida menos la fecha actual coincida con el tiempo de antelación de la alerta. Si coincide se muestra la alerta.
2.		Termina el CU.
Relaciones	**CU Incluidos**	No existen.
	CU Extendidos	No existen.

2.5 Conclusiones parciales

Con la realización del modelo de dominio, se logró comprender mejor el dominio del problema, pues se identificaron y detallaron los principales conceptos presentes en el entorno donde el sistema trabajará. A través del proceso de captura de requisitos quedaron plasmadas las prestaciones, características y restricciones del Subsistema Visualización de Información al Viajero. En este proceso se determinó que la transmisión de los itinerarios es la función principal del sistema. Además dio la posibilidad de identificar los casos de uso del sistema, lo cual siguiendo la metodología RUP, es de suma importancia en el desarrollo de software.

Capítulo 3: Diseño del Subsistema de Visualización de Información al Viajero.

3.1 Introducción

En este capítulo se describe la solución propuesta en función del objetivo general de la investigación. Se define la arquitectura a usar en el desarrollo del subsistema propuesto. También se exponen los patrones de diseño utilizados. Se presenta el diagrama de clases del diseño del sistema para comprender mejor la estructura del sistema a desarrollar. Además se muestra el diagrama de componentes y por último el estándar de codificación utilizado, con el objetivo de que todo el equipo de desarrollo entienda la implementación del subsistema.

3.2 Arquitectura de software

La arquitectura de software puede ser vista como la estructura del sistema en función de la definición de los componentes y sus interacciones. También puede considerarse como el puente entre los requisitos del sistema y la implementación. Es la base del diseño del sistema a desarrollar, razón por la cual es considerada como plan de diseño del sistema, debido a que es usada como guía para el resto de las tareas del desarrollo (Ivar Jacobson, 2000).

3.2.1. Patrón arquitectónico

Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software. Presenta un esquema genérico demostrado con éxito la solución a un problema particular y recurrente de diseño. El esquema de solución se especifica mediante la descripción de los componentes que lo constituyen, sus responsabilidades, así como la forma en que éstos colaboran entre sí (Ivar Jacobson, 2000).

En el desarrollo del subsistema se seleccionó el patrón n capas. Éste consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subtareas, las cuales se clasifican de acuerdo a un nivel particular de abstracción (Ivar Jacobson, 2000).Para el subsistema se establecieron tres capas,

denominadas: presentación, lógica de negocio y acceso a datos. La capa presentación es la que contiene la interfaz del sistema y tiene el objetivo de mostrar a los usuarios los itinerarios, infocintas, alertas y audiovisuales. La capa lógica de negocio es la que se encarga de extraer los datos de la capa inferior necesarios para realizar el procesamiento que permitirá visualizar la información. La capa acceso a datos es la encargada de obtener de la base de datos la información guardada por el subsistema de administración y que posteriormente será utilizada por la capa lógica de negocio.

3.3 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Además el modelo de diseño sirve de abstracción a la implementación del sistema, y es de ese modo utilizado como una entrada fundamental de las actividades de implementación (Ivar Jacobson, 2000).

3.3.1 Patrones de diseño

Los patrones del diseño son principios generales de soluciones que aplican ciertos estilos que ayudan a la creación de software. Es un par problema-solución, que se puede aplicar en nuevos contextos. Los patrones sugieren algo repetitivo y permiten codificar conocimientos, estilos y principios existentes que ya han sido probados como solución a un problema común (Rojas, 2010).

Patrones generales de software para asignar responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 1999). Para la implementación del Subsistema de Visualización de Información al Viajero, se utilizaron los siguientes patrones GRASP:

Experto: Este patrón asigna la responsabilidad al experto en la información, es decir la clase que contiene los atributos involucrados para llevar a cabo la responsabilidad. En el cual cada responsabilidad está asignada a la clase que cuenta con la información necesaria para realizarla (Larman, 1999). Ejemplo en la clase lectorXML se encuentran los métodos necesarios para acceder a los datos del fichero de configuración del sistema y ésta es la clase encargada de esta tarea.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos (Larman, 1999). Se ve en la clase controladora del sistema la cual es encargada de crear una instancia de la clase acceso a datos.

Bajo acoplamiento: Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas (Larman, 1999). Este patrón se evidencia en la clase de acceso a datos las cual tiene poca dependencia de otras clases.

Alta Cohesión: En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (Larman, 1999). Está presente en la clase controladora del sistema la cual se encarga de manejar la lógica del sistema con ayuda de las demás clases.

Controlador: Un controlador es un objeto de interfaz no destinado al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación (Larman, 1999). Este patrón se ejemplifica en la clase controladora la cual es la encargada de manejar la lógica general de la aplicación.

Patrones GOF[7]

Los patrones GOF proponen soluciones a problemas concretos. Se basan en la experiencia acumulada al resolver problemas reiterativos de programación. Ayudan a construir software basados en la reutilización y a construir clases reutilizables (Mühlrad, 2008). Los patrones GOF utilizados en la implementación del Subsistema de Visualización de Información al Viajero son:

Observador: Permite a los objetos captar dinámicamente las dependencias entre objetos, de tal forma que un objeto notificará a los objetos dependientes de él cuando cambia su estado, siendo actualizados automáticamente (Mühlrad, 2008). Está presente en la clase marcoInfocinta, la cual después de mostrase envía una notificación o señal a la clase controladora para que se eliminen los componentes de la infocinta mostrada.

---

7 Gang of Four. Traducida al español: pandilla de los cuatro.

Singleton: Este patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella (Mühlrad, 2008). Este patrón se evidencia en la clase acceso a datos de la cual se crea una instancia única.

3.3.2 Diagrama de clases del diseño del sistema

Los diagramas de clases del diseño brindan un mayor acercamiento a la forma y al contenido de la solución propuesta. El diagrama de clases del diseño, fue definido, a partir de los diferentes casos de uso del sistema. Este constituye la mayor aproximación al sistema que se está proponiendo, por lo que una buena modelación de las clases del diseño contribuye a que el producto final quede con mejor calidad. El diseño propuesto se estructura en consecuencia con el patrón tres capas las cuales son: presentación, acceso a datos y lógica de negocio (Ivar Jacobson, 2000).

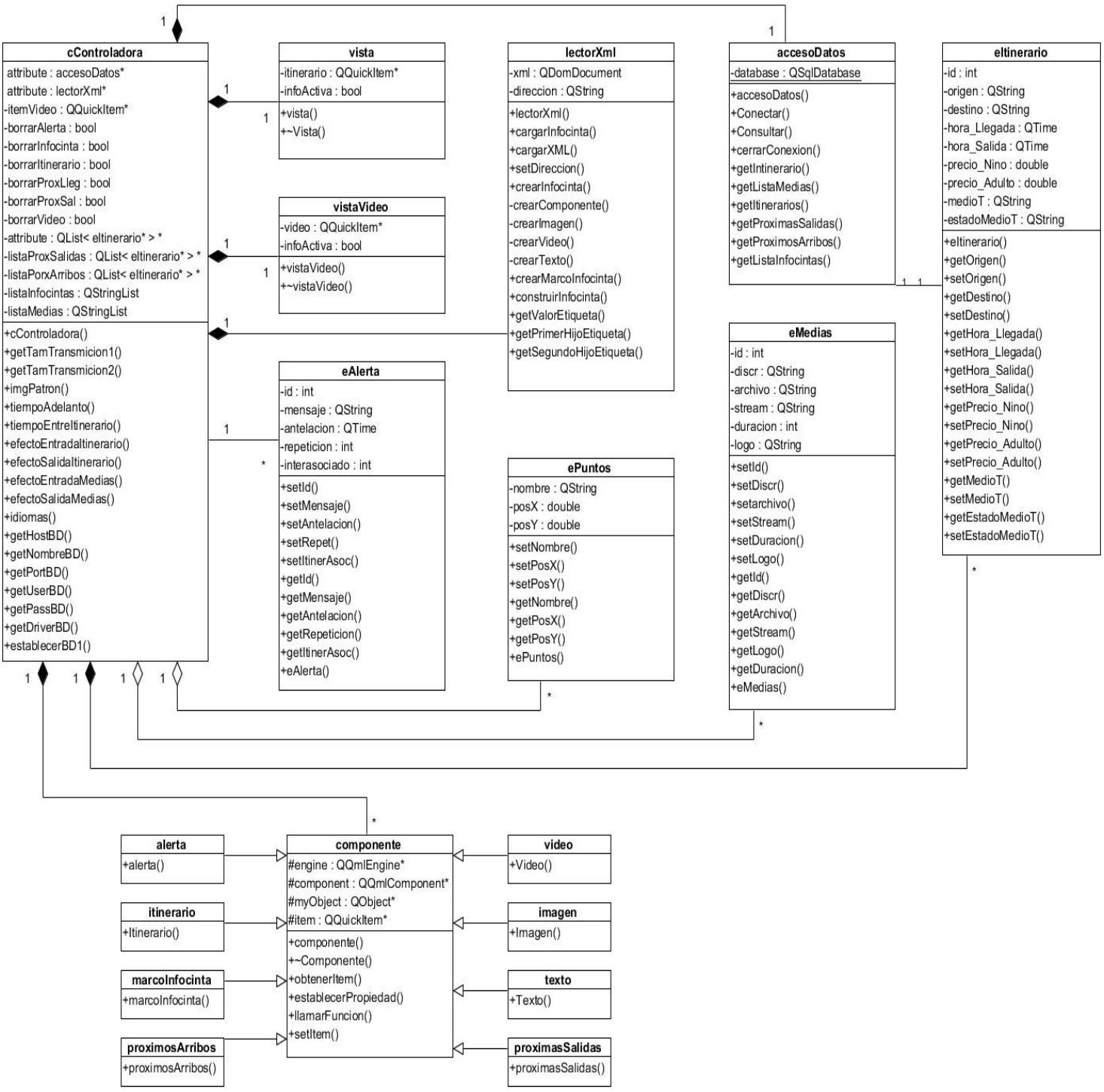


Fig. 3: Diagrama de clase del diseño del sistema.

La clase **vista** es la encargada de mostrar los itinerarios, próximas salidas, próximos arribos, rutas de los itinerarios e informaciones adicionales (infocintas y alertas). La clase **cControladora** contiene la lógica de la aplicación, creando los componentes visuales, obtenidos a través de la clase **accesoDatos** y **lectorXml**. Estos **componentes** visuales pueden ser de tipo: alerta, marcoInfocinta, itinerarios, proximosArribos, proximasSalidas, video, imagen y texto. La clase **elItinerario** es la encargada de representar los datos almacenados en la base de datos correspondientes a los itinerarios, a los cuales se le realizan las operaciones de selección para determinar los próximos arribos, próximas salidas y rutas a mostrar en pantalla.

3.3.3 Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de computo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar, los nodos poseen relaciones que representan medios de comunicación entre ellos (Ivar Jacobson, 2000). A continuación se muestra el diagrama de despliegue del entorno en el que estará la aplicación.

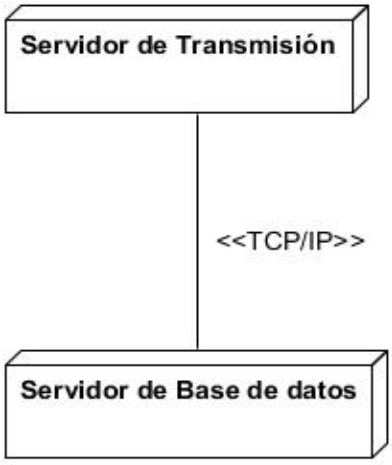


Fig. 4: Diagrama de despliegue del sistema.

3.4 Modelo de implementación

El modelo de implementación describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados. También describe y cómo dependen los componentes unos de otros (Ivar Jacobson, 2000).

3.4.1. Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación. Son otra forma de representar una vista estática del sistema, que muestra la organización y dependencia entre los componentes físicos que se necesitan para ejecutar la aplicación (Ivar Jacobson, 2000).

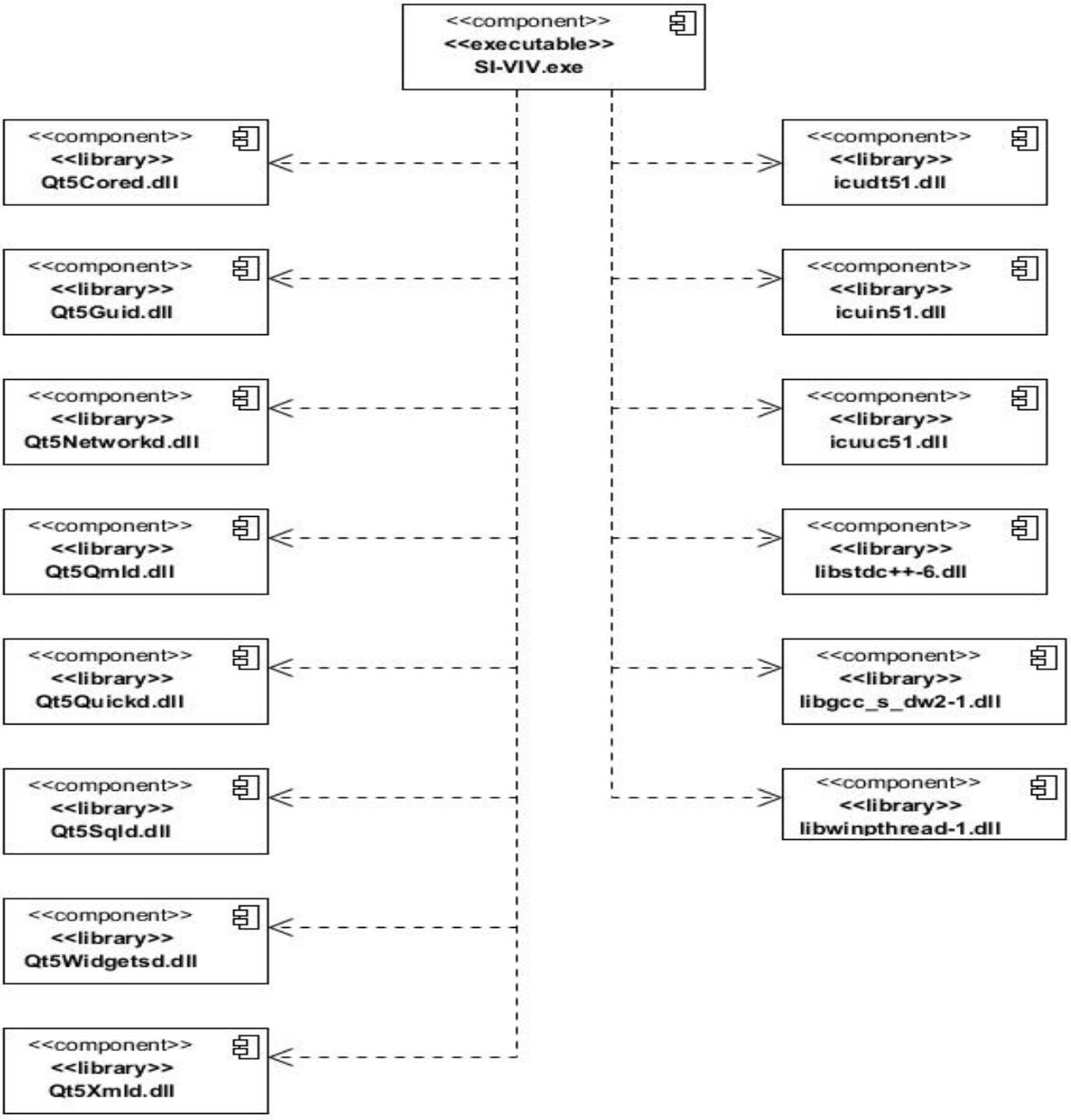


Fig. 5: Diagrama de componentes del sistema.

3.5 Estándar de codificación

Un estándar de codificación es un conjunto de reglas de notación y nomenclatura, específicas de cada lenguaje de programación, que se usan y se siguen durante la fase de implementación de una aplicación y reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores que no son detectados por los compiladores, reduciendo el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos (Remedio, 2007).

El uso de un estándar de codificación es de suma importancia para la implementación del subsistema porque permite reducir el número de errores en él y que toda persona fuera y dentro del equipo de desarrollo, pueda entender el código. Estas son algunas de las características con las cuales cumple el estándar utilizado en la implementación del Subsistema de Visualización de Información al Viajero.

- ✓ Los nombres de las variables, atributos, métodos y clases se escriben en letra minúscula, si está compuesto por más de una palabra, la primera empieza con minúscula y las demás comienzan por letra mayúscula. Ejemplo: atributo, metodoControlador, nombreAtributoCompuesto.

- ✓ Los métodos contienen comentarios los cuales indican que acción realiza el mismo, pero no describe su funcionamiento.

3.6 Conclusiones parciales

Se lograron resolver los problemas de programación que se presentaron en la implementación del subsistema de visualización a través de la aplicación de los patrones de diseños GRASP y GOF. Debido al patrón arquitectónico seleccionado, se logró conformar la estructura del subsistema para su implementación, obteniéndose un sistema flexible y reutilizable. El uso de un estándar de codificación posibilitó la obtención de un código bien estructurado y fácil de entender por el equipo de desarrollo.

Capítulo 4: Diseño y aplicación de pruebas al Subsistema de Visualización de Información al Viajero.

4.1 Introducción

Las pruebas realizadas a un software informático son de vital importancia, aseguran su calidad para que llegue a manos de los clientes cumpliendo todas las funcionalidades requeridas y quedando lo más libre posible de fallas y errores. En este capítulo se describen las pruebas realizadas al subsistema y sus resultados. Estas fueron aplicadas con el objetivo de asegurar, que la aplicación cumple con los requerimientos necesarios para su posterior despliegue.

4.2 Pruebas de software

Una vez generado el código fuente, un software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente. Para esto se emplean las pruebas de software. Una prueba de software puede definirse como el proceso de ejecutar un programa con el propósito de encontrar errores. Se reconoce como una buena prueba a aquella que tiene altas probabilidades de encontrar faltas que no han sido detectadas hasta el momento, su objetivo fundamental es demostrar la existencia de errores, nunca la ausencia de éstos (Pressman, 2005).

4.2.1 Pruebas de rendimiento

Para llevar a cabo las pruebas de rendimiento, la aplicación se ejecutó en tres estaciones de trabajo con características distintas. Se realizó un monitoreo de la Unidad Central de Procesamiento (CPU) y la Memoria de Acceso Aleatorio (RAM), para chequear su comportamiento. Esto permitió establecer límites referentes a la cantidad de componentes visuales a mostrar en pantalla, dependiendo de la potencia del hardware donde se ejecute la aplicación. A continuación se muestran las características de dichas estaciones de trabajo y los niveles de consumo de RAM y CPU durante las pruebas.

Estación de Trabajo #1

- **Sistema Operativo:**
  Ubuntu 12.4
  Gnome 3

- **Hardware:**
  Microprocesador: Intel Core i3 3.30 GHz
  RAM: 4 GB DDR 3

Tabla. 6: Prueba de rendimiento a estación de trabajo #1.

Cantidad de componentes visuales mostrados.	Porciento de uso del CPU	Consumo de RAM
2	11	48
4	15	70
6	21	127

Estación de Trabajo #2

- **Sistema Operativo:**
  Ubuntu 12.4
  Gnome 3

- **Hardware:**
  Microprocesador: Intel Core 2 duo 2.20 GHz
  RAM: 2 GB DDR 2

Tabla. 7: Prueba de rendimiento a estación de trabajo #2.

Cantidad de componentes visuales mostrados.	Porciento de uso del CPU	Consumo de RAM
2	12	69
4	21	100
6	27	132

Estación de Trabajo #3

- **Sistema Operativo:**
  Ubuntu 12.4
  Gnome 3

- **Hardware:**
  Microprocesador: Intel Core 2 duo 2.20 GHz
  RAM: 1 GB DDR 2

Tabla. 8: Prueba de rendimiento a estación de trabajo #3.

Cantidad de componentes visuales mostrados.	Porciento de uso del CPU	Consumo de RAM
2	12	70
4	23	110
6	30	145

Al realizar las pruebas en las tres estaciones de trabajo, la aplicación funcionó correctamente y se detectó que el incremento de la cantidad de componentes mostrados en una pantalla simultáneamente, provoca un consumo extra de recursos. El uso de la memoria RAM aumenta o disminuye en dependencia de la

cantidad de componentes mostrados en pantalla. Cabe resaltar que en la estación de trabajo número tres, la aplicación funcionó más lento que en las anteriores, debido a que la máquina cuenta con menos prestaciones que las primeras. Esto conlleva a un descenso de los FPS[8] y un poco de lentitud.

4.2.2 Pruebas de Caja negra

Son las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo del sistema sin tener mucho en cuenta la estructura interna del software. Es decir se centra principalmente en la estructura externa del sistema y en el cumplimiento de sus funcionalidades (Pressman, 2005).

Estas pruebas permiten encontrar:

-Funciones incorrectas o ausentes.

-Errores de interfaz.

-Errores en estructuras de datos o en accesos a las bases de datos externas.

-Errores de rendimiento.

-Errores de inicialización y terminación.

Diseño de caso de prueba

En la fase de prueba se realizaron dos iteraciones al sistema, en la primera iteración se encontró una no conformidad en la funcionalidad: establecer tamaño de pantalla de las transmisiones. Esta no conformidad se analizó y se le dio solución de manera inmediata. Posteriormente se realizó una segunda iteración de prueba para verificar que se había eliminado y para ver si se encontraban nuevos problemas en el funcionamiento del sistema. Esta última concluyó satisfactoriamente, no se encontró ningún problema en la aplicación. A continuación se presenta el caso de prueba resultante en la segunda iteración.

---

8 Fotogramas por segundo

Tabla. 9: Caso de prueba ajustar elementos iniciales.

Nombre de la sección	Escenario	Acción realizada	Respuesta del sistema	Resultado de la prueba
SC 1: Ajustar configuración de elementos iniciales.	EC 1: Establecer las propiedades de los elementos iniciales correctamente.	El usuario establece las propiedades de los elementos iniciales tales como: intervalo entre itinerarios, patrón de imagen, tamaño de la pantalla de las medias e itinerarios, y parámetros de base de datos.	El sistema guarda la configuración en un fichero XML, ajusta las propiedades guardadas en el XML.	Satisfactorio.

4.2.3 Pruebas de Caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo, se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa, se ejecuten todos los bucles en sus límites y con sus límites operacionales y ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2005).

La prueba del camino básico es una técnica de prueba de caja blanca. Permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2005).

Los pasos a seguir para la realización de las pruebas de caja blanca son los siguientes:

1. Generar el grafo de flujo de datos.
2. Calcular la complejidad ciclomática, V (G).
    V (G) = NA (Número de Aristas) - NN (Número de Nodos) + 2.
    V (G) = P (Nodos predicados) + 1.
    V (G) = Número de regiones.
3. Determinar los caminos independientes o básicos.
4. Generar un caso de prueba para cada camino de ejecución.

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2005).

Método entrarAlerta de la clase cControladora (ver Anexo 2)

Paso 1: Generar el grafo.

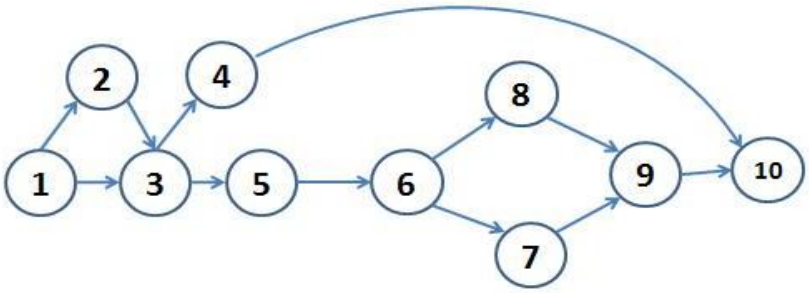


Fig. 6: Grafo del flujo de datos de la función entrarAlerta.

Paso 2: Calcular la complejidad.

V (G) = NA (Número de Aristas) - NN (Número de Nodos) + 2.
V (G) = 12 – 10 = 2 + 2 = 4

Paso 3: Determinar los caminos básicos.

CB 1:1 – 2 – 3 – 4 – 10

CB 2:1 – 2 – 3 – 5 – 6 – 8 – 9 – 10

CB 3:1 – 2 – 3 – 5 – 6 – 7 – 9 – 10

CB 4:1 – 3 – 5 – 6 – 8 – 9 – 10

Paso 4: Caso de prueba para el camino básico.

Caso de prueba para el camino básico 1:

Entrada: borrarAlerta == “true”;

Resultado esperado: borrar el componente alerta usado para liberar espacio en memoria.

Resultado de la prueba: satisfactorio.

Caso de prueba para el camino básico 2:

Entrada: deleteAlerta();

Resultado esperado: eliminar la alerta mostrada con el objetivo de liberar espacio de memoria.

Resultado de la prueba: satisfactorio.

Caso de prueba para el camino básico 3:

Entrada: listaAlertas->first()->getRepeticion() > 0

Resultado esperado: eliminar una repetición de la alerta.

Resultado de la prueba: satisfactorio.

Caso de prueba para el camino básico 4:

Entrada: listaAlertas->isEmpty() == ”true”;

Resultado esperado: actualizar la lista de alertas.

Resultado de la prueba: satisfactorio.

Método entrarVideo de la clase cControladora (ver Anexo 3)

Paso 1: Generar el grafo.

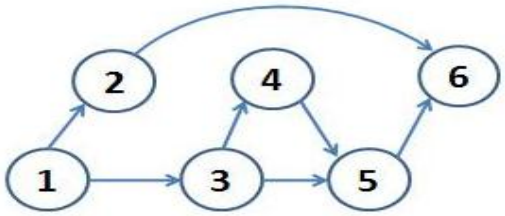


Fig. 7: Grafo del flujo de datos de la función entrarVideo.

Paso 2: Calcular la complejidad.

V (G) = NA (Número de Aristas) - NN (Número de Nodos) + 2.
V (G) = 7 – 6 = 1 + 2 = 3

Paso 3: Determinar los caminos básicos.
CB 1:1 – 2 – 6

CB 2:1 – 3 – 4 – 5 – 6

CB 3:1 – 3 – 5 – 6

Paso 4: Caso de prueba para el camino básico.

Caso de prueba para el camino básico 1:

Entrada: listaMedias.isEmpty()

Resultado esperado: actualiza la lista de la medias.

Resultado de la prueba: satisfactorio.

Caso de prueba para el camino básico 2:

Entrada: borrarVideo == “true”;

Resultado esperado: borrar el componente video usado para liberar espacio en memoria.

Resultado de la prueba: satisfactorio.

Caso de prueba para el camino básico 3:

Entrada: borrarVideo == “false”;

Resultado esperado: construye el componente video para su posterior visualización.

Resultado de la prueba: satisfactorio.

4.2.4 Pruebas de integración

“Aun cuando los módulos de un programa funcionen bien por separado, es necesario probarlos conjuntamente. Un módulo puede tener un efecto opuesto o inadvertido sobre otro módulo, las sub-funciones, cuando se combinan, pueden no producir la función deseada. Una imprecisión aceptada individuamente puede crecer hasta niveles inaceptables al combinar los módulos y los datos pueden perderse o malinterpretarse en la integración” (Natalia Juristo, 2005).

El Subsistema de Visualización de Información al Viajero en conjunto con el Subsistema de Administración conforman un sistema de información al viajero. El subsistema de visualización es el encargado de mostrar la información previamente almacenada en base de datos por el subsistema de administración. Ambos subsistemas trabajan sobre la misma base de datos, siendo esta la conexión fundamental para que juntos conformen el sistema de información al viajero. Para llevar a cabo las pruebas de integración, se utilizó la estrategia integración no incremental, debido a que el sistema en su totalidad cuenta

solamente de dos partes. Las funcionalidades críticas atendiendo a los requisitos funcionales del subsistema de visualización a comprobar son: mostrar itinerarios, reproducir medias y mostrar infocinta. A continuación se describe el proceso de aplicar las pruebas centrándose en las funcionalidades declaradas anteriormente.

- Para el caso de la funcionalidad mostrar itinerarios, se insertaron desde el subsistema de administración varios itinerarios en la base de datos. Estos itinerarios fueron mostrados de forma cíclica y constante en el subsistema de visualización satisfactoriamente. Durante la visualización de estos itinerarios previamente almacenados, se insertaron nuevos itinerarios, los cuales también se mostraron en el subsistema de visualización correctamente.

- Para comprobar que se visualizan las medias, en el subsistema de administración se creó una escaleta[9] con una duración de dos horas. Las medias asociadas a la escaleta se mostraron satisfactoriamente en el subsistema de visualización durante las dos horas programadas.

- En el caso de la funcionalidad mostrar infocintas, se creó en el subsistema de administración una infocinta inmediata, acción que genera el envío de una señal al subsistema de visualización encargado de mostrar la infocinta. Dicha infocinta no se mostró, debido a que los datos enviados en la señal no se interpretaron correctamente por el subsistema de visualización. Para resolver esta no conformidad se realizaron modificaciones en la funcionalidad encargada de capturar e interpretar la señal. En función de corroborar que fue resuelta la no conformidad, se creó otra infocinta inmediata, la cual se mostró en las dos transmisiones del subsistema de visualización de forma satisfactoria.

En general la aplicación de las pruebas a las funcionalidades críticas del subsistema tuvo resultados satisfactorios.

---

[9] Programación de los contenidos audiovisuales para el día. Consta de una fecha, hora de inicio, hora de fin y audiovisuales asociados a ella.

4.3 Conclusiones parciales

Con la aplicación de las pruebas de Caja negra y Caja blanca al Subsistema de Visualización al Viajero, se logró evaluar las funcionalidades y la estructura interna del subsistema. Se detectó que el desempeño gráfico de la aplicación se sustenta sobre la capacidad operacional del microprocesador y el tamaño de la memoria RAM que tenga la estación de trabajo donde se ejecute. A partir de los resultados satisfactorios de las pruebas de integración se determinó que el Subsistema de Visualización de Información al Viajero se integra correctamente con el subsistema de administración.

CONCLUSIONES

Una vez realizado el análisis de las soluciones existentes a nivel nacional e internacional se determinó que ninguna cuenta con las características necesarias para resolver la problemática que hoy enfrentan las terminales de pasajeros en Cuba. A través de este análisis también se logró identificar que los datos referentes a las próximas salidas, arribos, origen y destino, no pueden obviarse en los sistemas que tengan como objetivo mantener informados a los pasajeros. La metodología y las herramientas seleccionadas para el desarrollo del Subsistema de Visualización de Información al Viajero permitieron obtener un producto bien documentado en un lenguaje común, representando una garantía para la continuidad del desarrollo de futuras versiones de este subsistema.

Se lograron resolver los problemas de programación que se presentaron en la implementación del subsistema de visualización a través de la aplicación de los patrones de diseño GRASP y GOF. Por otro lado, debido al patrón arquitectónico empleado, se logró conformar una estructura adecuada, obteniéndose un sistema flexible y reutilizable. El uso del estándar de codificación definido por el equipo de desarrollo, permitió obtener un código bien estructurado y fácil de entender. Con la implementación del Subsistema de Visualización de Información al Viajero se dio cumplimiento a los 13 requisitos funcionales identificados.

Las pruebas de Caja negra y Caja blanca permitieron evaluar las funcionalidades y la estructura interna del subsistema. Con las pruebas de rendimiento se detectó que el desempeño gráfico de la aplicación se sustenta sobre la capacidad operacional del microprocesador y el tamaño de la memoria RAM que tenga la estación de trabajo donde se ejecute. A partir de los resultados satisfactorios de las pruebas de integración se determinó que el Subsistema de Visualización de Información al Viajero se integra correctamente con el subsistema de administración.

Finalmente se considera que con el desarrollo del Subsistema de Visualización de Información al Viajero se ha alcanzado el objetivo general y se cumplieron las tareas trazadas al inicio del trabajo de forma satisfactoria.

RECOMENDACIONES

- Realizar un estudio con el objetivo de buscar alternativas para mejorar el rendimiento visual de la aplicación tratando de eliminar las caídas de FPS que ocurren cuando se cargan muchos componentes visuales al mismo tiempo.

BIBLIOGRAFÍA Y REFERENCIAS BIBLIOGRÁFICAS

Alegsa. 2014. Alegsa. [En línea] 2014. [Citado el: 26 de 4 de 2014.] http://www.alegsa.com.ar/Dic/framework.php.

Berenguer, Abel Días. 2009. *Sistema de Administración y Configuración de la solución de Captura y Catalogación de Medias.* La Habana : s.n., 2009.

Brito, Henry. 2013. Concepto Definición. [En línea] 23 de marzo de 2013. [Citado el: 15 de noviembre de 2013.] http://conceptodefinicion.com/television/.

Erika Camacho, Fabio Cardeso, Gabriel Nuñez. 2004. *Arquitecturas de Software Guía de estudio.* 2004.

GMV. 2013. GMV. *GMV Innovating Solutions.* [En línea] 28 de mayo de 2013. [Citado el: 18 de noviembre de 2013.] http://www.gmv.com/en/Transportation/RailTransport/info-pass.html.

Gracerant, Iván. 2008. Synergix. *Synergix.* [En línea] 10 de julio de 2008. [Citado el: 12 de febrero de 2014.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

Guerrero, Rafael Martinez. 2010. PostgreSQL. *PostgreSQL.* [En línea] 2010. [Citado el: 15 de mayo de 2014.] http://www.postgresql.org.es/sobre_postgresql.

Guevara, Jorge Martínez Ladrón de. 2013. *Fundamentos de la Programacion en Java.* 2013. ISBN-978-84-96-285-36-2.

Ivar Jacobson, Grady Booch y James Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. ISBN: 84-7829-036-2.

Jimenez, Jorge Hontoria. 2011. TipeSoft. *PaaSOS TipeSoft -Servicion empresariales en la nuve.* [En línea] 21 de febrero de 2011. [Citado el: 26 de noviembre de 2013.] http://tipesoft.com/introduccin-a-qml-i/.

José H. Canós, Patricio Letelier y M. Carmen Penadés. 2014. Métodologías Ágiles en el Desarrollo de Software. [En línea] 2014. [Citado el: 5 de 4 de 2014.] http://noqualityinside.com.ar/nqi/nqifiles/XP_Agil.pdf.

Kiccillof, Carlos Reynoso y Nicolás. 2004. Estilos y Patrones en la Estrategia de Microsoft. [En línea] marzo de 2004. [Citado el: 22 de marzo de 2014.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Arquitectura_de_Software/Estilos_y_Patrones_en_la_Estrategia_de_Microsoft.pdf.

Laptev, I. 2009. *Improving object detection with boosted histograms.* 2009.

Larman, Craig. 1999. *UML y Patrones.* Prentice Hall, México : Pearson, 1999. ISBN-0-13-748880-7.

Laudon, K. Laudon y J. 2009. *Sistemsa de Informacion Gerencial.* México : Pearson-Prentice Hall, 2009.

León, Rodrigo Ronda. 2008. *Arquitectura de Información: análisis histórico-conceptual.* 2008. ISSN-1886-8592.

Luján, Sergio Mora. 2010. *C++ paso a paso.* Universidad de Alicante : s.n., 2010. ISBN:84-7908-888-5.

Ministerio de Industria, energía y Turismo. 2013. Television Digital. [En línea] 2013. [Citado el: 15 de noviembre de 2013.] http://www.televisiondigital.es/TelevisionDigital/Paginas/television-digital.aspx.

Montoro, Mario Pérez. 2009. Glossarium. [En línea] 2009. [Citado el: 15 de noviembre de 2013.] http://glossarium.bitrum.unileon.es/Home/visualizacion-de-la-informacion..

Mühlrad, Daniel. 2008. *Patrones de diseño.* 2008.

Natalia Juristo, Ana M. Moreno, Sira Vegas. 2005. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 2005.

Nokia. 2004. Nokia. [En línea] 2004. [Citado el: 25 de noviembre de 2013.] http://qt.nokia.com/products/developer-tools.

Oca, Dorgis Montes de. 2012. *Desarrollo del Módulo de Redacción de Noticias para la Plataforma de Televisión Informativa Primicia v2.0 .* [Tesis de Curso] La Habana : s.n., 2012.

Oliva, Maykel López. 2007. *Sistema Informativo a clientes en Estaciones ASTRO.* La Habana : s.n., 2007.

Oviedo, Universidad de. 2009. Universidad de Oviedo. *Página oficial de la escuela de Ingeniería Informática de la Universidad de Oviedo.* [En línea] 2009. http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf..

Paradigm, Visual. 2007. Sitio de descargas de software. [En línea] 2007. [Citado el: 22 de noviembre de 2013.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/..

Pressman, Roger S. 2005. *Ingeniería de Software. Un Enfoque Práctico.* Universidad Pontificia de Salamanca campus, Madrid : s.n., 2005.

Princelton University. 2003. Thefreedictionary. [En línea] 2003. [Citado el: 16 de noviembre de 2013.] http://encyclopedia2.thefreedictionary.com/subsystem.

Qt-proyect. 2013. Qt-proyect. [En línea] 2013. [Citado el: 25 de noviembre de 2013.] https://qt-project.org/doc/qtcreator-2.5/creator-overview.html..

Remedio, Pedro. 2007. *Estándares de Codificación.* Mexico : s.n., 2007.

Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. 2014. *Metodologías tradicionales vs. metodologías ágiles.* Universidad técnica particular de Loja : s.n., 2014.

Rojas, Juan Carlos Olivares. 2010. *Patrones de Diseño.* Ciudad Mexico : s.n., 2010.

Sanjuán, Olga Pérez. 2007. *El concepto de "television" en sus origenes.* 2007. ISSN: 02103923.

Sommerville, Ian. 2005. *Ingeniería de Software.* Madrid : Pearson Education s.a., 2005. ISBN: 84-7829-074-5.

Stuart K. Card, Jock Mackinlay y Ben Schneiderman. 1999. *Readings in Information Visualization Using Vision to Think.* 1999.

Swarco. 2012. Swarco. [En línea] 2012. [Citado el: 20 de noviembre de 2013.] http://www.swarco.com/latinamerica/Productos/Transporte-p%C3%BAblico/Sistemas-de-informaci%C3%B3n-de-pasajeros/LINARIA..

Toro, Amador Durán. 2000. *Metodología para la Elicitación de Requisitos de Sistemas Software.* Sevilla, España : s.n., 2000.

Wiegers, Karl E. 2005. *Software Requirements.* Segunda. Redmond : Microsoft Press, 2005. ISBN 0-7356-1879-8.

ANEXOS

Anexo 1

Entrevista no estructurada sobre características y cualidades del Subsistema de Visualización de Información al Viajero realizada al especialista y jefe de proyecto Ing. Félix Iván Romero Rodríguez y al especialista Rafael Lorente Miranda. Ambos pertenecientes al proyecto Plataforma de Televisión Informativa, PRIMICIA, del departamento de Señales Digitales del Centro GEYSED.

- ¿Cómo se debe estructurar la información a mostrar?
- ¿Cuántas secciones o pantallas conformarán al Subsistema de Visualización de Información al Viajero?
- ¿Cuáles son los tipos de medias y señales que serán visualizadas?
- ¿Cuáles son los atributos a visualizar en las señales?
- ¿Cuáles son las opciones de configuración para la visualización de los contenidos en el Subsistema de Visualización de Información al Viajero?
- ¿Qué tipo de información adicional se mostrará?
- ¿Cómo se mostrarán las informaciones adicionales?

Entrevista no estructurada realizada a cierta cantidad de personas que trabajan en terminales de pasajeros en Cuba:

- ¿Qué se debe mostrar en un sistema de información al viajero?
- ¿Cuáles son los datos que deben mostrarse referentes a los itinerarios?
- ¿Qué tipo de audiovisuales pueden visualizarse a través del sistema de información?

- ¿Cuál es la frecuencia de actualización de la información?

Anexo 2

```
void cControladora::entrarAlerta(){
 1   if (borrarAlerta) {
 2       delete alerta1;
 2       delete alerta2;
 2       alerta1=0;
 2       alerta2=0;
 2       borrarAlerta=false;
     }
 3   if (listaAlertas->isEmpty()) {
 4       actualizarListaAlertas();
     }else{
 5       alerta1 = new Alerta();
 5       alerta2 = new Alerta();
 5       alerta1->establecerPropiedad("anchors.top","parent.top");
 5       alerta1->establecerPropiedad("direccion","../../../../Recursos/Imagenes/ClearGlass.png");
 5       alerta1->establecerPropiedad("radius",8);
 5       alerta1->establecerPropiedad("z",1);
 5       alerta2->establecerPropiedad("anchors.top","parent.top");
 5       alerta2->establecerPropiedad("direccion","../../../../Recursos/Imagenes/ClearGlass.png");
 5       alerta2->establecerPropiedad("radius",8);
 5       alerta2->establecerPropiedad("z",1);
 6       if (listaAlertas->first()->getRepeticion()>0) {
 7           QString texto= baseD1->getTraducAlert(listaAlertas->first()->getId(),listaIdiomas.first());
 7           alerta1->establecerPropiedad("texto",texto);
 7           alerta2->establecerPropiedad("texto",texto);
 7           emit entrarCompVista2(alerta1,"bajarY");
 7           emit entrarCompVista1(alerta2,"bajarY");
 7           borrarAlerta= true;
 7           listaAlertas->first()->setRepeticion(listaAlertas->first()->getRepeticion()-1);
        }else
 8        deleteAlerta();
 9      connect(alerta2->obtenerItem(),SIGNAL(finish()),this,SLOT(deleteAlerta()));
     }
}10
```

Fig. 8: Código del método entrarVideo de la clase controladora.

Anexo 3

```
void cControladora::entrarVideo(){
    if (listaMedias.isEmpty()) {
        actualizarListaMedias();
    }else{
        if (borrarVideo){
            delete video;
            video=0;
            borrarVideo= false;
        }
        video = new Video();
        video->establecerPropiedad("vidWidth",100);
        video->establecerPropiedad("vidHeight",100);
        video->establecerPropiedad("anchors.horizontalCenter","parent.horizontalCenter");
        video->establecerPropiedad("direccion",listaMedias.at(0));
        video->establecerPropiedad("vVolumen",1.0);
        emit entrarCompVista2(video,"moverEntrar");
        connect(video->obtenerItem(),SIGNAL(videoMalo()),this,SLOT(deleteVideo()));
        connect (video->obtenerItem(),SIGNAL(finishVideo()),this,SLOT(deleteVideo()));
        borrarVideo=true;

    }

}
```

Fig. 9: Código del método entrarVideo de la clase controladora.

GLOSARIO

Fotogramas por segundo: Se refiere a la cantidad de cuadros que contiene un segundo de un video.

Memoria de Acceso Aleatorio: Tipo de memoria donde la computadora guarda información para que pueda ser procesada más rápidamente. En la memoria RAM se almacena toda información que está siendo usada en el momento.

Infografía: Son representaciones visuales, que permiten comunicar y transmitir información (objetos, procesos, sistemas o hechos) de un modo diferente al lenguaje escrito u oral.

Decodificador de video: Software que se encarga de codificar los archivos de audio y video, por ejemplo para almacenarlos sin que ocupen demasiado. También permite decodificarlos para que puedan reproducirse.