Universidad de las Ciencias Informáticas

Facultad 1

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

Propuesta de Subsistema de control de acceso para el Sistema de Planificación y Control del Servicio de Alimentación de la Universidad de las Ciencias Informáticas.

Autores: *Anisley Martínez Romero*

Reinaldo Jesús Companioni Sosa

Tutores: *Ing. Guillermo Gómez Urquiza*

Ing. Damián Cervantes Rodón

La Habana

Junio, 2012

Por este medio declaramos que Reinaldo Jesús Companioni Sosa y Anisley Martínez Romero somos los únicos autores del trabajo titulado Propuesta de Subsistema de control de acceso para el Sistema de Planificación y Control del Servicio de Alimentación de la Universidad de las Ciencias Informáticas y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma y haga el uso que estime pertinente, con carácter exclusivo.

Para que así conste firmamos el presente a los ____ días del mes de junio del año 2012.

_____ _____

Reinaldo Jesús Companioni Sosa Anisley Martínez Romero

_____ _____

Ing. Guillermo Gómez Urquiza Ing. Damián Cervantes Rodón

Resumen

El control de acceso del personal a los comedores de la Universidad de las Ciencias Informáticas (UCI) se gestiona actualmente a través del sistema CONTACC, desarrollado en el año 2007. CONTACC, a pesar de ser un sistema muy útil, tiene algunas limitaciones, por lo que se decide desarrollar el Subsistema de control de acceso a los comedores de la UCI (SCAC), con el objetivo de desarrollar una herramienta multiplataforma que aumente los niveles de organización del proceso de control de acceso a los comedores de la universidad. Para la implementación de SCAC se utilizó: Java como lenguaje de programación, NetBeans como entorno integrado de desarrollo, PostgreSQL, como sistema gestor de base de datos y Spring e Hibernate como marcos de trabajo, respetando los principios de independencia y soberanía tecnológica que rigen la mayoría de las soluciones implementadas por la universidad. SCAC se integra al Sistema de Planificación y Control del Servicio de Alimentación a través de una base de datos central.

Palabras clave: acceso a comedores, control de acceso, Universidad de las Ciencias Informáticas.

Índice

Introducción

Las tecnologías de la información y las comunicaciones (TIC), forman parte de la cultura tecnológica que rodea a la humanidad, y a su vez amplían las capacidades físicas y mentales de las personas, posibilitando así el desarrollo social. El acceso a las TIC ha provocado continuas transformaciones en las estructuras económicas, sociales y culturales, incidiendo en casi todos los aspectos de la vida.

Con el desarrollo acelerado de la informática y las TIC se ha tratado de controlar la entrada y salida a lugares y recursos de forma eficiente. La tecnología utilizada para este control de acceso se ha desarrollado con la aplicación de los propios sistemas informáticos. En la actualidad se encuentran sistemas de control de acceso en diversas áreas, que contribuyen a mejorar el rendimiento y la calidad en la producción, facilitan además el quehacer diario de todos los usuarios que de una forma u otra interactúan con estos sistemas. Las tecnologías aplicadas a los sistemas de seguridad han venido a revolucionar la forma en cómo se administran los accesos y se monitorean las áreas importantes.

La creciente evolución informática en nuestro país, le ha permitido a varias instituciones nacionales la implantación de sistemas de control de acceso. La Universidad de las Ciencias Informáticas (UCI) es una de las mayores universidades del país, en extensión y en personal, lo que exige a la universidad el uso de un sistema de control de acceso a los comedores que permita gestionar la entrada a los mismos de sus más de diez mil comensales.

En la Facultad 1 de la UCI se encuentra el Centro de Informatización Universitaria (CENIA), que se encarga de conducir el programa de informatización de la UCI, desarrollar productos, servicios y soluciones informáticas de alto valor agregado, basados en la soberanía tecnológica y que tributen a la Excelencia en los procesos universitarios (Saborit, 2011). CENIA es el encargado de desarrollar el Sistema de Planificación y Control del Servicio de Alimentación, que incluye la informatización del control de acceso a los comedores de la UCI; eliminando en gran medida las deficiencias de CONTACC, aplicación que actualmente se encuentra en explotación. Las principales deficiencias del sistema identificadas en los intercambios con los clientes, usuarios e involucrados fueron:

- No es un sistema que se puede ejecutar en entornos libres, por lo que no se ajusta a la política de migración de la universidad.
- Permite cargar un fichero con la información de los usuarios, sin importar su procedencia y los datos que almacena, esto sucede debido a que la aplicación no verifica que la información sobre los comensales esté actualizada, sea verídica o no haya sido alterada.

- No registra en base de datos la cantidad de comensales asignados a una puerta específica en un evento determinado (desayuno, almuerzo, comida y otros), lo que impide conocer con exactitud el número de personas que fueron asignadas al punto de control.
- No permite realizar una redistribución de los comensales en caso que se vea afectado el funcionamiento de dicha puerta.

Por las dificultades antes mencionadas el **problema de la investigación** estará enmarcado en: ¿Cómo mejorar el control de acceso del personal a los comedores de la Universidad de las Ciencias Informáticas fortaleciendo la organización del proceso y cumpliendo las políticas de soberanía tecnológica de la universidad? El **objeto de estudio** está constituido por el proceso de control de acceso. El **campo de acción** que enmarca la evolución del trabajo es el proceso de control de acceso para los comedores de la Universidad de las Ciencias Informáticas. Para dar solución a la problemática como **objetivo general** se plantea: Desarrollar una propuesta de subsistema que sea multiplataforma para el control de acceso del personal a los comedores de la Universidad de las Ciencias Informáticas, fortaleciendo la organización del proceso y cumpliendo las políticas de soberanía tecnológica de la universidad.

Tareas de la investigación:

1. Definición de los conceptos asociados al proceso de control de acceso.
2. Caracterización de las tendencias de los sistemas para el control de acceso.
3. Investigación de los antecedentes de las aplicaciones o soluciones similares**.**
4. Estudio del proceso de desarrollo a utilizar, así como las herramientas de modelado.
5. Estudio de los lenguajes de programación a utilizar, así como las herramientas de desarrollo.
6. Descripción de los procesos asociados al control de acceso de las personas a los comedores de la UCI**.**
7. Identificación de los requerimientos asociados a SCAC**.**
8. Diseño de los prototipos de interfaz.
9. Diseño del modelo físico y lógico de la base de datos de SCAC.
10. Definición de los principales elementos de la arquitectura del software.

11. Análisis de la propuesta de arquitectura desarrollada para SCAC.
12. Implementación de las funcionalidades de SCAC para el Sistema de Planificación y Control del Servicio de Alimentación.
13. Diseño de los casos de prueba basado en los requisitos funcionales.
14. Aplicación de los casos de prueba a SCAC.

Métodos científicos de la investigación

Para llevar a cabo la investigación, se emplean métodos teóricos y empíricos.

Métodos teóricos:

- **Análisis Histórico-Lógico:** En la presente investigación se utiliza este método para realizar el estudio del estado del arte e investigar acerca de otras aplicaciones o soluciones similares, los lenguajes, el proceso de desarrollo de software, el marco de trabajo y las herramientas.
- **Modelación:** Permitió realizar el modelado de todos los diagramas, esquemas y prototipos de interfaz asociados al SCAC.

Métodos empíricos:

- **Observación:** Este método es el instrumento que permite estudiar más de cerca el objeto de la investigación, las acciones, causas y consecuencias. Se puede observar cómo funciona el proceso de control de acceso y los principales problemas asociados a este.

Justificación de la investigación:

Con el siguiente trabajo de diploma se quiere lograr:

1. Un subsistema multiplataforma que controle el acceso del personal a los comedores de la UCI, fortaleciendo la organización del proceso y cumpliendo las políticas de soberanía tecnológica de la universidad que influya positivamente en la calidad de los servicios que se brindan a la comunidad universitaria.
2. Una aplicación que presente una interfaz intuitiva al usuario, propiciando una fácil interacción.

3. Recopilación bibliográfica sobre los principales elementos teóricos asociados al control de acceso, brindando a las personas interesadas una fuente de información confiable sobre el tema.
4. Un resumen de la documentación asociada al proceso de desarrollo del sistema, que pueda utilizarse como material de apoyo para futuras mejoras de la aplicación.

Capítulo 1: Control de acceso. Se exponen los elementos teóricos que sustentan el problema científico y los objetivos del trabajo. Se realiza un análisis de las tecnologías y herramientas de desarrollo que se deben utilizar y se justifica la selección de cada una de ellas.

Capítulo 2: Concepción de la propuesta de solución. Se especifican los principales elementos que describen la propuesta de solución, entre ellas los procesos asociados al control de acceso de las personas a los comedores de la UCI, sus requerimientos, los prototipos de interfaz y los modelos de datos.

Capítulo 3: Validación de la propuesta de solución. Se describen las estrategias, niveles, tipos y métodos de pruebas que son aplicados a un software, se define cuales de estas pruebas serán utilizadas para validar la propuesta de solución. Además, se diseñan los casos de prueba como herramienta para probar el funcionamiento del software.

.

Capítulo I Control de acceso

1.1 Introducción

En el capítulo se abordan características y conceptos importantes relacionados con los sistemas de control de acceso. Se presenta una panorámica general de las tendencias actuales de los sistemas de seguridad y la fundamentación de las tecnologías y herramientas que serán usadas para desarrollar el Subsistema de control de acceso a los comedores de la UCI.

1.2 Conceptos asociados al control de acceso

Para una mejor comprensión del término control de acceso se hace necesario citar algunas definiciones escritas sobre el mismo, como son:

El **control de acceso** es el mecanismo que le permite a los propietarios de los recursos definir, gestionar y hacer cumplir la condición de acceso aplicable a cada recurso, el control de acceso normalmente considera la autorización como base para generar la decisión de acceso (Yague, y otros, 2004).

El **control de acceso** es un conjunto más amplio de políticas dentro de un sistema de gestión de identidad que define las normas de todo lo que a un usuario se le permite hacer en el ámbito de una aplicación del sistema (Reed, 2007).

Control de acceso es un dispositivo que tiene por objeto impedir el libre acceso del público en general a diversas áreas que se denominan protegidas donde solo puede haber personal técnicamente capacitado. Deberán definirse los permisos, reglas o privilegios de cada uno de los que podrán acceder a determinada zona protegida. Estos privilegios podrán depender de la categoría o rango de la persona dentro de la empresa, de su función, de un determinado horario en el que puede ingresar o salir (Cosentino, 2012).

El **control de acceso** constituye una poderosa herramienta para proteger la entrada a una web completa o solo a ciertos directorios concretos e incluso a ficheros o programas individuales. Este control consta generalmente de dos pasos.

- En primer lugar, la autenticación, que identifica al usuario o a la máquina que trata de acceder a los recursos, protegidos o no.

- En segundo lugar, procede la cesión de derechos, es decir, la autorización, que dota al usuario de privilegios para poder efectuar ciertas operaciones con los datos protegidos, tales como: leerlos, modificarlos o crearlos (Álvarez, 2012).

El **control de acceso** es el proceso de autorizar a los usuarios, grupos y equipos a tener acceso a los objetos de la red. Los conceptos claves que componen el control de acceso son: permisos, derechos de usuario y auditoría de objetos (Microsoft, 2012).

Los conceptos antes citados convergen en puntos que definen claramente los sistemas de control de acceso, como son: la identificación de usuarios y sus niveles de acceso a lugares y/o recursos para evitar eventos no deseados. Pero el concepto dado por el Ing. Luis Rodríguez Barzosa se asocia y fundamenta el objetivo de la investigación: el control de acceso es un mecanismo que en función de la identificación ya autenticada permite acceder a áreas, datos o recursos (Rodríguez, 2011).

1.3 Definiciones relacionadas con el control de acceso

Algunos de los conceptos asociados al control de acceso son:

Acceso: Es la entrada o paso a un lugar, dígase institución, compañías, o simplemente a páginas web dentro de un sitio, o a otros servicios informáticos (Española, 2007).

Autenticación: Método que permite comprobar de forma segura la identidad de un usuario (Lucena, 2003).

Autorización: Sucede después de la autenticación y usa atributos o derechos, asociados con la identidad digital para determinar a qué recursos puede acceder dicha identidad digital. Se basa en políticas de control de acceso (reglas para especificar quién puede acceder, a qué recursos), modelos (formalismos para describir las políticas) y mecanismos (verificar la solicitud de acceso de un usuario para conceder o denegar el acceso) (Reed, 2007).

Identificación: Se usa para designar al acto de identificar, reconocer o establecer los datos e información principal sobre una persona. La identificación, además de ser un acto o acción a realizar en determinadas situaciones, también puede ser el nombre que llevan determinadas documentaciones que tienen por objetivo justamente establecer la identidad de una persona o individuo (Lucena, 2003).

1.4 Modelos de control de acceso

Los modelos que tradicionalmente han existido para llevar a cabo el control de acceso son:

- **El modelo de control de acceso discrecional (DAC):** Es un modelo no orientado al control del flujo de información. Los modelos DAC están basados en la idea de que el propietario de un objeto, su autor, tiene el control sobre los permisos del objeto. Es decir, el autor es autorizado a permitir u otorgar permisos para este objeto a otros usuarios. DAC admite la copia de datos desde un objeto a otro por usuarios autorizados de manera que un usuario puede permitir el acceso para copiar datos a otro usuario no autorizado. Este riesgo puede ser extendido a todo el sistema violando un conjunto de objetos de seguridad (Rodríguez, 2008).

- **El modelo de control de acceso mandatorio (MAC):** Es un modelo donde todos los sujetos y objetos son clasificados basándose en niveles predefinidos de seguridad que son usados en el proceso de obtención de los permisos de acceso. Para describir estos niveles de seguridad todos los sujetos y objetos son marcados con etiquetas de seguridad que siguen el modelo de clasificación de la información militar (desde “desclasificado” hasta “alto secreto”), formando lo que se conoce como política de seguridad multinivel (Rodríguez, 2008).

- **El control de acceso basado en tareas (TBAC):** Permite controlar el acceso en entornos representados por flujos de trabajo. El modelo TBAC extiende los tradicionales modelos de control basados en sujetos/objetos incluyendo aspectos que aportan información contextual basada en las actividades o tareas. El control de acceso en TBAC es garantizado por medio de “Etapas de autorización”. Las “Etapas de autorización” son un concepto abstracto introducido por TBAC para modelar y manejar un sistema de permisos relacionados con el progreso de las tareas o actividades dentro del contexto de un flujo de trabajo (Yuan, y otros, 2005).

- **Control de acceso basado en identidad (IBAC):** En este caso los permisos de acceso se asocian directamente al identificador del sujeto, es decir, el nombre del usuario. Por tanto se garantiza el acceso al recurso solo cuando exista dicha asociación. Con IBAC no se asocian etiquetas de seguridad a los usuarios, por lo que este es principalmente un mecanismo de control de acceso discrecional. Un ejemplo de IBAC son las Listas de Control de Acceso (ACL), encontradas comúnmente en sistemas operativos y servicios de seguridad en red.

Una ACL contiene los identificadores de los usuarios junto con sus derechos de acceso a un recurso determinado, como leer, escribir y ejecutar. Esta estructura básica de autorización únicamente

extiende el concepto de autenticación, ya que si el usuario no puede autenticarse correctamente ante el guardián del recurso, su solicitud de acceso es denegada. Entre más usuarios soliciten el acceso a un recurso más identificadores contendrá la ACL, lo que dificulta el manejo de estas listas y las hace una alternativa poco escalable. Por otra parte, la decisión de control de acceso no depende de alguna función o característica de la organización a la que pertenece el usuario sino solamente de los identificadores, así que su uso es inapropiado a nivel empresarial (Yuan, y otros, 2005).

- **Control de acceso basado en mallas (LBAC):** En este caso una colección totalmente ordenada de etiquetas de seguridad se combina con un grupo de categorías y forman una malla. Con ellas se obtienen un conjunto de clases de seguridad que varía desde la más baja a la más alta. Generalmente el modelo LBAC es manejable cuando existe un número relativamente pequeño de etiquetas de seguridad y categorías, por lo que es solo efectivo para ciertos escenarios de seguridad poco granulados y carentes de flexibilidad y escalabilidad. LBAC es un mecanismo de control de acceso obligatorio (Yuan, y otros, 2005).

- **Modelo de control de acceso basado en rol (RBAC):** Es un modelo que describe un grupo de usuarios que pueden estar actuando bajo un conjunto de roles y realizando operaciones en las que utilizan un conjunto de recursos. En una organización, un rol puede ser definido como una función que describe la autoridad y responsabilidad dada a un usuario en un instante determinado. El modelo RBAC incluye un conjunto de sesiones donde cada sesión es la relación entre un usuario y un subconjunto de roles que son activados en el momento de establecer dicha sesión. Cada sesión está asociada con un único usuario, mientras que un usuario puede tener una o más sesiones asociadas.

  Los permisos disponibles para un usuario son el conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario, sin tener en cuenta las sesiones establecidas por otros usuarios en el sistema. RBAC añade la posibilidad de modelar una jerarquía de roles de forma que se puedan realizar generalizaciones y especializaciones en los controles de acceso y se facilite la modelización de la seguridad en sistemas complejos (Rodríguez, 2008).

Los modelos DAC e IBAC son poco utilizados, pues son excesivamente simples y permiten un conjunto de operaciones ilimitadas sobre un recurso. Además de tener como única seguridad la identificación de personas, que no siempre es segura porque puede existir robo de la misma. En cuanto a los modelos MAC y LBAC tienen por inconveniente que son muy rígidos, debido a que las decisiones de seguridad la toma el sistema. SCAC utilizará el modelo RBAC, ya que el control de

acceso basado en roles permite expresar de forma sencilla y natural la política de acceso a los recursos de una organización compleja, facilitando las tareas administrativas al separar la asignación de individuos a funciones o perfiles de trabajo, y la definición de políticas de acceso (definición de roles en términos de lo que pueden hacer en el sistema). Permite asimismo, la construcción jerárquica de estas políticas de acceso, por herencia o especialización.

1.5 Tecnologías para el control de acceso

Hoy en día el desarrollo de las tecnologías posibilitan brindar soluciones efectivas, tanto para grandes empresas que requieren máxima seguridad y robustez según sus características, como para empresas de mediana o menor escala que requieren precios económicos y facilidad de uso.

- **Claves por teclado**: Constituye la opción más económica pero la menos segura. Actualmente se encuentran en desuso y hasta el momento no se han hecho aplicaciones donde puedan resurgir como una opción viable (Kathleen Jui, 2005).
- **Tarjetas de banda magnética**: Esta tecnología es una de las más conocidas y usadas. Se utilizan en todos los sistemas de tarjetas de crédito y compra. Su difusión, popularidad y bajo costo la ponen en una posición ventajosa respecto a otras tecnologías, pero es la más vulnerable. La banda magnética de la tarjeta puede ser dañada fácilmente, por lo que hay que manipularla con mucho cuidado para impedir que se raye o se pueda borrar el contenido de la tarjeta (Kathleen Jui, 2005).
- **Tarjetas de código de barras**: Es una tecnología de identificación automática. Un código de barras consiste en una serie de barras adyacentes, paralelas y separadas por pequeños espacios, que permite recoger datos con precisión y rapidez. Los diseños predeterminados de anchura son usados para codificar datos en el código. Para leer información en un símbolo de código de barras, un dispositivo de lectura se desliza a través del símbolo de un lado a otro, mediante dicha operación, la anchura de barras y los espacios son analizados por el decodificador del lector, recuperándose así los datos originales (Kathleen Jui, 2005).
- **Tarjetas de identificación por radio frecuencia (RFID):** La identificación por radio frecuencia es un método de reconocimiento automático sin contacto, es la tecnología más nueva y de más rápido crecimiento en el segmento de identificación automática. RFID permite reconocer de forma instantánea, la localización y monitoreo de personas, objetos y animales en una infinidad de aplicaciones que van, desde un simple inventario hasta sistemas complejos de casetas de cobro en

carreteras. Los sistemas de identificación por radio frecuencia cuentan generalmente con dos componentes:

- **El "*transponder*":** Pequeña etiqueta electrónica que contiene un minúsculo microprocesador y una antena de radio. Esta etiqueta contiene un identificador único que puede ser asociado a una persona o producto.
- **El lector:** Obtiene el identificador del "*transponder*" (Llamazares, 2007).

- **Memorias de contacto:** Brindan un alto nivel de seguridad, pues son altamente resistentes al desgaste, siendo ideales para ambientes industriales. Es una pastilla electrónica, de unos dieciséis milímetros de diámetro y encapsulada en acero inoxidable. No son recomendables para ambientes con alto grado de generación de corriente estática. Son muy confiables, puesto que su tecnología de avanzada evita la posibilidad de duplicarlas (Kathleen Jui, 2005).
- **Sistemas biométricos:** Su funcionamiento se basa en la lectura o reconocimiento de alguna parte del cuerpo humano; la huella dactilar, geometría de la mano, frecuencia de la voz, la retina o reconocimiento facial; sin necesitar el uso de las tarjetas. De todos, los más conocidos son los lectores de huellas dactilares, geometría de la mano e iris del ojo. La principal ventaja de estos sistemas es la seguridad, y su principal desventaja es su elevado precio de costo, además de la poca posibilidad de ser autónomos (generalmente por su complicada lógica trabajan con un software de análisis y una computadora conectada directa al lector) (Kathleen Jui, 2005).

SCAC utilizará como tecnología el lector de código de barras Voyager. El Voyager 9520 se ha concebido como un lector agresivo con buena profundidad de campo y velocidad de lectura. Este producto mantiene el exclusivo sensor infrarrojo patentado por Metrologic y un sistema de control que permite una activación totalmente automática y su uso como lector de manos libres. El Voyager decodifica todos los códigos de barras de una dimensión (Metrologic, 2006), que son los que se utilizan en la UCI para hacer las tarjetas de identificación o solapines.

1.6 Antecedentes de la investigación

Actualmente existen sistemas de control de acceso que son usados a nivel nacional e internacional. Estos sistemas son implementados para brindar mayor seguridad a las empresas que los utilizan. Algunas de estas aplicaciones son:

✓ Advanced Software - Control de Comedores

Control de Comedores es una aplicación que se puede encontrar en la mayoría de los colegios privados en España, destinada a resolver la problemática asociada al control de los accesos al comedor, permitiendo delimitar turnos e imprimir entradas y permitir que el usuario se informe del menú del día siguiente de forma automática. Incluye los elementos de gestión y clasificación de trabajadores, la gestión de tarjetas de acceso, el registro y control de acceso a comedores, así como la creación de informes y estadísticas del sistema. Se pueden agregar módulos opcionales como: Módulo de Exportación a Nómina; Módulo de Importación/Actualización de datos frecuentes; Módulo de Introducción de Marcajes mediante Huella dactilar; Módulo de Introducción de Marcajes desde una página web. Permite sincronizarse con varias terminales de control de acceso: biométricas, de proximidad, de banda magnética, de chip y muchos más, que se detallan en el sitio. Posee demostraciones flash de algunas aplicaciones y módulos opcionales (Advancedsoft, 2011).

Ventajas competitivas del producto

- Reduce el desperdicio de comida.
- Evita el robo de insumos.
- Evita el acceso de comensales no autorizados a los comedores.
- Incluye el proceso de distribución de comensales por puerta de acceso.
- Se comercializa como una solución compuesta por el sistema distribuidor de personas.
- Conexión con lector de código de barras.
- Se distribuye en forma de paquete que incluye el sistema de distribución y el de control de acceso (cajero).

✓ Gestión de Comedor

Gestión de Comedor es uno de los módulos que integran el Pharaoh Software una solución de capital humano e intelectual diseñado para ayudar, optimizar y mejorar la calidad de trabajo del área de recursos humanos (RRHH) de las empresas. Va encaminado a facilitar el control tanto del consumo de los comensales que la empresa autorice como de su proveedor de servicios de abastecimiento ("*Award Support*", 2011). Este sistema fue desarrollado en Estados Unidos de América y algunas de las empresas que lo usan son: General Motors, Carrefour Unilever y Central de Restaurant.

Ventajas competitivas del producto

- Rangos horarios para definiciones de desayunos, almuerzos, refrigerios y cualquier otra opción que la empresa otorgue de beneficios a sus empleados.
- Posibilidad de valorizar cada menú y emitir información por cantidad y dinero.
- Perfiles de usuario con diferente acceso (Consignatario, RRHH).
- Cantidades máximas diarias por menú definibles por empleados.
- Control de invitados.
- Control de personal habilitado integrado al sistema de control de acceso.
- Validación de la transacción del empleado por el supervisor del comedor en línea.
- Posibilidad de utilizar un equipo para un solo menú (terminal para control de refrigerio).
- Reportes de consumos mensuales o por rango de fechas, por empleado o centros de costos.
- Reportes de gestión por Empleado, Sector, Consumo, Rango de Fechas, Comedor agrupados de diferentes maneras y todos exportables a formato xls, formato del programa Microsoft Excel del paquete ofimático de Windows.

✓ FEED-FOOD

El sistema inteligente FEED-FOOD fue desarrollado para el control y administración automática de comedores empresariales de servicio del tipo comida rápida. Por medio del mismo se puede organizar y controlar el mecanismo de ingreso-egreso y consumo de comedores empresariales (Feed backelectronics, 2011). Algunas de las empresas que utilizan esta aplicación son la Alimentaria Argentina y REPSOL YPF GAS y su desarrollo tuvo lugar en Estados Unidos de América.

Ventajas competitivas del producto

- Disminución considerable en tiempos de espera y permanencia en los servicios.
- Reducción de errores de procesamiento.
- Informe detallado de consumos mensuales.
- Control general del movimiento del personal en el área.
- Informatización de los informes obtenidos.

- Estadísticas del funcionamiento.
- Disminución de costos administrativos.
- Sistematización del control y obtención de índices relativos al consumo por objetivo.
- Banco de datos para estadísticas.

✓ Sistema de Control de Acceso a Comedores

CONTACC tiene como objetivo fundamental gestionar los accesos a los comedores de la UCI, este proceso consiste en: registrar accesos comprobando que dicha persona existe, tiene autorización para pasar por esa puerta y no ha accedido con anterioridad, sincronizar accesos o actualizar cada cierto tiempo la información registrada en la base de datos local con la base de datos central a través de un servicio web; y brindar reportes donde se muestre al usuario la cantidad de personas por tipo (trabajadores y estudiantes) que han tenido acceso hasta el momento (Flores, y otros, 2007). Su desarrollo estuvo a cargo de un equipo de trabajo de la propia universidad.

Ventajas competitivas del producto

- Evita el acceso de comensales no autorizados a los comedores.
- Conexión con lector de código de barras.
- Cantidad máxima de accesos permitidos por la puerta de acceso en un evento determinado.
- Reportes de consumos por categorías.

✓ Sistema para el control de acceso a los laboratorios de producción

Este sistema controla la entrada de personas a los laboratorios productivos. En el mismo se chequea qué personas tienen acceso o no a los laboratorios, verificando que estén en la base de datos correspondiente, mediante el número de la identificación.

De este sistema, en la UCI existen varias implementaciones, cada una de ellas específica para el área productiva donde se encuentra, lo que hace que no exista una base de datos centralizadas con todos los datos referentes a los laboratorios de producción, además son aplicaciones que no cuentan con una amplia documentación (Fernández, y otros, 2010).

Ventajas competitivas del producto

- Evita el acceso a los laboratorios de personas no autorizadas.

- Conexión con el lector de código de barras.

Valoración de los sistemas

Las características y funcionalidades que aportaron los sistemas analizados son: control de puertas, horarios y permisos, manejo de registro, redistribución de comensales e interfaz amigable y sencilla. A pesar de todos los aportes antes mencionados ninguna de estas aplicaciones resuelve los problemas que existen en la universidad con el control de acceso a los comedores, debido a que son sistemas que fueron creados con un fin específico y no para ser usados de forma genérica, sus licencias son muy caras, no cumplen con las políticas definidas por la universidad y no se pueden integrar al Sistema de Planificación y Control del Servicio de Alimentación. CONTACC por su parte presenta problemas de seguridad, no es multiplataforma y tampoco permite la redistribución de personas.

1.7 Proceso, herramientas y lenguajes

Proceso de desarrollo del software

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. No existe ningún proceso de desarrollo que sea de aplicabilidad universal. La realidad es que estos varían porque tienen lugar en diferentes contextos, desarrollan diferentes tipos de sistemas y se ajustan a diferentes tipos de restricciones del negocio (plazo, costos, calidad y fiabilidad). Aunque existen muchos procesos de desarrollo de software diferentes, se pueden mencionar algunas actividades fundamentales que son comunes para todos ellos, tales como (Sommerville, 2005):

1. Especificación del software: Se debe definir la funcionalidad del software y las restricciones en su operación.
2. Diseño e implementación del software: Se debe producir software que cumpla su especificación.
3. Validación del software. Se debe validar el software para asegurar que hace lo que el cliente desea.
4. Evolución del software: El software debe evolucionar para cubrir las necesidades cambiantes del cliente.

Proceso de desarrollo con enfoque ágil basado en el nivel 2 del Modelo de Capacidad y Madurez Integrado (CMMI)

Actualmente la industria del software se caracteriza por un gran dinamismo y variabilidad. Esto, unido a un mercado caracterizado por el rápido desarrollo de aplicaciones y la reducción de la vida de los

productos, obliga a las organizaciones a: incrementar la productividad, disminuir el tiempo de reacción y adaptarse rápidamente a los cambios y a las nuevas necesidades de los clientes (Boehm, 2006). Atendiendo a esta situación y al proceso de desarrollo definido por CENIA, en la presente investigación se definió emplear un proceso de desarrollo con enfoque ágil que permite:

- Valorar al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
- Desarrollar software que funcione por encima de una completa documentación.
- Valorar la colaboración con el cliente por encima de la negociación contractual.
- Responder a los cambios más que seguir estrictamente un plan (Beck, y otros, 2001).

CMMI es un modelo de calidad que establece 5 niveles de madurez: Inicial, Gestionado, Definido, Gestionado cuantitativamente y Optimizado. Cada nivel tiene un conjunto de procesos asociados, que garantizan las mejores prácticas que se han de seguir en la empresa y que aseguran la calidad del producto final, por ejemplo los procesos del nivel 2 de CMMI son:

- Administración de Requerimientos (REQM).
- Planificación de Proyectos (PP).
- Monitoreo y Control del Proyecto (PMC).
- Medición y Análisis (MA).
- Aseguramiento de la Calidad de Productos y Procesos (PPQA).
- Administración de la Configuración (CM).
- Administración de Acuerdos con Proveedores (SAM).

Para asegurar la mejora del proceso definido, el mismo estará basado en las metas y prácticas del nivel 2 de CMMI el cual asegura que el proceso sea gestionado además de ejecutarse, que se planifique, revise y evalúe para comprobar que cumple con los requisitos (SEI, 2011).

El proceso de desarrollo con enfoque ágil basado en el nivel 2 de CMMI tiene definido el siguiente ciclo de vida:

Estudio Preliminar: Se realiza un estudio profundo de la organización cliente que posibilita obtener la información requerida para determinar el alcance del proyecto, así como la estimación del costo, tiempo y esfuerzo.

Modelado de Negocio: Se comprende el negocio de la entidad con el objetivo de que el software a desarrollar cumpla con las expectativas del cliente.

Requisitos: El objetivo fundamental es desarrollar el modelo del sistema, identificando los requisitos funcionales y no funcionales con las descripciones correspondientes en cada caso.
Análisis y Diseño: Se realiza el análisis y el modelado del sistema a partir de los requisitos definidos previamente.
Implementación: A partir de los artefactos obtenidos durante el análisis y diseño se procede a realizar la implementación del software en términos de componentes de implementación.
Pruebas Internas: Se realizan las pruebas internas con el equipo del proyecto en cada una de las iteraciones o versiones finales próximas a ser liberadas, según lo defina el proyecto.
Pruebas de Liberación: Pruebas realizadas por parte de la oficina o institución encargada de la calidad y certificación del proyecto, a todos los entregables obtenidos.
Despliegue: Se realiza la entrega de la aplicación al cliente, así como la configuración y prueba en el ámbito del cliente. Las pruebas realizadas durante esta fase incluye pruebas de aceptación y pruebas pilotos. Se debe realizar además capacitaciones a los trabajadores del sistema.
Soporte: Por un tiempo limitado el proyecto ofrecerá un servicio para resolver conflictos y problemas de usabilidad y rendimiento del software entregado al cliente, suministrándole actualizaciones y parches a errores.

1.7.1 Notación para el Modelado de Procesos de Negocio (BPMN)

Es una notación gráfica estandarizada que permite el modelado de procesos de negocio en un formato de flujo de trabajo. Tiene como principal objetivo proveer una notación estándar que sea fácilmente leíble y entendible por parte de todos los involucrados e interesados del negocio.
Ventajas:

- Permite hacer un mejor uso de la gestión de procesos del negocio y servicios web normalizando el método de notación que sirve como ayuda en la automatización de los procesos.
- Facilita la lectura y comprensión de los procesos. Mediante diagramas del proceso del negocio se pueden mapear los procesos a los lenguajes de ejecución del negocio para automatizarlos usando las notaciones que definen las normas BPMN (Object Management Group, 2011).

1.7.2 Herramientas de modelación

Las herramientas de modelado de sistemas informáticos, son herramientas que se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. Estas herramientas, permiten crear un simulacro del sistema, a bajo costo y riesgo mínimo. A bajo costo porque, es un conjunto de gráficos y textos que representan el sistema, pero no son el sistema físico real (el cual es más

costoso). Además, minimizan los riesgos, porque los cambios que se deban realizar (por errores o cambios en los requerimientos), se pueden realizar más fácil y rápidamente sobre el modelo que sobre el sistema ya implementado (Ramírez, 2009).

Visual Paradigm 8.0

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: modelado del negocio, requisitos, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. También cuenta con abundantes tutoriales de modelado. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar. Presenta características como:

- Entorno de creación de diagramas para BPMN.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales Entorno Integrado de Desarrollo.
- Disponibilidad en múltiples plataforma (Paradigm, 2010).

El proyecto decide utilizar Visual Paradigm por tres razones fundamentales:

- Posee una curva de aprendizaje menos prolongada en el tiempo en comparación con otras herramientas de modelado.
- Es una herramienta multiplataforma.
- Soporta BPMN y UML, para el modelado del negocio.

Evolus Pencil 1.3.4

Es una herramienta libre y de código abierto que se utiliza para crear diagramas y prototipos de interfaz gráfica de usuario que todos puedan usar, es una extensión para Firefox que actúa como una herramienta de dibujo. Fue premiada en el 2010 por Mozilla, permite graficar con total libertad, diagramar, utilizar botones, editar textos y muchas otras opciones (Firefox, 2011).

Características principales:

- Construir plantillas de diagramación y prototipado.
- Conexión entre páginas.
- Multiplataforma: Puede ser instalado tanto en Windows, como Linux, así como agregarse como complemento para el Firefox.

1.7.3 Lenguajes

Lenguaje Unificado de Modelado (**UML) 2.1**

UML es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el ciclo de desarrollo de un producto de software. UML ofrece un estándar para describir modelos, incluyendo aspectos conceptuales tales como procesos de negocio y funciones de sistema, además de aspectos concretos como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reutilizables. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos, además prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que las notaciones y métodos usados para el diseño orientado a objetos han evolucionado, los modeladores solo tienen que aprender una única notación. UML se puede usar para modelar distintos tipos de sistemas, tanto de software como de hardware, además de permitir la organización de mundo real (Booch, y otros, 1999).

Java

El lenguaje de programación Java surge a principios de los años 90 en los laboratorios de Sun Microsystems. Java es un lenguaje de programación multipropósito, reúne todas las características de un ambiente orientado a objetos: es sencillo, cuenta con capacidad de generación de aplicaciones distribuidas, robusta, segura, de arquitectura neutral, portable, multihilo, dinámico y de alto rendimiento.

Las principales características por las que se define utilizar Java como lenguaje de programación son:

- **Orientado a objetos:** Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

- **Distribuido:** Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir tomas y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.
- **Portabilidad:** La portabilidad radica en que se pueden programar las aplicaciones solo una vez y ejecutar en cualquier computadora que tenga instalado el intérprete de Java.
- **Multihilo:** Permite la ejecución de varias tareas a la vez.
- **Robusto:** Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución (Java, 2010).

Plataforma Java

La plataforma Java es el nombre del entorno o plataforma de computación originaria de la compañía "*Sun Microsystems*", capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino una máquina virtual encargada de la ejecución de aplicaciones, y un conjunto de librerías estándar que ofrecen funcionalidades comunes.

Su principal ventaja es que su entorno de desarrollo es independiente de la plataforma sobre la que se trabaje, es decir, sus aplicaciones son funcionales independientemente del sistema operativo sobre el que estén operando. Es una tecnología orientada al desarrollo de software con el cual se puede realizar cualquier tipo de programa (Java 2 Platform, 2010).

Entorno Integrado de Desarrollo (IDE)

Un entorno de desarrollo integrado es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos (HispaNetwork Publicidad y Servicios, 2006).

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: editor de texto, compilador, intérprete, herramientas para la automatización, depurador, sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

1.7.4 Herramientas de desarrollo

NetBeans 7.0.1

El NetBeans es un IDE, disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permite a los desarrolladores la creación de páginas web, aplicaciones de escritorio y aplicaciones móviles utilizando fundamentalmente la plataforma Java. Su instalación y actualización es muy simple y una vez instalado se le pueden adicionar módulos que permiten extender sus funcionalidades. Brinda facilidades para el modelado con UML y posee un diseñador gráfico para juegos y aplicaciones para celulares, empleando la plataforma Java Platform Micro Edition (J2ME) (NetBeans.org, 2011).

Marcos de trabajo para Java:

Spring 3.0.2

En el desarrollo de SCAC se va a utilizar este marco de trabajo pues es un marco de aplicaciones de código abierto que permite la construcción de aplicaciones empresariales en Java. Proporciona componentes basados en patrones de diseño probando que puede ser integrado en todos los niveles de la arquitectura de una aplicación, Spring Framework ayuda a aumentar la productividad en el desarrollo y mejorar la calidad de las aplicaciones y el rendimiento. Su principal novedad está en la inversión de control[1] o inyección de dependencia que permite independizar las diferentes capas de una aplicación (Spring Source Commint, 2010).

Hibernate

El desarrollo de software orientado a objetos y el uso de bases de datos relacionales pueden invertir mucho tiempo en los entornos actuales. Hibernate es una herramienta que realiza el mapeo entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java. El término utilizado es mapeo de objetos a base de datos (ORM[2]) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

---

[1] **Inversión de control (IoC):** Es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones.

[2] **ORM:** es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

Entre las principales ventajas de Hibernate se puede observar el trabajo con objetos, atributos y relaciones, evitando así los conjuntos y tuplas de las bases de datos relacionales. Potente Lenguaje de Consulta de Hibernate (HQL[3]), lo que le facilita al desarrollador realizar consultas sin la necesidad de utilizar sentencias Lenguaje de Consulta Estructurado (SQL). Posee además una comunidad activa, con un amplio número de usuarios y buena documentación (Community, 2012).

1.7.5 Sistema gestor de base de datos (SGBD)

Un SGBD es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Las características de un SGBD son (HispaNetwork Publicidad y Servicios, 2009).

• **Abstracción de la información:** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Sin importar si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

• **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

• **Seguridad:** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

• **Respaldo y recuperación:** Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

[3] **HQL:** es el lenguaje de consultas de Hibernate. HQL es completamente orientado a objetos y comprende nociones como: herencia, polimorfismo y asociación.

• **Control de la concurrencia:** En la mayoría de los entornos, lo habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información o almacenarla. También es frecuente que los accesos se realicen de forma simultánea. Por lo que un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

PostgreSQL 8.4

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia Distribución de Software BerkeLey (BSD) y con su código fuente disponible libremente. PostgreSQL utiliza un modelo cliente-servidor[4] y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Entre las ventajas que brinda este gestor de base de datos se encuentran:

- Puede ser utilizado en los principales sistemas operativos: Linux, Unix, Windows, entre otros.
- Incorpora una estructura de datos "*array*".
- Permite la declaración de funciones propias.
- Soporta el uso de índices, reglas y vistas.
- Aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multihilo.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Puede ser utilizado, modificado y distribuido gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos (The PostgreSQL Global Development Group, 2011).

DataBase4 (for) Objects 6.4 *(DB4O)*

DB4O es un novedoso motor de base de datos orientado a objetos. Sus siglas se corresponden con la expresión DataBase4 (for) Objects, que a su vez es el nombre de la compañía que lo desarrolla: db4objects, Inc. Algunas de las ventajas que brinda el uso de DB4O son (VERSANT CORP, 2010):

- Gran velocidad de desarrollo.
- No hay mapeos entre objetos y tablas.
- El código de acceso a la base es muy sencillo y entendible.

[4] **Cliente-servidor:** Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta.

- Fácil copia de seguridad (la base completa está en un solo archivo).
- No necesita administración.
- Tiene un recolector de basura que borra los objetos que no son referenciados.
- Las búsquedas se hacen directamente usando objetos.
- Multiplataforma.
- Nativo en Java y .Net.
- Búsquedas usando objetos y sencillas consulta por ejemplos (QBE[5]).

1.7.6 Herramientas para la administración de bases de datos

PgAdmin III 1.12

Es una popular herramienta de código abierto para la administración y desarrollo en la plataforma PostgreSQL. Puede ser usado en Linux, FreeBSD, Solaris, Mac OSX y Windows. PgAdmin III está diseñado para responder a las necesidades de los usuarios; desde la escritura de simples consultas SQL hasta para el desarrollo de bases de datos complejas. La interfaz gráfica soporta todas las características y hace fácil la administración. La conexión con el servidor puede hacerse a través del Protocolo de Control de Transmisión/Protocolo de Internet (TCP / IP) o *Unix Domain Sockets* (en plataformas Unix), y puede utilizar *Secure Sockets Layer* (SSL) para la seguridad (PgAdmin PostgresSQL Tools, 2010).

1.8 Conclusiones parciales

En el presente capítulo se expone un marco informativo y conceptual como resultado de la investigación realizada para dar solución al problema propuesto. Se realizó un estudio del estado del arte acerca de otras aplicaciones con propósitos similares a la propuesta, lo que corroboró la necesidad de desarrollar una nueva aplicación, se realizó además un profundo estudio de las tecnologías, herramientas y lenguajes de programación idóneos para el desarrollo del subsistema, teniendo en cuenta las que ya están definidas por el Centro de Informatización Universitaria.

[5] **QBE:** Es un mecanismo que utiliza DB4O para acceder a los datos almacenados.

Capítulo II Concepción de la propuesta de solución

2.1 Introducción

En este capítulo se identifican las necesidades del cliente a través del estudio del negocio, se describen las características del sistema y el flujo actual del proceso de negocio relacionado con el objeto de estudio que lo soporta. Se especifica además el proceso que será objeto de automatización y los requisitos principales del software mediante los requerimientos funcionales y no funcionales, definiéndose la propuesta de sistema.

2.2 Proceso del negocio

Descripción del proceso de control de acceso

Tabla 2.1 Descripción del proceso control de acceso.

Nombre:	Control de acceso a comedores.
Objetivos:	Registrar los accesos de las personas, teniendo en cuenta la cantidad de accesos previstos sin que sea afectada la asignación del evento.
Evento(s) que lo generan:	Inicio del evento.
Precondiciones:	Ha iniciado la hora del evento.
Poscondiciones:	
Reglas de Negocio:	• La tarjeta de identificación tiene asignado un código de barras y un número de identificación que la identifica. • A cada persona le corresponde una tarjeta de identificación. • Solo el cajero tiene acceso a la información que se genera. • Cada puerta del comedor debe estar custodiada por un cajero. • El cajero permanecerá en su puesto de trabajo chequeando el acceso de los usuarios durante el evento e informará cualquier suceso que afecte el proceso. • Ningún usuario sin tarjeta de identificación, puede entrar al comedor si no ha sido aprobado por un directivo. • El usuario solo tendrá acceso a una puerta, para un mismo evento. • Para cada evento se define la cantidad de accesos válidos por usuario. • Para cada puerta en un evento se define un plan de raciones permitidas.
Responsables:	Cajero.

Clientes internos:	Ninguno.
Clientes externos:	Usuario.
Entradas:	Listado de usuarios.
Salidas:	Listado de usuarios actualizado.
Actividades:	• El usuario solicita el acceso. • El cajero verificar los datos del usuario. • El cajero informa del problema. • El cajero registra el acceso. • El usuario accede al comedor.

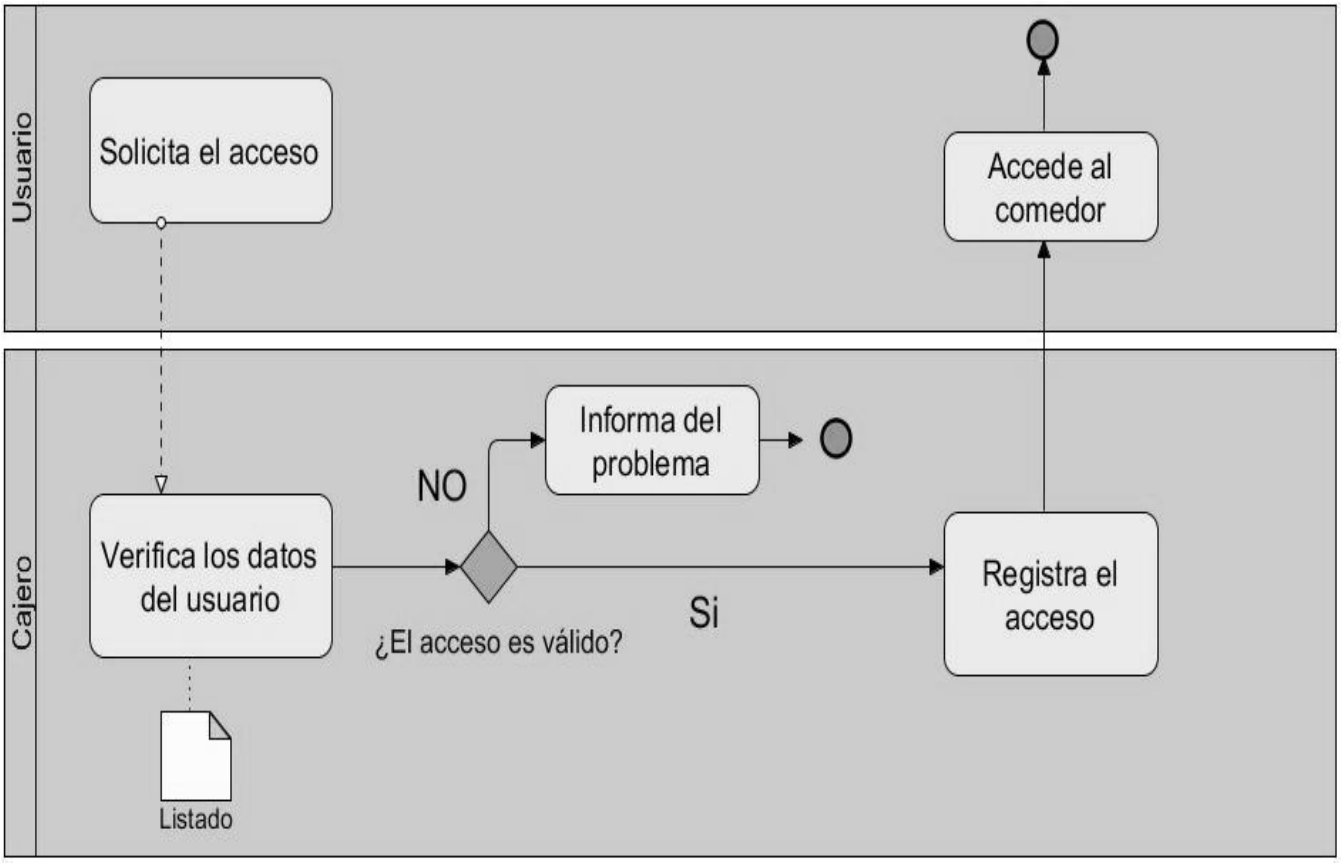

Figura 1 Diagrama del proceso de control de acceso a los comedores de la UCI.

Descripción del flujo básico

Tabla 2.2 Descripción del flujo básico del proceso control de acceso.

Nombre	Descripción
Control de acceso	El usuario solicita el acceso al comedor por una puerta del mismo, luego el cajero verifica en el listado los datos del usuario y si no ha accedido anteriormente al evento que se encuentra desarrollándose, registra el acceso de la persona y el usuario accede al comedor para recibir el servicio.
Responsable	Cajero.
Entrada	Listado.
Salida	Listado.

Conceptos asociados al modelo de proceso del negocio:

Usuario: Toda persona vinculada a la UCI: estudiantes, profesores, trabajadores de las áreas de la universidad, tercerizados, entre otros.

Tarjeta de identificación: Identificación de los usuarios para el acceso a los eventos que se desarrollan en el comedor.

Cajero: Persona capacitada para el manejo del subsistema en cualquiera de los eventos.

Evento: Actividad que se desarrolla en un comedor (desayuno, almuerzo, comida y otros).

2.3 Descripción de la propuesta de solución

SCAC es una aplicación de escritorio, que está integrada al Sistema de Planificación y Control del Servicio de Alimentación, su objetivo principal es controlar la entrada de los comensales por cada una de las puertas de los comedores de la universidad. La aplicación posibilita además: asignar a una puerta el punto de control que le corresponde, gestionar los tiempos de sincronización entre la base de datos local y el servidor central, descargar todos los eventos que se van a desarrollar en los comedores, gestionar el plan definido para la puerta en un evento determinado, redistribuir las personas de una puerta a otra en caso de que fuese necesario y ofrecer un reporte con la cantidad de accesos por categorías (estudiantes, trabajadores) y exportarlo en formato Excel y PDF. El subsistema debe poseer dos niveles de acceso: el administrador de la aplicación que sería el usuario encargado de gestionar toda la información referente a la configuración del subsistema; y el cajero, que es el usuario encargado de validar la entrada de las personas por cada uno de los puntos de acceso al comedor.

2.4 Funcionalidades del sistema

Requisitos Funcionales: Son capacidades o condiciones que el sistema debe cumplir, sin tomar en consideración ningún tipo de restricción física, de manera que se mantienen invariables sin importar con qué propiedades o cualidades se relacionen (Booch, y otros, 1999).

Listado de las funcionalidades:

1. Autenticar usuario.
2. Activar punto de control.
3. Crear usuario.
4. Modificar usuario.
5. Eliminar usuario.
6. Modificar contraseña.
7. Descargar eventos.
8. Descargar puntos de control.
9. Registrar cantidad de veces que va a ser cambiado el plan.
10. Sincronizar acceso.
11. Registrar plan.
12. Modificar plan.
13. Mostrar datos de la persona.
14. Registrar acceso.
15. Cantidad de personas que acceden por evento.
16. Mostrar reportes persona por categoría.
17. Redistribuir comensales.
18. Mostrar mensajes de aviso dada una cantidad de acceso.
19. Consultar cantidad de acceso por persona.

Cada requerimiento funcional es descrito a través de la especificación de requisitos, a continuación se muestran las descripciones de las funcionalidades más importantes del subsistema, las restantes

podrán consultarse en el **Anexo 1**.

Tabla 2.3 Especificación de requisitos Activar punto de control.

<table>
<tr><th>Nº</th><th>Nombre</th><th>Descripción</th><th>Complejidad</th><th>Prioridad para cliente</th></tr>
<tr><td>RFCA 2</td><td>Activar punto de control.</td><td>Se accede en el menú “Configuración” a la opción “Activar punto de control” y se muestra una interfaz con los datos necesarios para activar un punto de control y poder realizar el acceso de los comensales por el mismo en un evento determinado.</td><td>Alta.</td><td>Alta.</td></tr>
<tr><td></td><td colspan="4">Prototipo</td></tr>
<tr><td></td><td colspan="4"> 
</td></tr>
<tr><td></td><td>Campos</td><td>Tipos de Datos</td><td colspan="2">Regla o Restricciones</td></tr>
<tr><td></td><td>• Punto de control activo.</td><td>• Texto.</td><td colspan="2">• No procede.</td></tr>
<tr><td></td><td>• Activar.</td><td>• Texto.</td><td colspan="2">• Se debe seleccionar un punto de control.</td></tr>
<tr><td></td><td>Observaciones</td><td colspan="3">1. Debe existir conexión con la base de datos para activar un punto de control.
2. Solo se muestra para activar en la selección los puntos de control que no hayan sido activados.</td></tr>
</table>

Tabla 2.4 Especificación de requisitos Registrar acceso.

<table>
<tr><th>Nº</th><th>Nombre</th><th>Descripción</th><th>Complejidad</th><th>Prioridad para cliente</th></tr>
<tr><td rowspan="5">RFCA 13</td><td>Registrar acceso.</td><td>Mediante un lector de código de barra se obtiene la información de los comensales para registrar en la base de datos local el acceso del mismo por un punto de control, en un evento y una fecha determinada.</td><td>Alta.</td><td>Alta.</td></tr>
<tr><td colspan="4">Prototipo</td></tr>
<tr><td colspan="4">  </td></tr>
<tr><td>Campos</td><td>Tipos de Datos</td><td colspan="2">Reglas o Restricciones</td></tr>
<tr><td>• Lector.</td><td>• Campo de texto.</td><td colspan="2">• El código de barra tiene que ser válido.</td></tr>
<tr><td>Observaciones</td><td colspan="4">1. Si el valor del campo <Lector> es nulo o es un código erróneo el sistema debe mostrar un mensaje de error.
2. El número de accesos de un comensal a un evento debe ser menor o igual a la cantidad de accesos permitidos, de lo contrario el sistema debe mostrar un mensaje denegándole el acceso.
3. No se pueden registrar accesos si el plan asignado al punto de control en el evento se agotó.
4. Si la persona no reservó para ese evento en la fecha que corresponde, el sistema debe mostrar un mensaje comunicándole que no puede acceder al comedor.
5. Si el comensal no está asignado al punto de control en el evento que se desarrolla no puede realizar el acceso y el sistema debe mostrar por cuál de los puntos de control tiene acceso al servicio.</td></tr>
</table>

2.5 Características del sistema

Requisitos no funcionales: Son propiedades o cualidades que el producto debe tener. Son fundamentales en el éxito del producto y normalmente están vinculados a requisitos funcionales (Booch, y otros, 1999).

Usabilidad

- ✓ Facilidad de uso: El subsistema debe presentar una interfaz amigable al usuario, propiciando una fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Además esta interfaz debe ser de manejo cómodo donde la curva de aprendizaje para los usuarios sea lo menos inclinada posible y que posibilite en estos una rápida adaptación.
- ✓ Especificación de la terminología utilizada: El subsistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vista a una mayor comprensión por su parte sobre la herramienta de trabajo.
- ✓ Menús: El subsistema debe presentar una serie de menús, que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.

Soporte

- ✓ Grupo de soporte y asesoría: el sistema contará con un grupo de soporte y asesoría al cliente.

Interfaz

- ✓ Interfaz externa: La interfaz deberá ser sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo de su empleo.
- ✓ Interfaz interna: La interfaz interna estará determinada por los desarrolladores, construyendo así una vista escalable de las clases o agrupaciones de clases que permitirán un mejor encapsulamiento de las funcionalidades y una mayor abstracción modular del sistema.

Portabilidad

- ✓ El subsistema deberá funcionar sobre cualquier plataforma que soporte la instalación de la Máquina Virtual de Java 7.0.

Seguridad

- ✓ Garantizará que la información sea modificada y vista únicamente por quien tenga permiso para esto.
- ✓ El acceso al subsistema debe estar restringido por el uso de claves asignadas a cada uno de los usuarios. Solo podrán ingresar al subsistema las personas que estén registradas.
- ✓ El subsistema no permitirá activar un mismo punto de control para 2 puertas diferentes.
- ✓ La aplicación solo debe utilizar la base de datos local creada para un evento en una fecha específica para que no pueda ser copiada y utilizada posteriormente.

Restricciones de diseño

- ✓ El subsistema se desarrollará utilizando Java como lenguaje de programación.
- ✓ Los marcos de trabajo que se utilizarán son: Spring 3.0.2 e Hibernate.
- ✓ El IDE de desarrollo y mantenimiento del subsistema es NetBeans 7.0.1.
- ✓ El sistema gestor de bases de datos será PostgreSQL 8.4.
- ✓ El sistema operativo a utilizar en el entorno de desarrollo deberá ser: Linux.

Hardware

- ✓ Para el desarrollo: Una computadora con procesador Pentium 4 a 2.0 GHz o superior, 2 GB de RAM recomendada o superior.
- ✓ Para explotación del cliente: Una computadora con procesador Pentium 4 a 1.6 GHz o superior, 512 MB de RAM recomendada o superior.

Software

- ✓ Debe estar instalada la Máquina Virtual de Java 7.0 o superior.

2.6 Entidades que integran la base de datos

Las tablas siguientes representan las entidades que integran la base de datos y contienen la información más importante de las mismas en cuanto a atributo, tipo de datos de los atributos y otros campos necesarios que definen una entidad.

Tabla 2.5 Descripción de la entidad Acceso.

Nombre de la entidad	Acceso.						
Descripción de la entidad	Esta entidad es de gran importancia ya que es la que almacena los accesos a los diferentes puntos de control.						
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_punto	El identificador del punto al que se va a tener acceso.	Entero.	No	1	Vacío Caracteres Letras	No	Si
id_evento	El identificador del evento al que se va a tener el acceso.	Entero.	No	2	Vacío Caracteres Letras	No	Si
racion	Es el número de veces que ha pasado	Entero.	No	3	Vacío Caracteres	No	Si

	una persona.				Letras		
id_persona	El identificador de la persona que va a tener dicho acceso.	Entero.	No	43	Vacío Caracteres Letras	No	Si
fecha	La fecha en la que va a tener acceso.	Texto.	No	04/02/12	-	No	Si

Tabla 2.6 Descripción de la entidad Persona autorizada.

Nombre de la entidad	Persona autorizada.						
Descripción de la entidad	Esta entidad almacena los datos necesarios de todas las personas de la UCI.						
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_persona_autorizada	Identificador de la persona.	Texto.	No	127000	Vacía	No	Si
asignada	Permite conocer si la persona fue asignada a un punto de control.	*"Boolean"*.	No	Verdadero/ Falso	Vacía	No	No
activa	Se activa si desea darle autorización a la persona.	*"Boolean"*.	No	Verdadero/ Falso	-	No	Si
codigo_barra	Código que se utiliza para conceder el acceso al comedor a una persona.	Texto.	No	E10017	Error	No	Si
foto	Identificador de la foto de una persona.	Texto.	No	foto_02	-	No	Si

Tabla 2.7 Descripción de la entidad Punto de control.

Nombre de la entidad	Punto de control.						
Descripción de la entidad	Es un punto de acceso a los comedores de la universidad.						
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**

id_punto_control	Identificador de un punto de control.	Texto.	No	0111	Vacía	No	Si
id_estructura	Identificador de la estructura a la que pertenece el punto de control.	Texto.	No	01	Vacía.	No	Si
capacidad	Cantidad máxima de personas que pueden ser asignadas al punto de control.	Entero.	No	200	Vacía.	No	Si
ip	Dirección ip de la máquina que activó el punto de control.	Texto.	Si	10.53.8.123	10.23.0.0.0.125	No	Si
nombre_punto_control	Se escribe el nombre del punto de control.	Texto.	No	Puerta 322.	Vacío	No	Si
Descripcion	Se describe el punto de control.	Texto.	Si	Esta es la puerta número 2 que pertenece al comedor 2 del complejo 3.	-	-	Si
Activo	Se activa si cuando es asignado a un punto de acceso al comedor.	*"Boolean".*	No.	Verdadero o Falso.	-	-	Si

Tabla 2.8 Descripción de la entidad Reservación.

Nombre de la entidad	Reservación.						
Descripción de la entidad	Es la entidad donde se recogen todos los datos de una reservación.						
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_reservacion	Se describe el identificador para una reservación.	Entero.	No	1	* /@#~ Vacío	No	Si
fecha	Se registra la fecha en	Fecha.	No	2012-05-	Vacía	No	No

	la que es atendida la reservación.			12			
solapin	Número de solapín de la persona autorizada.	Texto.	No	H12742	* /@#~ Vacío	No	Si
activo	Permite conocer si la reservación ha sido cancelada o no.	*"Boolean".*	No	Verdadero/ Falso	Vacío	No	Si

Tabla 2.9 Descripción de la entidad Evento.

Nombre de la entidad	Evento.
Descripción de la entidad	Esta entidad describe las actividades que ocurren en los comedores de la universidad diariamente, los cuales pueden ser desayuno, almuerzo y comida.

Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
Id_evento	Identifica al evento.	Entero.	No	1	ass	No	Si
nombre_evento	Se describe el nombre del evento.	Texto.	No	Comida.	Vacío / caracteres	-	Si
numero_acceso	Indica las raciones que son permitidas para una persona en un evento.	Entero.	No	1	Caracteres	-	Si
categoria	Regular, no regular doble, no regular venta.	Texto.	No	Regular.	-	No	No
hora_inicio	Indica la hora de comienza del evento determinado.	Tiempo.	No	08:20:00	-		
hora_fin	Hora en la que se termina el evento (Desayuno, Almuerzo y comida o cualquier otro evento que se defina).	Tiempo.	No	10:20:00	-	-	Si
activo	Se activa o no si se desea que el evento este en función.	*"Boolean"*	No	Verdadero/ Falso	-	-	Si

2.7 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, este debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si todo el código fuente fuese escrito por un único programador.

En el caso de las declaraciones, para declarar una clase se escribió el nombre de la clase con la primera letra en mayúscula y el resto del nombre en minúscula, si son nombres compuesto la primera letra de la segunda palabra empieza con mayúscula también. Los métodos de una clase tienen una forma encamellada (*CamelCase*). Las variables se declararon en la mayoría de los casos usando abreviaturas y las constantes se declararon en mayúscula.

Los comentarios se definen comenzando con los caracteres /* y terminando con */ Ejemplo: /* esto es un comentario */, siempre que se quiera comentar grandes instrucciones de códigos. Si solo se desea comentar una sola instrucción se utiliza //. Cuando se implementa un método se pone un comentario en el encabezado que permita visualizar al mismo de forma más rápida.

2.8 Patrones y arquitectura

Patrones de diseño

Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basadas en la experiencia que brindan una estructura conocida para todos los programadores. Permiten tener una estructura de código común, ahorran tiempo en la construcción de un software, son fáciles de comprender, mantener y extender, además dan una mejor imagen de profesionalidad y calidad. Existen varios tipos de patrones que se agrupan en dos grandes grupos: los Patrones de Asignación de Responsabilidades (GRASP) y los patrones Banda de los Cuatro (GoF).

Para realizar el diseño del Sistema de control de acceso para los comedores de la Universidad de las Ciencias Informáticas se aplicaron los patrones de diseño GRASP.

Bajo acoplamiento: Uno de los principios para proteger al software frente al cambio es mantener bajo el acoplamiento entre clases. El acoplamiento de una clase es el conjunto de dependencias que tiene con otras clases, cuanto menor sea el acoplamiento entre clases, menor influencia tendrá los cambios. En la solución que se ofrece, cada clase se relaciona solo con quien lo necesita para realizar sus procedimientos (o métodos).

Alta cohesión: Al asignar responsabilidades en el diseño, se buscan soluciones que asignen los métodos a las clases de forma coherente, completa y relacionada. De esta forma, se obtienen clases cohesionadas. Una clase cohesionada facilita el cambio, al realizar un cambio en una clase muy cohesionada, todos los métodos que pueden verse afectados, toda la información que se necesita controlar, estará a la vista, en el mismo fichero. Este patrón se relaciona con el de bajo acoplamiento, porque un diseño cohesionado tendrá un bajo acoplamiento entre clases. Ejemplos de este patrón se pueden encontrar en todas las clases de las capas de negocio y acceso a datos, donde cada clase realiza los métodos que le competen, según su concepción y finalidad.

Experto: Se tiene en consideración a qué clase debe pertenecer un método, este principio sugiere que se asigne a la clase que más sepa del método (es decir al experto). Esto es una consecuencia del principio de alta cohesión, ya que si se asignan los métodos a las clases que tienen la información necesaria para ejecutarlos, se están creando clases altamente cohesionadas. Al igual que el patrón anterior, un ejemplo de la utilización de este patrón se puede encontrar en la clase **PuntoControlDao** la cual se encarga de obtener específicamente los datos referentes a las personas que se manejan en el subsistema.

Creador: Permite decidir cuáles serán las clases creadoras de otras clases. Este patrón se tiene muy en cuenta a la hora de estructurar las clases según la arquitectura, donde cada clase de una capa superior crea su similar en la inferior (en conjunto con el marco de trabajo, ya que Spring es quien maneja los objetos del negocio), así mismo se tiene de ejemplo que la clase **AccesoNeg** es quien concibe (y en conjunto con el marco de trabajo) crea la clase **AccesoDao**, la cual utilizará para sus operaciones (Oliveras, 2010).

Dentro de los patrones GoF conocidos se utilizó el patrón de creación:

Instancia única: Garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella. Esto se logra haciendo que sea la propia clase la responsable de controlar la existencia de una única instancia. Este patrón se utiliza en la clase **FachadaDao** la cual se utiliza en la creación del fichero DB4O.

2.9 Arquitectura

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de software o Arquitectura lógica. Se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiar el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo

una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado (Guglielmetti, 2004).

El sistema desarrollado está basado en una arquitectura por capas, la cual es una especificación de la arquitectura cliente-servidor. La arquitectura por capas brinda un estilo de programación donde el objetivo primordial es la separación de la lógica de negocio de la lógica de diseño. En el caso específico del sistema desarrollado la arquitectura está compuesta por tres capas (ver Figura 2), capa de presentación, capa de negocio y capa de acceso a datos. El uso de Spring aporta el patrón de inyección de dependencia con el cual se delega la responsabilidad de instanciar los objetos en archivos de configuración Lenguaje de Marcas Extensibles (XML). Esto resulta muy útil puesto que evita cualquier dependencia que pueda existir entre los objetos, garantizando la comunicación entre las capas horizontales y también entre los módulos y permitiendo el desacoplamiento entre las capas. A continuación se detallan cada una de las capas y los componentes que conforman la arquitectura:

Capa de presentación

- **Formularios de interfaz de usuario**

Todos los módulos tienen un diseño y comportamiento estándar con vistas a facilitar el trabajo con ellos y la estandarización del subsistema completo. Su uso se basa en la explotación de los formularios de interfaz de usuario que provee el IDE facilitando el desarrollo del sistema.

- **Gestión de formularios de interfaz de usuario**

Se basa en acciones contempladas en la gestión de formularios de interfaz de usuario donde cada acción se corresponde con funcionalidades de captura o visualización de los datos que tiene asociado un formulario cuya forma depende de la funcionalidad específica para la cual fue concebida dicha acción.

Capa lógica del negocio

El diseño de esta capa depende directamente del negocio específico al que se refiera cada módulo. El negocio recibe datos y/o información capturada en las interfaces de usuario, gestiona o procesa la misma, de ser necesario solicitándola a la capa de acceso a datos y finalmente enviársela a la presentación nuevamente para que esta la muestre al usuario en el punto donde se inició la petición.

- **Interfaces de negocio**

El pilar fundamental de la arquitectura radica en el uso de las interfaces como recurso que se utiliza para brindar funcionalidades que representan un nivel de abstracción. Este nivel es precisamente el que asegura el patrón bajo acoplamiento conjuntamente con otros elementos.

- **Lógica de negocio**

Componentes encargados de implementar las funcionalidades descritas en las interfaces de negocio, definiendo las reglas de negocio específicas de cada acción.

Capa acceso a datos

Definir la estrategia de persistencia de una aplicación es una de las decisiones de Arquitectura más importantes. En una aplicación estándar más del 50 % del código generado está relacionado con lógica de persistencia.

Por tanto, el Acceso a Datos es la capa más crítica y sensible a cambios de la Arquitectura, pues controla todo lo concerniente a la información que se encuentra en la fuente de almacenamiento (Origen de datos). Al ser la capa inferior no conoce los niveles superiores, únicamente se limita al manejo de la información, ya sea para persistirla o proporcionarla para su procesamiento y propagación por la aplicación. Todo este manejo es responsabilidad de los Objetos de Acceso a Datos (DAOs por sus siglas en inglés) (Carreño, 2011).

- **Interfaces de acceso a datos**

Representan las funcionalidades que brinda este nivel, es decir, las referidas a los DAOs. Su uso y funcionamiento es el mismo que las de la capa lógica de negocio.

- **Lógica de acceso a datos**

Todas las funcionalidades del acceso a datos radican en los DAOs, es decir este ensamblado implementa la totalidad de las operaciones de persistencia y obtención de datos explotando los recursos que brinda Hibernate que cumple perfectamente con el objetivo de este nivel, dígase trabajo con procedimientos almacenados y métodos de persistencia o consultas.

Origen de datos

Corresponde a la fuente de datos del subsistema y es común a todos los módulos del subsistema en general.

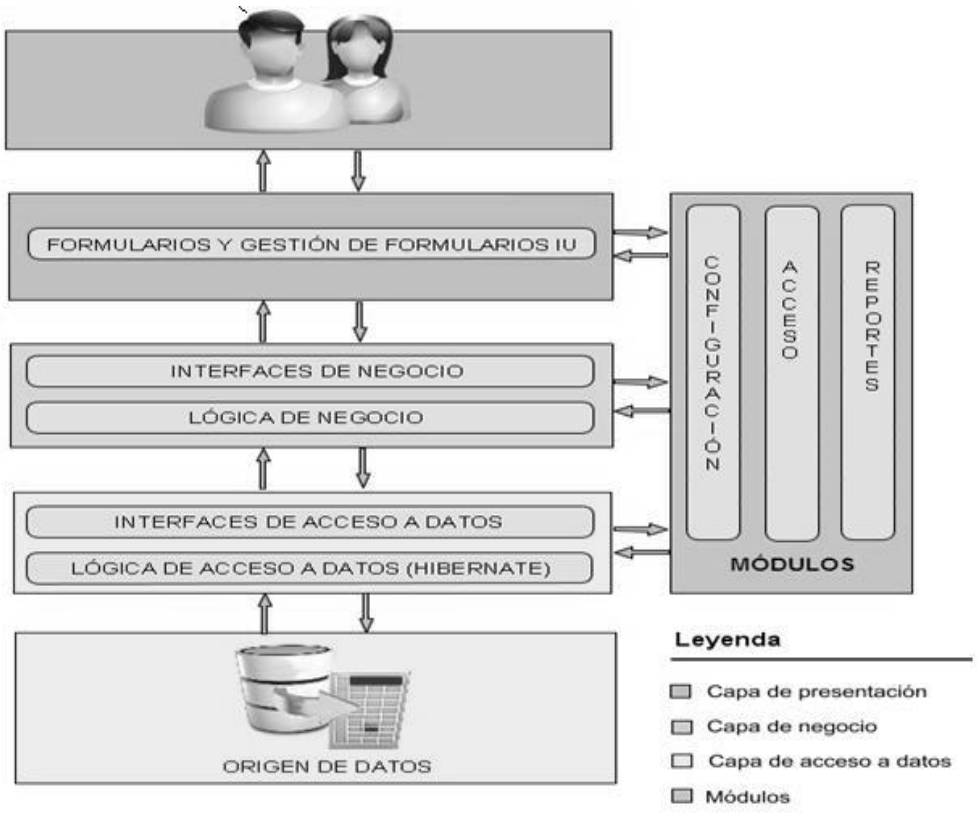

Figura 2 Arquitectura por capas.

2.10 Diseño del esquema de base de datos del subsistema

Las bases de datos necesitan una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán. La puesta en práctica de la base de datos es el paso final en el desarrollo de aplicaciones de soporte del negocio y se conforman con los requisitos del proceso del negocio, que es la primera abstracción de la vista de la base de datos.

La base de datos del Sistema de Planificación y Control del Servicio de Alimentación cuenta con 36 tablas, sin embargo, el Subsistema de control de acceso a los comedores de la UCI utiliza solamente 18 de estas tablas; 10 de datos, 2 nomencladores y 6 de relaciones. La nomenclatura de dichas entidades es de la siguiente forma: las tablas de datos tbd_<nombre>, los nomencladores tbn_<nombre> y las que almacenan relaciones entre entidades tbr_<nombre1>_<nombre2>. A continuación se muestra el diseño del esquema de base de datos propuesto a través del modelo lógico y físico.

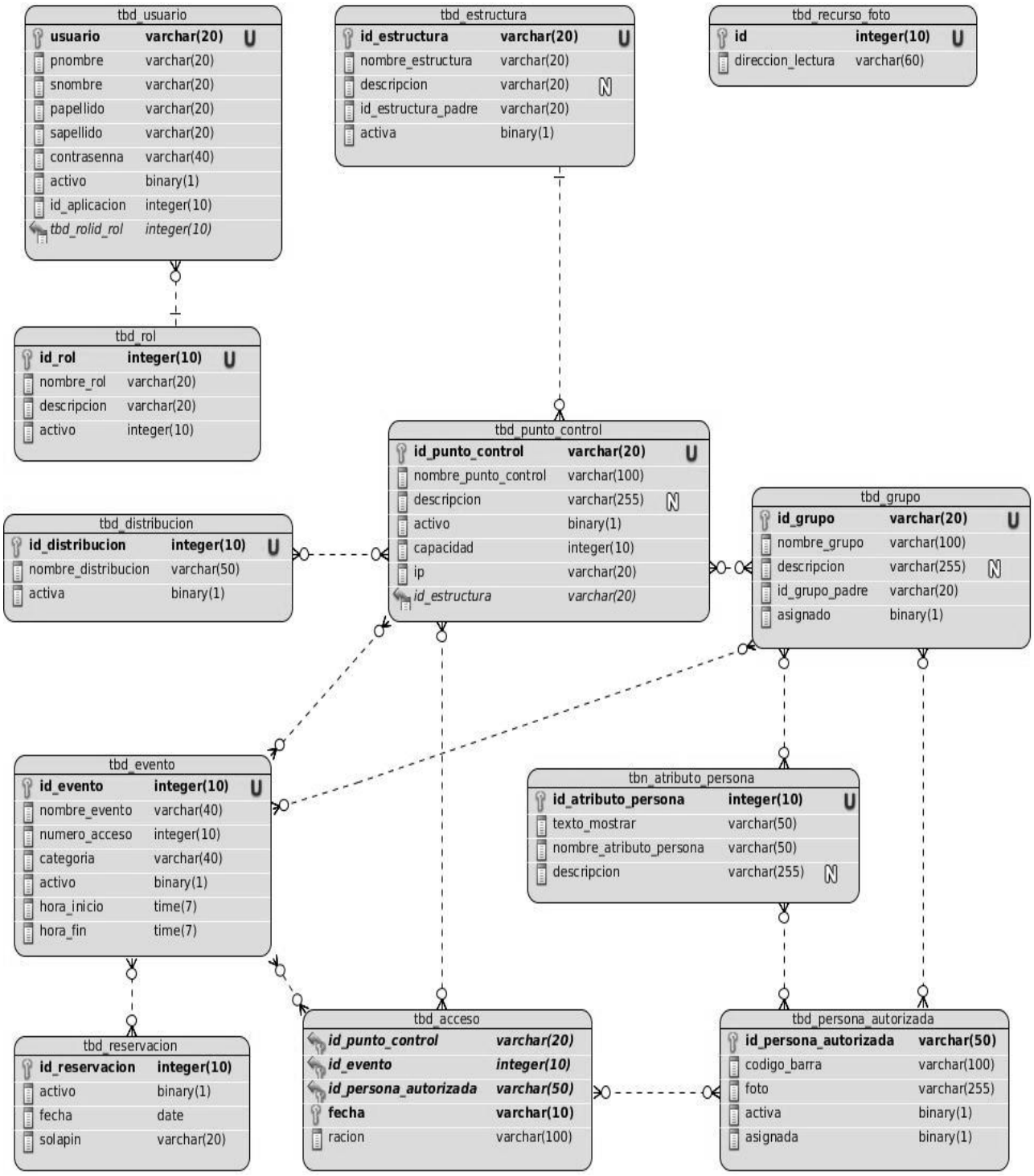

Figura 3 Modelo lógico de la base de datos.

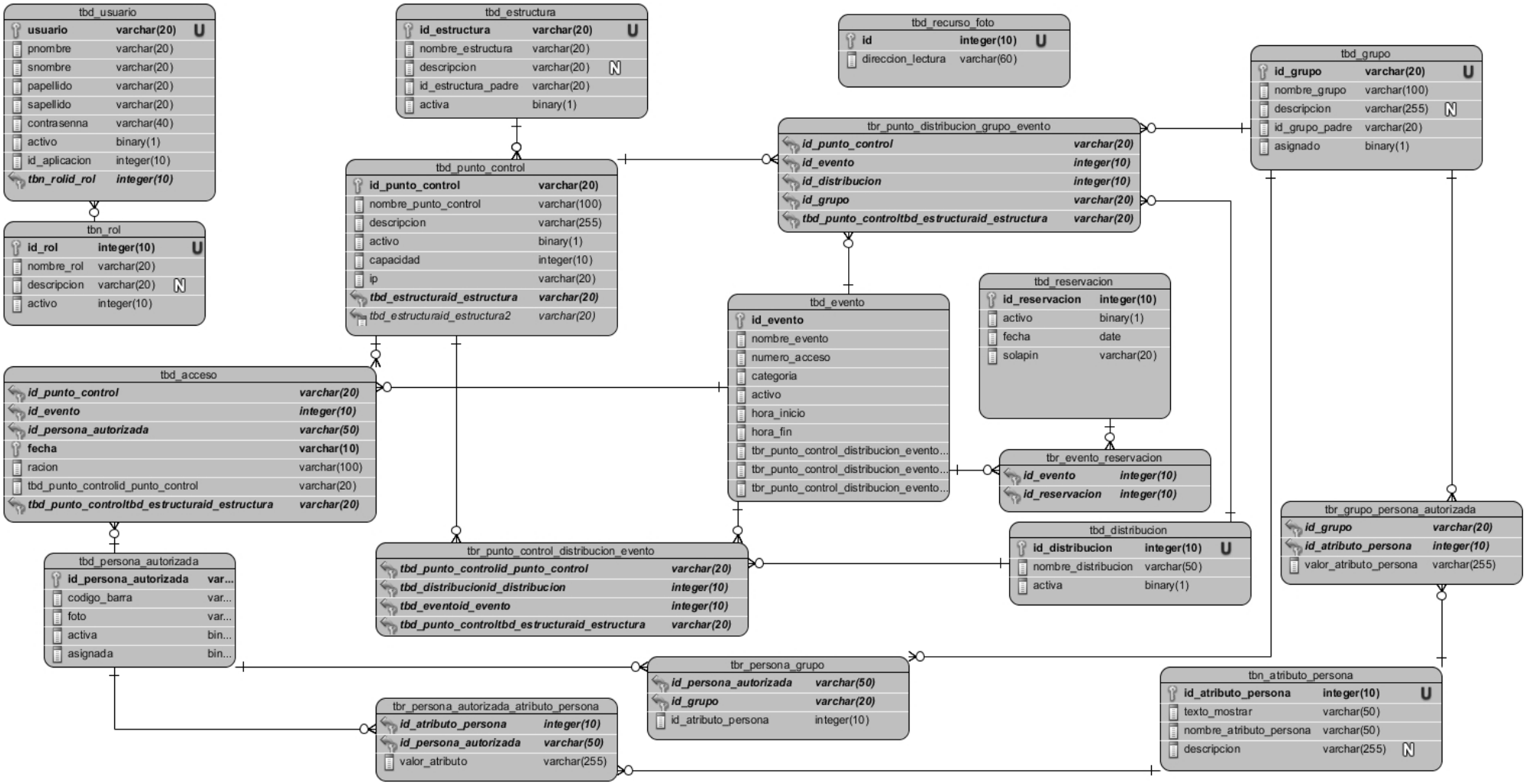

Figura 4 Modelo físico de la base de datos.

2.11 Diagrama de despliegue

La relación física de los distintos nodos que componen el subsistema y la distribución de los componentes sobre dichos nodos unidos por conexiones de comunicación estará representada por el diagrama de despliegue siguiente:

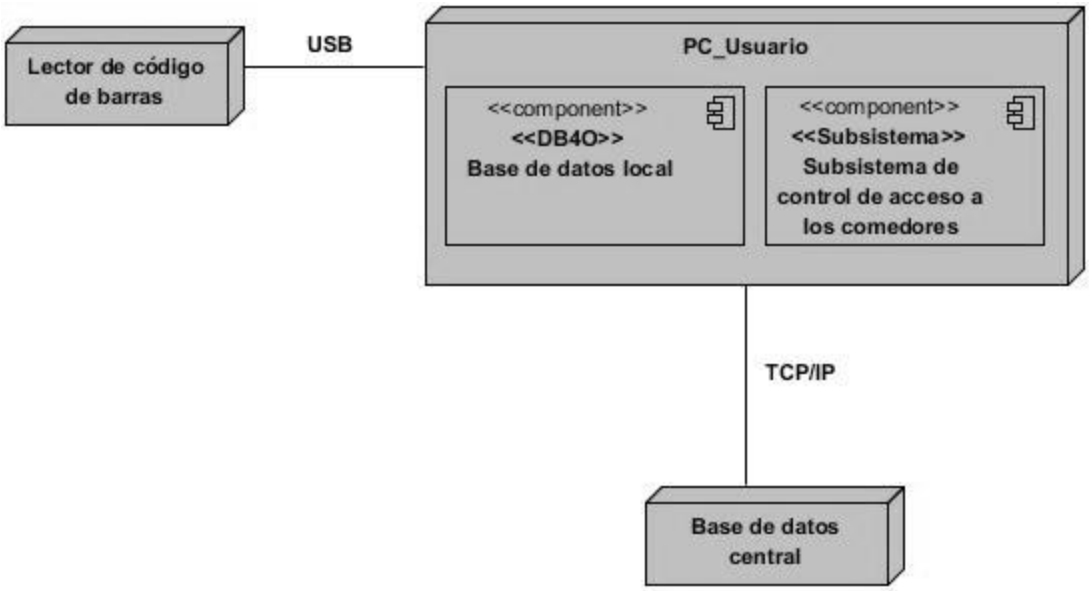

Figura 5 Diagrama de despliegue.

PC_Usuario: Nodo físico donde estará funcionado la aplicación para controlar el acceso y donde se encontrará el fichero DB4O.

Base de datos centralizada: Nodo físico que almacena toda la información del Sistema de Planificación y Control del Servicio de Alimentación.

Lector de código de barra: Dispositivo de identificación automática que se utiliza para la captura del código de barras de los solapines de los comensales.

Bus de Serie Universal (USB): Es una entrada o acceso que permite conectar el lector de código de barras con la PC_Usuario.

TCP/IP: Es un conjunto de protocolos que permite enlazar cualquier tipo de computadoras, sin importar el sistema operativo que usen o el fabricante. Este permitirá el intercambio de datos entre la base de datos local y la base de datos central.

2.12 Conclusiones parciales

Como resultado del capítulo se describe el proceso de negocio para ganar en claridad y tener una panorámica de las funcionalidades que la aplicación debe cumplir, así como detallar los requisitos funcionales y no funcionales del subsistema en la especificación de requisitos. También se describe la arquitectura y los patrones de diseño utilizados, se muestran los modelos físicos y lógicos de datos propuestos para satisfacer las necesidades del subsistema, además se propone la distribución física de SCAC.

Capítulo III Validación de la propuesta de solución

3.1 Introducción

El desarrollo de un software es algo complejo, son innumerables las posibilidades de errores, por lo que debe ir acompañado de alguna actividad como las pruebas, que garanticen la calidad del software. En este capítulo se realiza la validación a la solución propuesta, de manera que se puede comprobar la eficiencia de las clases u operaciones utilizadas para dar respuesta a los distintos requisitos planteados por el cliente.

3.2 Matriz de trazabilidad

La matriz de trazabilidad representa una relación entre los requisitos, tareas de ingeniería y casos de prueba de aceptación. La trazabilidad de requisitos se define como la habilidad para describir y seguir la vida de un requisito en ambos sentidos, hacia sus orígenes o hacia su implementación, a través de todas las especificaciones generadas durante el proceso de desarrollo de software.

Tabla 3.10 Matriz de trazabilidad.

Identificador	Especificación de requisito	Tareas	Caso de Prueba	Iteración
1 2 3 4	• Crear usuario. • Modificar usuario. • Eliminar usuario. • Modificar contraseña.	Crear un nuevo usuario. Modificar un usuario. Eliminar usuario. Consultar estado del usuario. Modificar la contraseña.	0122__CENIA_UD_ CA_CP_Gestionar usuario.	1
5	• Autenticar usuario.	Autenticar un usuario.	0122__CENIA_UD_ CA_CP_Autenticar usuario.	1
6	• Registrar la cantidad de veces que va ser cambiado un plan.	Registrar la cantidad de veces que va ser cambiado un plan.	0122__CENIA_UD_ CA_CP_ Registrar la cantidad de veces que va ser cambiado un plan.	1
7	• Sincronizar acceso.	Sincronizar acceso.	0122__CENIA_UD_	1

			CA_CP_Sincronizar acceso.	
8 9	• Registrar plan. • Modificar plan.	Registrar un plan. Modificar un plan.	0122__CENIA_UD_ CA_CP_ Registrar plan. 0122__CENIA_UD_ CA_CP_Modificar plan.	1
10	• Mostrar datos de personas.	Muestra los datos de la persona cuando se registra en el acceso.	0122__CENIA_UD_ CA_CP_Mostrar datos persona.	1
11	• Registrar los accesos.	Registrar los accesos.	0122__CENIA_UD_ CA_CP_ Registrar los accesos.	2
12	• Mostrar reportes de persona por categoría.	Crear reportes de persona.	0122__CENIA_UD_ CA_CP_Mostrar reportes persona por categoría.	2
13	• Redistribuir comensales.	Redistribuir a los comensales por los diferentes puntos de control.	0122__CENIA_UD_ CA_CP_Redistribuir comensales.	2
14 15	• Descargar evento. • Activar punto de control.	Descargar evento. Activar punto de control.	0122__CENIA_UD_ CA_CP_Descargar evento. 0122__CENIA_UD_ CA_CP_Activar punto de control.	2
16	• Mostrar mensajes de aviso dada una cantidad de acceso.	Mostrar los mensajes de aviso.	0122__CENIA_UD_ CA_CP_Mostrar mensaje de aviso dada una cantidad de acceso.	2

3.3 Pruebas de software

Las pruebas de software son un factor importante para asegurarse de que se va por el camino correcto para alcanzar la calidad esperada. Son los procesos que permiten verificar y revelar la calidad de un producto. Estas se integran dentro de las diferentes fases del ciclo de vida del software. Son un elemento crítico para la garantía de la calidad y representa una revisión final de las especificaciones, del diseño y de la codificación (Pressman, 2005).

3.3.1 Estrategia de prueba

Según Pressman una estrategia de prueba del software: integra los métodos de diseño de caso de pruebas en una serie bien planeada de pasos que desembocará en la eficaz construcción del software. La estrategia proporciona un mapa que describe los pasos, además de cuánto esfuerzo, tiempo y recursos consumirán (Pressman, 2005).

En la investigación se proponen utilizar algunas de las estrategias definidas por Pressman en su libro *Ingeniería de software: Un enfoque práctico,* en donde se pretende que el producto se pruebe desde la parte más pequeña del software hasta la más grande en forma de espiral usando los niveles, tipos, técnicas y herramientas de prueba (Pressman, 2005).

3.3.2 Niveles de prueba

Tabla 3.11 Niveles de prueba.

Niveles de prueba				
Prueba	**Objetivo**	**Participante**	**Ambiente**	**Método**
Unitario	Detectar errores en los datos, lógica y algoritmos.	Programadores.	Desarrollo.	Caja Blanca.
Integración	Detecta errores de interfaces y relaciones entre componentes.	Programadores.	Desarrollo.	Caja Blanca, Top Down, Bottom Up.
Funcional	Detecta errores en la implementación de requerimientos.	Probadores y Analistas.	Desarrollo.	Caja Negra.

Sistema	Detecta fallas en el cubrimiento de los requerimientos.	Probadores y Analistas.	Desarrollo.	Caja Negra.
Aceptación	Detecta fallas en la implementación del sistema.	Probadores, Analistas y Clientes.	Productivo.	Caja Negra.

A SCAC se le aplicaron pruebas de integración y de sistema las cuales demostraron que la aplicación respondía satisfactoriamente a la integración con otros subsistemas y cumplía con los requisitos definidos.

3.3.3 Tipos de prueba

- ✓ **Unidad:** Centra el proceso de verificación en la menor unidad del diseño del software (Módulo). Aquí se prueban los caminos de control importantes, con el fin de descubrir errores dentro del ámbito de un módulo (Pressman, 2005).
- ✓ **Integración:** Es una técnica sistemática para construir la estructura del programa mientras que al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. Existen dos tipos de integración: la primera es no incremental "*big bang*". Se combinan todos los módulos por anticipado, se prueba todo el producto. La segunda es una integración incremental en donde se desarrollan módulos pequeños y funcionales que hacen que los errores sean más fáciles de aislar y corregir (Pressman, 2005).
- ✓ **Validación:** Puede definirse de muchas formas, pero una simple definición es que la validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente. Existen dos criterios de validación: la prueba alfa se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado. La prueba beta se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación en línea del software en un entorno que no puede ser controlado por el desarrollador (Pressman, 2005).
- ✓ **Recuperación:** Es una prueba del sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente. Si la recuperación es automática hay

que evaluar la corrección de la inicialización, de los mecanismos de recuperación del estado del sistema, de la recuperación de datos y del proceso de re-arranque. Si la recuperación requiere la intervención humana, hay que evaluar los tiempos medios de reparación (TMR) para determinar si están dentro de límites aceptables.

- ✓ **Seguridad:** En este caso el acceso al sistema incluye un amplio rango de actividades: piratas informáticos que intentan entrar en los sistemas por deporte, empleados disgustados que intentan penetrar por venganza e individuos deshonestos que intentan penetrar para obtener ganancias personales ilícitas. La prueba de seguridad intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho y de accesos impropios.
- ✓ **Resistencia:** Ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. Por ejemplo:

  Incrementar las frecuencias de datos de entrada en un orden de magnitud con el fin de comprobar cómo responden las funciones de entrada.

  Diseñar pruebas especiales que generen 10, interrupciones por segundo, cuando las normales son una o dos; ejecutar casos de prueba que requieran el máximo de memoria o de otros recursos; diseñar casos de prueba que puedan dar problemas en un sistema operativo virtual.

El tipo de prueba que se le realizó al subsistema fue la validación donde dentro de la misma se aplicó la Métrica para la Calidad de la Especificación de los Requisitos de Software, la cual consiste en dividir el número de requisitos para todos los revisores que tuvieron interpretaciones idénticas $(\mathbf{Nui})$ entre la cantidad de requisitos de software $(\mathbf{Rt})$.

$$\mathbf{Q1} = \mathbf{Nui}/\mathbf{Rt}, \quad \mathbf{Rt} = \mathbf{R}f + \mathbf{F}nf$$

Donde $\mathbf{R}f$ es la cantidad de requisitos funcionales y $\mathbf{F}nf$ es la cantidad de requisitos no funcionales.

Cuanto más cerca de 1 esté el valor de $\mathbf{Q1}$ menor será la ambigüedad de la especificación (Pressman, 2005).

RT= 19+10 = 29

$$\mathbf{Q1} = \mathbf{Nui}/\mathbf{Rt}$$

Q1=26/29= 0.89

El valor de Q1= 0.89, resultado que demuestra que los requisitos del Subsistema de control de acceso

se encuentran con un alto nivel de especificidad, es decir que la ambigüedad de los requisitos es baja. Es importante destacar que un requisito es ambiguo si tiene múltiples interpretaciones.

3.3.4 Métodos de prueba

Se trata de diseñar pruebas que tengan la probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo. Cualquier producto de ingeniería se puede probar de dos formas (Pressman, 2005):

Pruebas de caja blanca: Desarrollar pruebas de forma que se asegure que la operación interna se ajusta a las especificaciones, y que todos los componentes internos se han probado de forma adecuada. En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y examinado el estado del programa en varios puntos. La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba. Entre los métodos que se realiza está la complejidad ciclomática, que representa el número de caminos independientes del conjunto básico de un programa. Esta medida ofrece al probador de software un límite superior para el número de pruebas que debe realizar para garantizar que se ejecutan al menos una vez cada sentencia. La complejidad ciclomática se calcula mediante la cantidad de nodo predicados (nodo de los cuales salen más de una arista) más 1 o la cantidad de aristas menos la cantidad de nodos más 2 (Pressman, 2005).

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
 File fichero = new File("/lib/security/Punto Control.txt"); 1
 PuntoControl punto1 = null; 1
 String puntoC = jComboBox1.getSelectedItem().toString(); 1
 if (puntoC != "--Seleccione--") { 2
 for (int i = 0; i < puntos.size(); i++) { 3
 if (puntoC.equals(puntos.get.getNombrePuntoControl())) { 4
 if (fichero.exists()) { 5
 punto.setActivo(false); 6
 obj.modificar(punto); 7
 }
 try { 8
 punto1 = puntos.get; 9
 punto1.setActivo(true); 10
 obj.modificar(punto1); 11
 FileOutputStream fos = new FileOutputStream(fichero); 12
 ObjectOutputStream out = new ObjectOutputStream(fos); 13
 out.writeObject(punto1); 14
 } catch (IOException ex) { 15
 Logger.getLogger(AsignarPunto.class.getName()).log(Level.SEVERE, null, ex); 16
 }
 }
 } 17
 } 18
if (puntoC == "--Seleccione--") { 19
 JOptionPane.showMessageDialog(rootPane, "Seleccione un punto de Control"); 20
 }
 this.setVisible(false);// TODO add your handling code here: 21
} 22
```

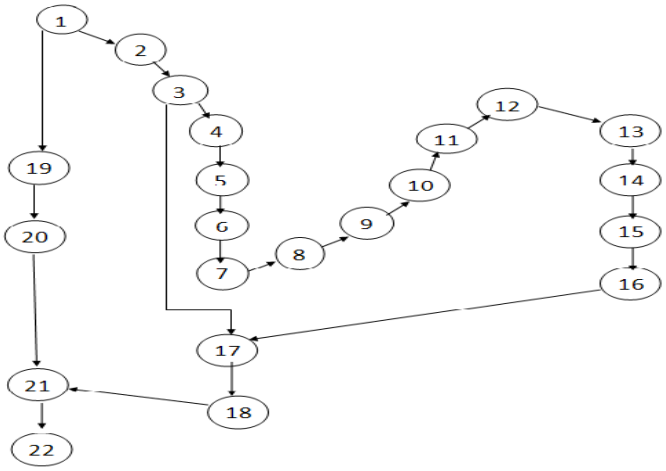

Figura 6 Grafo de flujo o grafo del programa.

V (G) = Arista – Nodo +2

V (G) = 23-22+1= 3

V (G) = Nodo predicado +1

V (G) = 2+1= 3

Después de realizar el cálculo se llega a la conclusión de que el algoritmo presentado anteriormente tiene una complejidad ciclomática de 3, esto evidencia que existen a lo sumo tres caminos básicos por donde recorrer el algoritmo.

En la siguiente tabla se muestran los caminos básicos.

Tabla 3.12 Caminos básicos.

No.	Caminos básicos
1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,21,22
2	1,2,3,17,18,21,22
3	1,19,21,22

Después de conformado los caminos básicos del flujo, se procede a ejecutar los casos de prueba para este procedimiento. Se debe realizar un caso de prueba por cada camino básico. Para realizar los casos de prueba es necesario cumplir con las siguientes exigencias:

Descripción: Se hace la entrada de los datos necesarios, validando que ningún parámetro obligatorio pase nulo al procedimiento y no se entre ningún dato erróneo.

Datos de ejecución: Se especifica cada parámetro para que cumpla una la condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento.

Resultados esperados: Se expone el resultado que se espera devuelva el procedimiento.

Resultado de la prueba realizada a los caminos básico del algoritmo activar punto.

Tabla 3.13 Camino básico #1.

Camino básico #1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,21,22
Descripción	Se activa el punto de control.
Datos de ejecución	PuntoControl punto1 = null , puntoC != "--Seleccione--", puntoC.equals(puntos.get.getNombrePuntoControl) , fichero.exists,
Entrada	punto.setActivo(false), obj.modificar(punto); punto1 = puntos.get; punto1.setActivo(true); obj.modificar(punto1);
Respuesta	Muestra en la pantalla principal un cartel con el nombre del punto activo.

Tabla 3.14 Camino básico #2.

Camino básico #2	1,2,3,17,18,21,22
Descripción	Existe una lista de puntos inactivos, pero no se activa ningún.
Datos de ejecución	If(puntoC == "--Seleccione--") { }
Entrada	String puntoC = "--Seleccione--"
Respuesta	Muestra un mensaje de diálogo que dice "Seleccione un punto de control".

Tabla 3.15 Camino básico #3.

Camino básico #3	1,19,21,22
Descripción	Se seleccionó el punto, pero no se entra en el procedimiento porque no hay conexión.
Datos de ejecución	catch (IOException ex) { Logger.getLogger(AsignarPunto.class.getName()).log(Level.SEVERE, null, ex);}
Entrada	No hay conexión con la base de datos.
Respuesta	El sistema cierra la interfaz.

Pruebas de caja negra: Realizar pruebas de forma que se compruebe que cada función es operativa. En la prueba de la caja negra, 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. Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Casos de prueba

Las siguientes tablas son una muestra de los casos de prueba de las funcionalidades más importantes del subsistema, los restantes pueden encontrarse en el **Anexo 2**.

Tabla 3.16 Caso de prueba Activar punto de control.

Escenario	Descripción	Activar	Respuesta del sistema	Flujo central
EC 1.1 Activar punto de control correctamente.	Se activa el punto de control donde va tener lugar el evento.	V(Puerta 222)	Se activa el punto de control seleccionado y se muestra el mensaje “El punto de control se activo correctamente.”.	1- Se accede al menú “Configuración”. 2- Se selecciona la opción “Activar punto de control”. 3- Se muestra una ventana para llenar los datos (Punto de control activo, Activar (Editable)) y seleccionar el punto de control. 4- Se selecciona el botón “Aceptar” de la ventana y se muestra el mensaje “El punto de control se activo correctamente.”. 5- Se regresa a la ventana principal para realizar cualquier otra acción.

EC 1.2 Activar sin seleccionar el punto de control.	Nos e activa el punto de control que se desea.	I(Vacío)	No se activa el punto de control y se muestra el mensaje "Debe seleccionar un punto de control."	1- Se accede al menú "Configuración". 2- Se selecciona la opción "Activar punto de control". 3- Se muestra una ventana para llenar los datos (Punto de control activo, Activar (Editable)). 4- Se deja el campo Activar vacío y se selecciona el botón "Aceptar". 5- Se muestra el mensaje de información "Debe seleccionar un punto de control." y se da "Aceptar" en el mensaje mostrado regresando a la para activar correctamente el punto de control.
EC 1.3 Cancelar la operación.	Se cancela la operación sobre la activación del punto de control.	NA	Se cancela la operación sobre la activación.	1- Se accede al menú "Configuración". 2- Se selecciona la opción "Activar punto de control". 3- Se muestra una ventana para llenar los datos (Punto de control activo, Activar (Editable)). 4- Se cancela la operación sobre la activación del punto de control y se regresa a la ventana principal realizar otra acción.

Tabla 3.17 Caso de prueba Registrar acceso.

Escenario	Descripción	Lector	Respuesta del sistema	Flujo central
EC 1.1 Registrar acceso correctamente.	Se registra mediante el lector del código de barra el acceso al evento.	V(*******)	El sistema confirma el éxito de la acción en un cuadro de color verde que muestra a la derecha de la pantalla.	1- Se accede a la aplicación y se activa un punto de control y un evento con anterioridad. 2- Se marca el solapín con el lector de código de barra y se verifica el acceso al mismo. 3- Se muestra en un cuadro de texto a la derecha con color verde que se le ha permitido el acceso pues ha reservado para el evento.
EC 1.2 Registrar acceso introduciendo el número de solapín incorrecto.	Introducir en el campo lector el código para el solapín.	I(sgsdgs)	No se permite el acceso y se muestra el mensaje “Entre un código de barra valido.”	1- Se accede a la aplicación y se activa un punto de control y un evento con anterioridad. 2- Se introduce un código de barra incorrecto en el sistema. 3- Se muestra el mensaje “Entre un código de barra valido.” y se da “Aceptar” en el mensaje mostrado. 4- Se limpia el campo del lector y se mantiene la ventana para realizar otra acción.
EC 1.3 Registra acceso de una persona no autorizada.	Se deniega el acceso al evento por no tener autorización.	V(*******)	Muestra en un cuadro de color gris a la derecha de la pantalla el mensaje “Acceso denegado”.	1- Se accede a la aplicación y se activa un punto de control y un evento con anterioridad.

				2- Se introduce un código de barra incorrecto en el sistema. 3- Se muestra en el cuadro inferior a la izquierda en gris el acceso denegado por no estar autorizado.
EC 1.4 Registra acceso de una persona que no reservó.	Leer el código de un solapín que la persona no reservó para ese evento.	V(*******)	Muestra en un cuadro de color azul a la derecha de la pantalla el mensaje “Usted no reservó”.	1- Se accede a la aplicación y se activa un punto de control y un evento con anterioridad. 2- Se introduce un código de barra incorrecto en el sistema. 3- Se muestra en el cuadro inferior a la izquierda en azul el mensaje “Usted no reservó”.

Pruebas de integración

Tabla 3.18 Prueba de Integración Int_1 Subsistema de asignación de comensales.

Caso de Prueba
Número de caso de prueba: Int_1
Subsistema a integrar: Subsistema de asignación de comensales.
Condiciones de ejecución: El Subsistema de asignación de comensales haya introducido los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba: Comprobar que el Subsistema de control de acceso a los comedores de la UCI consulte los datos introducidos en la base de datos por el Subsistema de asignación de comensales.
Entradas/Pasos de ejecución: El Subsistema de asignación de comensales introduce en la base de datos central los datos y el Subsistema de control de acceso a los comedores consulta estos datos y crea el fichero DB4O.
Resultado esperado: Se crea el fichero DB4O con los datos.
Evaluación: Prueba satisfactoria.

Tabla 3.19 Prueba de Integración Int_2 Subsistema de asignación de comensales.

Caso de Prueba
Número de caso de prueba: Int_2
Subsistema a integrar: Subsistema de asignación de comensales.
Condiciones de ejecución: El Subsistema de asignación de comensales creo el fichero DB4O y no exista conexión con la base de datos central.
Descripción de la prueba: Comprobar que el Subsistema de control de acceso a los comedores de la UCI funcione correctamente con el fichero DB4O creado por el Subsistema de asignación de comensales.
Entradas/Pasos de ejecución: El Subsistema de asignación de comensales crea el fichero DB4O y el Subsistema de control de acceso a los comedores verifica la autenticidad del fichero y utiliza los datos que contiene el mismo.
Resultado esperado: El Subsistema de control de acceso a los comedores de la UCI funciona correctamente.
Evaluación: Prueba satisfactoria.

Tabla 3.20 Prueba de Integración Int_3 Subsistema de reservación, planificación y menú.

Caso de Prueba
Número de caso de prueba: Int_3
Subsistema a integrar: Subsistema de reservación, planificación y menú.
Condiciones de ejecución: El Subsistema de reservación, planificación y menú haya introducido los datos de las reservaciones hechas por las personas en la base de datos central.
Descripción de la prueba: Comprobar que el Subsistema de control de acceso a los comedores de la UCI consulte los datos introducidos en la base de datos por el Subsistema de reservación, planificación y menú.
Entradas/Pasos de ejecución: El Subsistema de reservación, planificación y menú introduce en la base de datos central los datos de las reservaciones realizadas por las personas y el Subsistema de control de acceso a los comedores consulta estos datos y crea el fichero DB4O.
Resultado esperado: Se crea el fichero DB4O con los datos.
Evaluación: Prueba satisfactoria.

Resultado de las pruebas aplicadas al subsistema

Al concluir la primera iteración del ciclo de desarrollo fueron analizados 15 requisitos funcionales, obteniéndose 10 no conformidades, de las cuales 3 fueron errores de funcionalidades y 7 de documentación. Concluida la primera iteración de prueba, y resueltas las no conformidades encontradas, comienza una segunda iteración de desarrollo que concluyó con el proceso de prueba de los 19 requisitos funcionales, detectándose 3 no conformidades, de las cuales solo 1 era de funcionalidad y 2 de documentación. Al final de la segunda iteración todas las no conformidades

quedaron resueltas, dando paso a una tercera iteración de pruebas donde no se encontraron no conformidades. De esta forma, concluye el período de pruebas y el subsistema cuenta con la calidad requerida para ser usado.

3.4 Conclusiones parciales

En este capítulo se hace una descripción de las estrategias, niveles, técnicas y métodos de pruebas, que son aplicadas a un software, se define cuáles de las pruebas son más factibles, para comprobar el correcto funcionamiento del subsistema y tratar de eliminar los errores encontrados en el menor tiempo posible. Se diseñan los casos de prueba que serán aplicados al subsistema. Las pruebas realizadas a la aplicación corroboraron el buen funcionamiento de la misma. Además, se verifica la correcta integración de SCAC con el Subsistema de asignación de comensales y el Subsistema de reservación, planificación y menú a través de las pruebas de integración.

Conclusiones generales

Después de realizada la investigación se llega a las siguientes conclusiones:

- Los sistemas homólogos estudiados no se ajustan a las políticas de desarrollo del centro, ni permiten integrarse al Sistema de Planificación y Control del Servicio de Alimentación de la Universidad de las Ciencias Informáticas, sin embargo aportaron funcionalidades como el control de puertas, horarios y permisos, manejo de registro y redistribución de comensales.
- El proceso de desarrollo del software estuvo guiado satisfactoriamente por el proceso de desarrollo ágil con segundo nivel de CMMI, cumpliendo con la obtención de los artefactos definidos.
- Con la implementación de SCAC se obtuvo un subsistema multiplataforma, que aumenta los niveles de organización del proceso y se ajusta a las necesidades de la Dirección de Alimentación de la Universidad de las Ciencias Informáticas.
- Las pruebas funcionales y de integración aplicadas al subsistema, revelaron las no conformidades existentes, que fueron eliminadas culminando el producto con la calidad requerida.

Recomendaciones

A partir de los resultados de la investigación y la experiencia adquirida durante el desarrollo del Subsistema de control de acceso a los comedores de la Universidad de las Ciencias Informáticas, se recomienda:

- Incorporar un reporte que permita conocer en qué instante de un evento se registra la mayor cantidad de accesos.
- Valorar la implantación del subsistema en los comedores de la Universidad de las Ciencias Informáticas.

Bibliografía

1. **Advancedsoft. 2011.** Advancedsoft. *Sistemas de Seguridad y Control de Personal.* [En línea] España e Inglaterra, 5 de Julio de 2011. [Citado el: 23 de octubre de 2011.] http://www.advancedsoft.net/. sn.
2. **Álvarez, Gonzalo. 2012.** Departamento de Tratamiento de la Información y Codificación. *Página principal del TIC.* [En línea] Departamento de Tratamiento de la Información y Codificación del Instituto de Seguridad de la Información del Consejo Superior de Investigaciones Científicas de España., 24 de enero de 2012. [Citado el: 2012 de febrero de 17.] http://www.iec.csic.es/criptonomicon/autenticacion/control.html.
3. **Alvarez, Miguel Angel. 2001.** DesarrolloWeb.com. [En línea] 18 de julio de 2001. [Citado el: 15 de noviembre de 2011.] http://www.desarrolloweb.com/articulos/497.php..
4. **ATI Asociación de Técnicos de Informática. 2009.** Introducción a CMMi. *Introducción a CMMi.* [En línea] ATI Asociación de Técnicos de Informática, 9 de marzo de 2009. [Citado el: 23 de novimbre de 2011.] http://www.ati.es/spip.php?article1135.
5. **Award Support. 2011.** Award Support. *Pharaoh Software.* [En línea] Pharaohsoftware, 2011. [Citado el: 20 de noviembre de 2011.] http://www.pharaohsoftware.com.ar.
6. **Beck, K. y Beedle, M. 2001.** Manifesto for Agile Software Development. *Manifesto for Agile Software Development.* [En línea] 24 de febrero de 2001. [Citado el: 4 de noviembre de 2011.] http://agilemanifesto.org/principles.html.
7. **Boehm, B. 2006.** *A View of 20th and 21st Century Software Engineering, Proceedings of 28 International Conference on Software Engineering*. ACM: Shanga : s.n., 2006.
8. **Booch, G, Rumbaugh, J y Jacobson, I. 1999.** *El Lenguaje Unificado de Modelado.* México : Addison Wesley Iberoamericana, 1999. s.n.
9. **Carreño, Carlos. 2011.** *Introducción a Los Estilos de la Arquitectura del Software.* Lima : Universidad Ricardo Palma, 2011.
10. **Community, Hibernate - Jboss. 2012.** Hibernate - Jboss Community. *Hibernate - Jboss Community.* [En línea] 2012. [Citado el: 20 de enero de 2012.] http://www.hibernate.org/.
11. **Cosentino, Ing. Luis. 2012.** Revista Negocios de Seguridad. *Revista Negocios de Seguridad.* [En línea] 2012. [Citado el: 3 de abril de 2012.] http://www.rnds.com.ar.

12. **Cujilema, Diego Armando. 2011.** *Control de Acceso.* Ríobamba – Ecuador : Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, 2011.

13. **Desarrolloweb. 2010.** Desarrolloweb. *Desarrolloweb.* [En línea] 17 de 04 de 2010. [Citado el: 24 de enero de 2012.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.).

14. **Española, Diccionario Manual de la Lengua. 2007.** TheFreeDictionary. *TheFreeDictionary.* [En línea] Larousse Editorial, S., 2007. [Citado el: 24 de noviembre de 2011.] http://es.thefreedictionary.com/acceso.

15. **Española, Real Academia. 2010.** Diccionario de la Real Academia Española, Vigésima Segunda Edición. *Diccionario de la Real Academia Española, Vigésima Segunda Edición.* [En línea] 2010. [Citado el: 12 de Diciembre de 2011.] http://buscon.rae.es/draeI/.

16. **Feed backelectronics. 2011.** Feed backelectronics. *Sistema Inteligente FEED-FOOD.* [En línea] Feed backelectronics, 2011. [Citado el: 23 de noviembre de 2011.] http://www.feedbackelectronics.com.ar/prod_contcomedores.html.

17. **Fernández, Elena Maydelin y Blaya Chiu, Guillermo. 2010.** *DESARROLLO DE UN SISTEMA PARA LA GESTIÓN DEL CONTROL DE ACCESO EN LA UNIVERSIDAD DE LA CIENCIAS INFORMÁTICAS.* Ciudad de la Habana : UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS, 2010. SN.

18. **Firefox, Mozilla. 2011.** EVOLUS PENCIL. *EVOLUS PENCIL.* [En línea] 2011. [Citado el: 7 de diciembre de 2011.] http://pencil.evolus.vn/en-US/Home.aspx.

19. **Flores, Coronado,Adisley y Rovira, Prieto, Imirys. 2007.** *Sitemas control de acceso a comedores en la Univercidad de las Ciencias Informáticas.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2007.

20. **Gamma, Erich, y otros. 1994.** *Design Patterns: Elements of Reusable Object-Oriented Software.* México : Addison-Wesley, 1994.

21. **Garcia, Juaquin. 2005.** CMM - CMMI Nivel 2. *IngenieroSoftware.* [En línea] 26 de noviembre de 2005. [Citado el: 11 de diciembre de 2011.] http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php.

22. **Guglielmetti, Marcos. 2004.** Mastermagazine. *Mastermagazine.* [En línea] 2004. [Citado el: 23 de enero de 2012.] (http://www.mastermagazine.info/termino/3916.php.

23. **hibernate.org. 2010.** hibernate.org [En línea] [Citado el: 28 de Febrero del 2010] Disponible en: Disponible en: http://www.hibernate.org/). *hibernate.org.* [En línea] 28 de febrero de 2010. [Citado el: 25 de noviembre de 2011.] http://www.hibernate.org/.

24. **HispaNetwork Publicidad y Servicios. 2006.** Definiciones IDEDiccionario de Términos técnicos de Internet. *Glosario.net.* [En línea] 27 de octubre de 2006. [Citado el: 12 de noviembre de 2011.] http://tecnologia.glosario.net/terminos-tecnicos-internet/ide-860.html.

25. **—. 2009.** Diccionario de Términos Bibliotecarios. *Glosario.net.* [En línea] 26 de abril de 2009. [Citado el: 11 de febrero de 2012.] http://cultura.glosario.net/terminos-bibliotecarios/sgbd-12441.html.

26. **IDENTIFIC-CAR. 2006.** IDENTIFIC-CAR. [En línea] 2006. [Citado el: 6 de diciembre de 2011.] http://indentificar.com.ar/ms9520.html.

27. **Integrado, Entorno. 2009.** Entorno Integrado. *Entorno Integrado.* [En línea] 2009. [Citado el: 27 de octubre de 2011.] http://petra.euitio.uniovi.es.

28. **Java 2 Platform, Enterprise Edition (J2EE) Overview. 2009.** Java 2 Platform. *Java 2 Platform.* [En línea] 2009. [Citado el: 7 de diciembre de 2011.] http://java.sun.com/j2ee/overview.html.

29. **Java. 2010.** Java. *Java.* [En línea] 5 de mayo de 2010. [Citado el: 06 de diciembre de 2011.] http://www.java.com/es/download/faq/whatis_java.xml.

30. **Kathleen Jui, Baechli Suan. 2005.** *Comparación de las tecnologías de control de acceso a las instalaciones en una organización.* Guatemala : Universidad de San Carlos de Guatemala, 2005. sn.

31. **Llamazares, Juan Carlos. 2007.** Ecojoven. [En línea] 2007. [Citado el: 16 de noviembre de 2011.] http://ecojoven.com/dos/03/RFID.html.

32. **Lucena, López, Manuel José. 2003.** *Criptografía y Seguridad en Computadores. 3ra ed. 2003.* Manuel José Lucena López. ... Febrero de 2003. : s.n., 2003.

33. **Microsoft. 2012.** Windows Server. *Introducción al control de acceso.* [En línea] Microsoft, 30 de marzo de 2012. [Citado el: 17 de febrero de 2012.] http://technet.microsoft.com/es-es/library/cc785144%28WS.10%29.aspx.

34. **Netbeans.org. 2011.** Netbeans. *Netbeans.* [En línea] Oracle Corporation and/or its affiliates, 2011. [Citado el: 15 de noviembre de 201.] http://www.netbeans.org..

35. **Object Management Group. 2011.** BPMN Information Home. *BPMN Information Home.* [En línea] 23 de octubre de 2011. [Citado el: 23 de enero de 2012.] http://www.bpmn.org/.

36. **Oliveras, Rojas,Juan Carlos. 2010.** *Patrones de Diseño.* México : s.n., 2010.

37. **Paradigm, Visual. 2010.** Visual Paradigm for UML 8.0 Release Notes. *Visual Paradigm for UML 8.0 Release Notes.* [En línea] 16 de agosto de 2010. [Citado el: 06 de diciembre de 2011.] http://www.visual-paradigm.com/support/vpuml/releasenotes/800.jsp#new1.

38. **PdAdmin PostgresSQL Tools. 2010.** PdAdmin PostgresSQL Tools. *PdAdmin PostgresSQL Tools.* [En línea] 2010. [Citado el: 5 de diciembre de 2011.] http://www.pgadmin.org/.

39. **Pérez, Sánchez, Jesús. 2010.** Agil Spain. *Agil Spain.* [En línea] Creative Commons Attribution-ShareAlike 2.5 Spain License, 17 de noviembre de 2010. [Citado el: 20 de noviembre de 2011.] http://www.agile-spain.com/.

40. **Pressman, Roger S. 2005.** *Ingeniería del Software. Un enfoque práctico 5 ta.edición.* España : Impreso en: Imprenta Fareso .S.A, 2005. ISBN:84-481-321-4-9.

41. **Ramírez, Jose Ramón Salazar. 2009.** *Herramienta para modelado y configuración de modelos de características.* MÁLAGA : Escuela Técnica Superior de Ingeniería Informática, 2009. s.n.

42. **Reed, Archie. 2007.** CIO information,news and tips -.com. *SearchCio.* [En línea] 2007. [Citado el: 7 de 12 de 2011.] http://searchcio.techtarget.com/searchCIO/downloads/DGIM_Ch1Excerpt.pdf.

43. **Rodríguez, Berzosa, Luis. 2008.** Black - Portal. [En línea] 29 de noviembre de 2008. [Citado el: 9 de diciembre de 2011.] http://www.bl4ck-p0rtal.com.ar/foro/index.php?topic=988.0. SMF © 2011.

44. **Rodríguez, Berzosa, Luis. 2011.** Control de acceso. *Articulos Invitados.* [En línea] 2011. [Citado el: 11 de noviembre de 2011.] http://www.iec.csic.es/criptonomicon/articulos/expertos69.html. Copyright © 1997-2000.

45. **Saborit, Ing. Yunier. 2011.** *Presentación al Frente Universitario.* Ciudad de la Habana : s.n., 2011.

46. **SEI. 2011.** Software Engineering Institute. *Software Engineering Institute.* [En línea] SEI, 2011. [Citado el: 4 de noviembre de 2011.] http://www.sei.cmu.edu/cmmi/index.cfm.

47. **Sommerville, Ian. 2005.** *[SOMMERVILLE, Ian. Ingeniería del software [en línea]. 7ma Edición [s.l.]: Pearson Educación, 2005 [fecha de consulta: 2 Enero 2012]]:.* s.l. : Pearson Educación, 2005.

48. **Spring Source Commint. 2010.** Spring Source Commint,Aplicaciones Framework. *Spring Source Commint.* [En línea] enero de 2010. [Citado el: 5 de noviembre de 2011.] http://www.springframework.net/..

49. **TarjeSol2005. 2005.** TarjetarSol . *TarjetarSol .* [En línea] 2005. [Citado el: 16 de noviembre de 2011.] http://www.tarjetastarjesol.com/tarjetas-pvc/trajetas-con-codigo-de-barras.php..

50. **The PostgreSQL Global Development Group. 2011.** Postgresql. *Postgresql.* [En línea] 4 de junio de 2011. [Citado el: 16 de noviembre de 2011.] http://www.postgresql.org/.

51. **VERSANT CORP. 2010.** VERSANT CORP. *Java & .Net Object Database. Java & .Net Object .* [En línea] VERSANT CORP, 2010. [Citado el: 5 de diciembre de 2011.] http://www.db4o.com/espanol/.

52. **Yague, Maienma I., Maña, Antonio y Maria Troya, Jose. 2004.** *Libro avances de la criptologia y seguridad de la informacion.* University of Málaga : Computer Science Department, 2004. ISBN84-7978-650-7, desposito legal M.37.736-2004.

53. **Yuan, Eric y Tong, Jin. 2005..** *Attributed Based Access Control (ABAC).* Computer Society Washington, DC, USA : The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2012, 2005. ISBN:0-7695-2409-5.

Glosario de términos

***Array*:** Colección ordenada de elementos de un mismo tipo de datos, agrupados de forma consecutiva en memoria.

CamelCase: Estilo de escritura que se aplica a frases o palabras compuestas.

Código de barra: Es un código basado en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información. Es decir, las barras y espacios del código representan pequeñas cadenas de caracteres.

Comensales: Cada una de las personas que acceden a los comedores.

Chip: Conocido como circuito integrado, está formado por varios componentes miniaturizados, tales como: transistores, resistencias, condensadores y otros.

Evento: Es un suceso de importancia que se encuentra programado, dígase desayuno, almuerzo o comida.

Flash: Programa de edición multimedia desarrollado originalmente por Macromedia (ahora parte de Adobe) que utiliza principalmente gráficos vectoriales, pero también imágenes ráster, sonido, código de programa, flujo de vídeo y audio bidireccional para crear proyectos multimedia. Flash es el entorno desarrollador y *"Flash Player"* es el programa (la máquina virtual) utilizado para ejecutar los archivos generados con flash.

Multiplataforma: Dicho de una aplicación o de un producto informático que puede ser utilizado por distintos sistemas o entornos.

TIC: Tecnologías de la Información y la Comunicación, son el conjunto de tecnologías desarrolladas para gestionar información y enviarla de un lugar a otro.

Tercerizados: Persona que no se encuentra vinculada a los procesos sustantivos de la universidad.

Web: Vocablo inglés que significa "red", "telaraña" o "malla". El concepto se utiliza en el ámbito tecnológico para nombrar a una red informática y, en general, a Internet (en este caso, suele escribirse como web, con la W mayúscula).