UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA INTEGRAL DE INFORMACIÓN PARA LA DIVISIÓN DE CIENCIAS BÁSICAS

T E S I S P R O F E S I O N A L
QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN
P R E S E N T A :
SALOMÓN RAMÍREZ CONTLA

DIRECTORA: ING. IRENE PATRICIA VALDEZ Y ALFARO
CODIRECTOR: ING. CARLOS ROMÁN ZAMITIS

CIUDAD UNIVERSITARIA, MÉXICO D.F. 2008.

Agradecimientos.

A mi familia que quiero mucho y siempre me apoya en todo.

A mis directores y sinodales porque han dedicado su tiempo y esfuerzo a este trabajo.

Índice.

Índice de figuras y códigos.

Capítulo 1. Problema, requerimientos y propuesta de solución.

1.1 Problema.

Debido a la gran cantidad de información que implica llevar registros del personal de la División de Ciencias Básicas es necesario un medio que proporcione un acceso a dicha información de manera rápida y segura al personal autorizado para ello.

Actualmente los registros del personal, la programación semestral de actividades académicas y otros datos son registrados en una base de datos en Microsoft Access 2000 consistente de varios archivos, cada uno con diferentes tablas que contienen la información que es necesario almacenar y modificar constantemente. Estos datos son consultados y modificados por medio de otros archivos de Access que muestran formularios e informes. Estos últimos son los archivos a los que tienen acceso los usuarios regulares. Todos los archivos se encuentran en un servidor Linux y compartidos a clientes Windows por medio del servicio Samba. El problema con esta manera de permitir acceso a la base de datos es que los usuarios que tienen permiso de modificación de los archivos de Access pueden alterar o eliminar datos y/o archivos que se desea no sean modificados. Otro problema de tener tantos archivos es que el mantenimiento del conjunto se dificulta.

La interfaz actual, conformada por los formularios de Access, permite el acceso a la base de datos a múltiples usuarios que cuenten con el sistema operativo Windows, Microsoft Access y cuenta de Samba en el servidor Linux, pero se requiere facilitar el acceso y los medios para administrar el mismo de manera amigable y rápida, así como aumentar las medidas de seguridad para evitar la pérdida de datos y el acceso sin autorización a éstos.

Otro problema es que la base de datos que se maneja actualmente no tiene un diseño lo suficientemente adecuado para poder llevar a cabo mantenimiento de manera sencilla y dificultaría mejoras ulteriores al sistema.

1.2 Requerimientos.

Se requiere un sistema con una interfaz amigable y clara que permita el acceso a una base de datos a usuarios que dispongan de una cuenta en ese sistema. El sistema debe ser capaz de llevar registros de datos personales de personal, manejo de entidades de la división, manejo de currículums y programación académica semestral.

Detalladamente, los requerimientos son:

- **Módulo de personal:**
  Guardar registro de los siguientes datos personales:
  - Título.
  - Nombres y apellidos.

- Dirección particular.
- Dirección de oficina.
- Cédula profesional.
- RFC.
- CURP.
- Teléfonos de casa, celular y de oficina.
- Dirección de correo electrónico.
- Número de trabajador.
- Número de folio.
- Grado académico.
- Licenciatura principal.
- Fecha de ingreso al personal de la DCB.
- Género.
- Estado civil.
- Nacionalidad.

- **Módulo de currículums:**
  Los registros de personal contarán con un currículum capturable y editable por la persona a quien hace referencia el registro y por los administradores del sistema. Este currículum debe contener datos de:

  Grados académicos obtenidos:
  - Nombre.
  - Nivel.
  - Institución.
  - Área de especialización.
  - Años de inicio, fin y titulación.

  Experiencia académica y profesional:
  - Actividad o puesto desempeñado.
  - Institución o empresa.
  - Periodo desempeñado.
  - Logros durante la experiencia académica y profesional.
  - Descripción del campo principal de actividades profesionales.
  - Área de especialidad.

  Materias impartidas el semestre pasado y antepasado con referencia al actual:
  - Nombre.
  - Clasificación.
  - Cantidad de horas de teoría.
  - Cantidad de horas de laboratorio.

  Lista de materias impartidas.

- Nombre.
- Semestre al que corresponde.
- Veces que se ha impartido.
- Años o periodos en que se ha impartido.

Premios recibidos.
- Nombre.
- Institución otorgante.
- Descripción.
- Año.

Asociaciones profesionales.
- Nombre.
- Tipo de membresía.
- Fecha de expedición de membresía.
- Fecha de expiración de membresía.

Productos del quehacer académico.

Cantidades en los siguientes rubros:
- Libros.
- Notas de clase.
- Materiales didácticos.
- Manuales de prácticas.
- Artículos.
- Artículos de divulgación.
- Memorias.
- Patentes.
- Trabajos en la industria.
- Foros.
- Servicios a la industria.
- Convenios con la industria.

Datos de los productos más importantes:
- Tipo de producto.
- Nombre.
- Ficha bibliográfica si aplica.

Así como nombramientos en otras facultades y/o divisiones y tipos de éstos.

- **Módulo de organización de entidades de la DCB:**
  Se debe llevar registro de cada entidad que forma parte de la organización de la división. Cada entidad cuenta con los siguientes campos:

- Nivel de la entidad (división, coordinación, departamento, etc.)
- Área.
- Clave que utiliza USECAD.
- Siglas.
- Ubicación.
- Jefe.
- Entidades dependientes.
- Conjunto de personal que la integra.

- **Módulo de programación académica semestral.**
  Con el sistema se podrá llevar a cabo la programación de actividades semestrales de la División: programar fechas de exámenes y asignar horarios y salones de clase a cada grupo de las diferentes asignaturas.

  De cada asignatura se guardará:
  - Semestre en el que se imparte.
  - Clave.
  - Entidad a la que pertenece.
  - Número de créditos.
  - Plan.
  - Horas semanales.
  - Horas de laboratorio a la semana.
  - Tipo (teoría o laboratorio).

  De cada grupo se guardará:
  - Número de grupo.
  - Bloque (si se trata de un grupo de primer semestre).
  - Asignatura.
  - Cupo.
  - Cantidad de oyentes que se aceptan.
  - Cantidad de alumnos inscritos.
  - Profesor que imparte la clase.
  - Salón.
  - Horario.
  - Sección (A, B o C).
  - Si es de primer ingreso o no.
  - Cantidad de alumnos calificados con NP, 5, 6, 7, 8, 9 y 10.

Otros requerimientos del sistema son:

- **Acceso remoto seguro.**
  El sistema debe poder ser accedido remotamente por los diferentes tipos de usuario únicamente desde direcciones IP especificadas dentro de la División de Ciencias Básicas de la Facultad de Ingeniería. Únicamente se podrá acceder al sistema con cuenta.

- **Roles.**
  Existirán diferentes tipos de cuenta (administradores, profesores, etc.) y cada tipo de cuenta sólo tendrá acceso a ciertos módulos del sistema. Por ejemplo, una cuenta de tipo profesor no contará con permisos para crear cuentas o modificar datos del personal. Y para la programación, cada usuario sólo podrá modificar grupos de las asignaturas de su entidad.

- **Facilidad de uso.**
  El sistema debe contar con una interfaz amigable e intuitiva que indique bien qué tipo de información está desplegando y las operaciones sean fáciles de llevar a cabo.

- **Mantenimiento.**
  Hacer cambios al sistema debe ser sencillo. La documentación debe ser completa.

- **Validación de datos de entrada.**
  El sistema debe ser capaz de validar cierto tipo de información ingresada por los usuarios antes de almacenarla en la base de datos.

- **Consistencia e integridad en la base datos.**
  Como en cualquier otro sistema que utiliza una base de datos, un buen diseño en ésta es indispensable.

- **Extensibilidad.**
  Se requiere que cuente con características que permitan añadir funcionalidad posteriormente.

- **Escalabilidad.**
  Inicialmente el número de usuarios del sistema no es muy grande, pero debe poder aumentar sin que esto represente alguna dificultad.

1.3 Propuesta de solución.

He propuesto un sistema computacional con arquitectura cliente liviano-servidor como solución: una aplicación web. Esto se debe a que resulta más sencillo para los usuarios acceder a una página de internet para ver y modificar datos que entrar a una carpeta compartida en un ambiente Windows, abrir un archivo de base de datos Access. Además así los usuarios sólo verán la información que les compete. Algunas otras de las ventajas de una aplicación de este tipo:

Ventajas de la arquitectura cliente-servidor:

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. La seguridad es implementada en el servidor, no en los clientes.

- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.

- Encapsulación (ocultamiento de información): los clientes perciben siempre una interfaz estable sin percatarse de los posibles cambios internos en el servidor. Por lo tanto no es necesario actualizar nada en los clientes ya que los cambios en la aplicación se ven reflejados inmediatamente en las vistas (páginas) que entrega el servidor.

Ventajas de la utilización de clientes livianos en la arquitectura cliente-servidor:

- Pocos requerimientos: los usuarios del sistema sólo necesitan un navegador de internet y acceso a la red para usar la aplicación. Todo el procesamiento de negocio es realizado por el servidor. No es necesario tener nada más instalado.

- Alta accesibilidad: la aplicación puede ser empleada desde cualquier plataforma de sistema operativo y con cualquier navegador, a cualquier hora y desde los lugares que se autorice.

- Actualización del cliente no requerida: cuando un usuario accede al sistema y solicita un recurso, su aplicación cliente siempre recibe contenidos actualizados y con los formatos más recientes del sistema. Los clientes no necesitan descargar nada extra para hacer uso de la versión más actual de la aplicación en el servidor.

Desventajas:

- En ciertos casos las aplicaciones web incrementan el tráfico de una red. La respuesta del sistema depende de la congestión de tráfico en la red. También, si el acceso es desde un lugar lejano, la respuesta puede ser lenta en comparación con una aplicación de escritorio.

- En una aplicación muy concurrida y con contenidos que son modificados muy frecuentemente, en ocasiones los clientes desplegarán información desactualizada. Esto se debe a que a pesar de que cuando la reciben ésta es actualizada, si otro usuario la modifica inmediatamente después de haber sido recibida por el primer cliente, el cambio sólo se mostrará hasta su próxima solicitud al servidor. Esto se puede solucionar con tecnologías asíncronas como AJAX pero demanda una carga considerable de codificación. Cabe enfatizar que esta situación sólo se produce en sistemas con contenidos accedidos y modificados muy frecuentemente y la aplicación que se desarrolla en la presente tesis no es el caso.

Capítulo 2. Conceptos básicos.

2.1 Ingeniería de software.

La elaboración de un sistema computacional comprende mucho más que simplemente la programación de software. El proceso debe seguir una secuencia de pasos contribuyentes a un entendimiento pleno del problema y así tener un panorama claro de la situación. Lo anterior facilita el proceder al diseño y la implementación del programa que dará solución al problema. La ingeniería de software es un enfoque sistémico para el desarrollo, la operación, el mantenimiento y retiro del software.

Durante este desarrollo se utiliza un modelo de las actividades involucradas. Con éste se determina el orden de las etapas y los criterios de transición entre ellas. Este modelo es llamado **ciclo de vida del software** y sus etapas son**:**

Análisis. Es el proceso para definir claramente el problema. Esto conlleva entrevistas, investigación, revisión de terminología, estudio del sistema existente, descripción de necesidades de información y de los límites de la aplicación.

Diseño. Con base a lo anterior ahora es posible realizar una propuesta de solución, definir los pasos a seguir, evaluar las opciones, definir datos que serán manejados y las funciones que deben realizarse con ellos, la elección de algoritmos y estructuras de datos.

Codificación o desarrollo. En esta etapa se elige el lenguaje de programación, el entorno de desarrollo. Se definen los módulos y se escribe código.

Pruebas. Consiste en la puesta en marcha del sistema para su revisión. Se hacen comparaciones de salidas previstas con salidas obtenidas, corrección de errores, mejoras, y estudios de eficiencia.

Mantenimiento. Son instalaciones, actualizaciones, adiciones de funcionalidad, reparaciones de errores al sistema para que siga siendo funcional.

En la Figura 2—1 vemos la manera en que cada etapa se relaciona con las demás durante el ciclo. Las flechas indican la transición entre etapas.

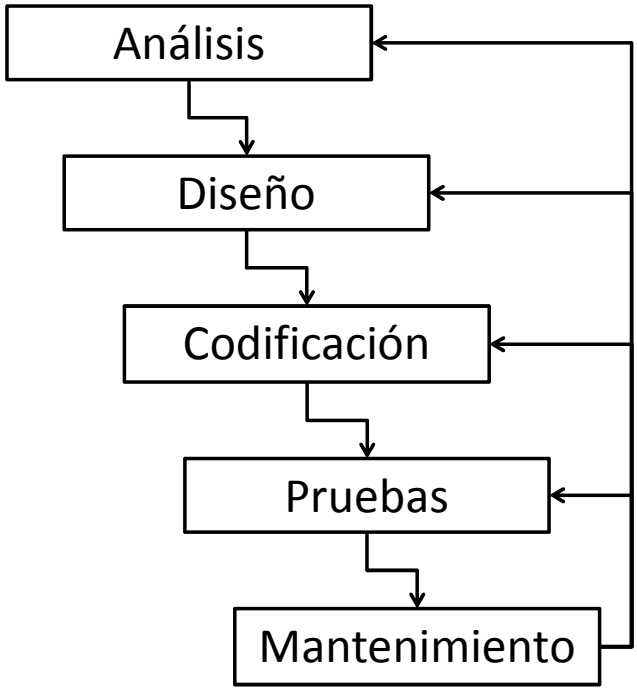


Figura 2—1 El ciclo de vida del software.

2.2 Paradigmas de programación y el paradigma orientado a objetos.

Un paradigma de programación es un estilo de programación derivado de una manera específica de representación y manipulación de conocimiento. Durante el desarrollo de software el paradigma provee y determina el enfoque del programador sobre cómo debe funcionar un sistema computacional. Por ejemplo, en el paradigma funcional un programa puede ser concebido como una secuencia de evaluaciones matemáticas mientras que en el paradigma orientado a objetos el programador lo concibe como un conjunto de objetos que interactúan entre sí.

Así como diferentes grupos de ingeniería de software defienden diferentes metodologías, diferentes lenguajes de programación defienden diferentes paradigmas. Algunos lenguajes son diseñados para reforzar a un paradigma en particular, y otros lenguajes tratan de utilizar varios paradigmas a la vez.

A continuación se mencionan algunos paradigmas de programación y los lenguajes en los que se implementan:

Imperativo.

Describe la computación como un conjunto de declaraciones que cambian el estado de un programa. Se basa también en llamadas a rutinas y subrutinas, métodos o funciones que contienen una serie de pasos computacionales que se llevan a cabo para obtener un resultado, y éstas rutinas pueden ser llamadas en cualquier parte del programa y por cualquier rutina o subrutina. C, Pascal, Perl y BASIC son ejemplos de este paradigma.

Declarativo.

Consiste en crear una aplicación con base en componentes lógicos que especifican o declaran un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. Pero ésta es encontrada por medio de inferencias lógicas, sin indicar exactamente qué pasos seguir, a diferencia del paradigma imperativo. Erlan, Prolog implementan el paradigma declarativo.

Funcional.

Este considera la computación como la evaluación de funciones matemáticas y evita los datos y los estados variables. Enfatiza la utilización de funciones, en contraste con el paradigma imperativo que enfatiza los cambios en el estado del programa. Ejemplos: LISP, Scheme.

El paradigma Orientado a Objetos.

Se basa en el concepto de objeto. Un objeto es aquello que tiene estado (atributos definidos), comportamiento (métodos) e identidad (características que lo definen como único). El conjunto de atributos y métodos de objetos similares están definidos en una clase. Un objeto es una instancia, una concretización de lo que define abstractamente una clase.

Principios del paradigma orientado a objetos:

- **Abstracción.** Consiste en una descripción simplificada o especificación de un objeto real, que enfatiza atributos o propiedades de interés para nuestros fines mientras que suprime los que no son necesarios.

- **Encapsulación.** Se trata de ocultar los detalles internos de un objeto, tales como su funcionamiento al resto de objetos que interactúan con él en el mismo sistema.

- **Modularidad.** Un sistema debe poder ser descompuesto en un conjunto de módulos coherentes e independientes. Esto facilita el trabajo de las partes por separado sin tener que alterar siempre el sistema en su totalidad.

- **Jerarquía o herencia.** Las abstracciones pueden ser organizadas jerárquicamente por niveles de generalización o especialización. Una clase que es una especialización de otra hereda los atributos y métodos de la clase original. Esto evita la repetición de código.

En menor grado, otros de sus principios son:

- **Tipificación.** La definición precisa de un objeto de tal forma que objetos de diferentes tipos no puedan ser intercambiados.

- **Concurrencia.** Es la propiedad que distingue un objeto que está activo de uno que no lo está.

- **Persistencia.** Es la propiedad de un objeto de permanecer en el tiempo. Es decir, de conservar su información durante periodos de tiempo indefinidos.

2.3 Patrones de diseño y el patrón MVC.

Los patrones son lineamientos para la codificación de un sistema computacional. Ofrecen soluciones probadas para problemas definidos y no triviales, de modo que describen el contexto en el que se pueden aplicar y los elementos que estarán involucrados.

Son los desarrolladores de software quienes, con base en la experiencia adquirida elaborando sistemas, crean los patrones. Por ejemplo, los desarrolladores de aplicaciones web crean patrones de diseño para ese contexto y otros desarrolladores los pueden utilizar o tomarlos como una guía para desarrollar sus propias aplicaciones.

Objetivos de los patrones de diseño:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Utilizar un patrón de diseño no es indispensable pero en los casos en que se trabaja sobre un problema que ya ha sido solucionado con anterioridad, su uso facilita mucho el desarrollo del sistema. Tampoco es necesario apegarse cien por ciento a los lineamientos de un patrón. Hay que recordar que sólo son guías.

El patrón MVC.

El Modelo-Vista-Controlador es un patrón ampliamente utilizado en aplicaciones interactivas. Éste divide claramente la funcionalidad de los objetos involucrados en el manejo y presentación de los datos para minimizar el grado de acoplamiento entre ellos. Relaciona (mapea) las tareas comunes de una aplicación (entrada, procesamiento y salida de datos) con el modelo gráfico de interacción que emplea el usuario. Así, se encapsula la funcionalidad del sistema y el usuario sólo tiene que interactuar con las vistas proporcionadas.

El MVC divide la aplicación en tres capas y desacopla sus respectivas responsabilidades. Cada capa lleva a cabo tareas específicas y delega el resto a las otras dos capas. Las tres capas son:

- Un **modelo** que representa los datos de negocio y la lógica u operaciones del mismo que dictan el acceso y la modificación de los datos reales. El modelo notifica a las vistas sobre sus cambios y le permite hacer consultas sobre su estado.

- Una **vista** o capa de presentación que despliega el contenido del modelo. Accede a la información y especifica cómo debe ser mostrada. Actualiza la presentación cuando cambia el modelo y también permite al usuario enviar datos de entrada a un controlador.

- Un **controlador** define la conducta de la aplicación. Despacha las solicitudes del usuario y selecciona qué vistas serán entregadas como respuesta. Interpreta la información de entrada del usuario y la relaciona con acciones que serán realizadas por el modelo. En una aplicación gráfica el *input* del usuario incluye el hacer clic en un botón o seleccionar una opción de un menú. En una aplicación web este *input* son las solicitudes HTTP GET y POST que se envían al servidor. Generalmente una aplicación tiene un controlador por cada conjunto de funcionalidades relacionadas.

La Figura 2—2 muestra las relaciones entre las capas de modelo, vista y controlador dentro de una aplicación basada en el patrón MVC.

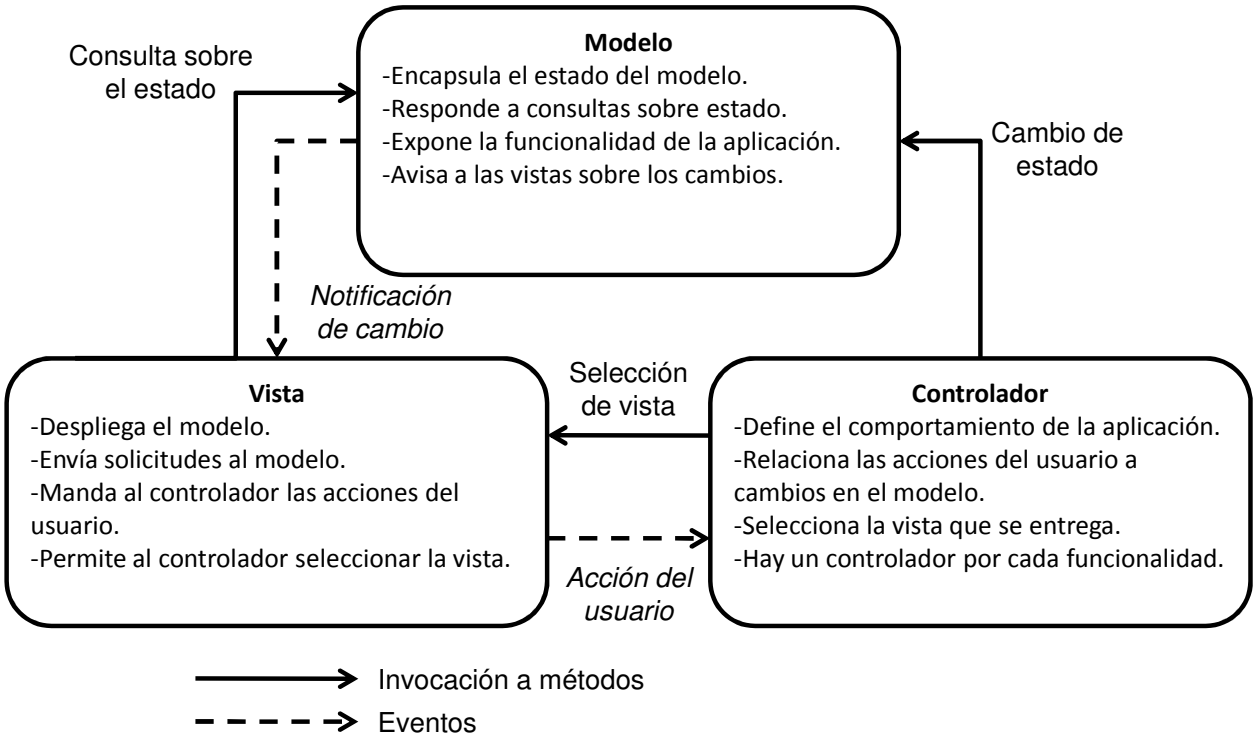


Figura 2—2 El patrón MVC.

La separación de responsabilidades entre los objetos de modelo, los de vista y los de controlador reduce la duplicación de código y hace más fácil el mantenimiento de una aplicación. También hace más sencillo el manejo de los datos ya sea que se añadan nuevos depósitos de datos o se cambie la presentación de éstos porque la lógica de negocio se mantiene separada de los datos. Y si se desea agregar nuevos tipos de clientes que utilicen estos datos no es necesario cambiar la lógica de negocio sino simplemente crear las nuevas vistas que se entregan a cada nuevo tipo de cliente.

2.4 El lenguaje unificado de modelado (UML).

UML es un estándar para notación y diagramas de modelaje de sistemas orientados a objetos. Describe la semántica esencial de los diagramas y símbolos utilizados por los desarrolladores para describir los sistemas. UML empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

UML es independiente del ciclo de desarrollo que se sigue, por lo tanto puede aplicarse en un tradicional ciclo en cascada, o en un ciclo en espiral o incluso en los métodos de desarrollo ágil. Es importante aclarar que UML no es un método de desarrollo: no describe cómo llevar el análisis al diseño y de éste al código; no describe los pasos para producir código a partir de unas especificaciones.

Este lenguaje se puede usar para modelar sistemas de software, hardware y de otros tipos, y los diagramas que especifica son:

- Diagramas de Casos de Uso para modelar los procesos de negocio.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

Estos diagramas son realizados durante el análisis y el diseño de un sistema computacional y permiten visualizar la estructura que tendrá un sistema a desarrollar.

Ya que en el presente trabajo se incluyen algunos diagramas de UML, se explica brevemente a continuación la manera de interpretar algunos de dichos diagramas.

Diagramas de casos de uso.

Los casos de uso son una técnica para capturar información de cómo un sistema o negocio trabaja o de cómo se desea que trabaje. Es una técnica para captura de requisitos.

Los casos de uso describen por medio de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario. Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.

Un punto importante acerca de los casos de uso es que son descripciones de la funcionalidad de un sistema independientes de la implementación que se siga. También resulta importante señalar que están basados en el lenguaje natural, es decir, en un lenguaje entendible para cualquier tipo de usuario.

Con los casos de uso se puede dividir el conjunto de necesidades del sistema de acuerdo al tipo de usuarios que participan en el mismo. Así es posible hacer un diagrama de casos de uso que permita ver claramente las acciones accesibles a los administradores, por ejemplo, y en otro diagrama las correspondientes a usuarios comunes. Los elementos básicos de un diagrama de casos de uso son los actores y los casos de uso (acciones).

Tipos de actores en un diagrama de casos de uso:

- Principales: personas que usan el sistema.
- Secundarios: personas que mantienen o administran el sistema.
- Material externo: dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- Otros sistemas: sistemas con los que el sistema interactúa.

Ocasionalmente una misma persona física puede interpretar varios papeles como actores distintos, el nombre del actor describe el papel desempeñado.

Para identificar y especificar un caso de uso se deben analizar del actor sus secuencias de interacción y los escenarios donde participa desde el punto de vista del usuario del sistema. Los casos de uso intervienen durante todo el ciclo de vida de software. El proceso de desarrollo estará dirigido por los casos de uso ya que estos muestran los requerimientos de la aplicación. Un escenario es una instancia de un caso de uso.

UML define cuatro tipos de relación en los Diagramas de Casos de Uso:

- Comunicación: indica que hay comunicación entre un actor y/o un caso de uso (acción). Se indica con una línea que conecta ambos elementos.

- Inclusión: una instancia de un caso de uso origen incluye también el comportamiento descrito por un caso de uso destino. Para indicar esta relación se utiliza la etiqueta «include» (antes de UML 2.0 se utilizaba «uses»).
- Extensión: un caso de uso origen extiende el comportamiento del caso de uso destino. Se utiliza la etiqueta «extend».
- Herencia: un caso de uso origen hereda la especificación del caso de uso destino y posiblemente la modifica y/o amplía.

Parámetros para la construcción de un caso de uso.
Un caso de uso debe ser simple, claro y conciso. Generalmente son pocos actores asociados a cada caso de uso. Las preguntas clave para definir un caso de uso son:

- ¿cuáles son las tareas del actor?
- ¿qué información crea, guarda, modifica, destruye o lee el actor?
- ¿debe el actor notificar al sistema los cambios externos?
- ¿debe el sistema informar al actor de los cambios internos?

Otra parte importante de los casos de uso, además de los diagramas son sus descripciones. Resultan muy útiles los documentos que describen cada caso de uso de un diagrama. En estos documentos se explica la forma de interactuar entre el sistema y el usuario.

La descripción del caso de uso comprende:
- el inicio: ¿cuándo y qué actor lo produce?
- el fin: ¿cuándo se produce y qué valor devuelve?
- la interacción actor-caso de uso: ¿qué mensajes intercambian ambos?
- objetivo del caso de uso: ¿qué lleva a cabo o intenta?
- cronología y origen de las interacciones.
- repeticiones de comportamiento: ¿qué operaciones son iteradas?
- situaciones opcionales: ¿qué ejecuciones alternativas se presentan en el caso de uso?

Aunque siempre es necesario incluir todos los elementos mencionados en la descripción, sí es importante incluir las definiciones con información que resulte suficiente para explicar claramente cada caso de uso y para que con base en éstas se pueda llevar a cabo la implementación.

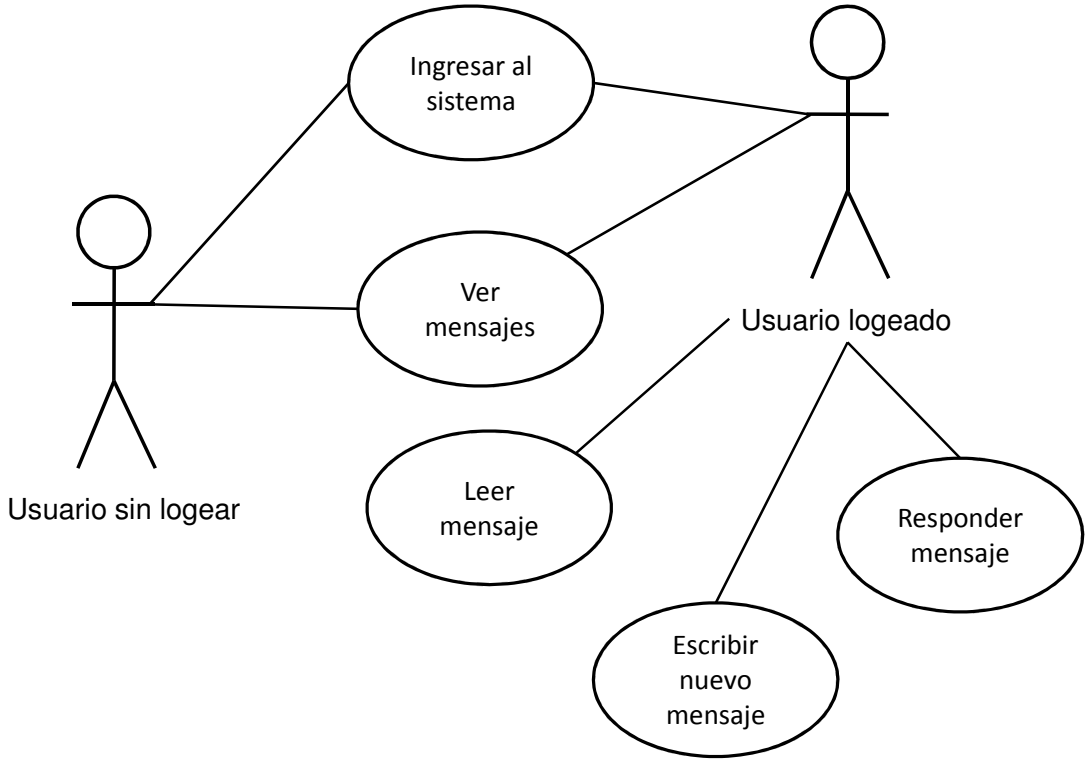


Figura 2—3 Ejemplo de un diagrama de casos de uso con 2 actores y 5 casos de uso.

Diagramas de actividades.

En un diagrama de actividades se muestran estados de actividad y las transiciones entre estos para llegar al término de una actividad que incluye al resto. Con ellos puede describirse con detalle un caso de uso, por ejemplo.

El nivel de detalle de la actividad puede ser muy general o muy preciso dependiendo de a quién va dirigido el diagrama. En un diagrama conceptual y por lo tanto más general, la actividad es alguna tarea que debe ser realizada y se describe a grandes rasgos; en un diagrama de especificación o de implementación donde se requiere más detalle, la actividad es un método de una clase, por ejemplo. Generalmente se utilizan para describir los pasos de un algoritmo.

Un diagrama de actividades se puede usar para entender el comportamiento de alto nivel de la ejecución de un sistema, sin profundizar en los detalles internos de los mensajes. Los parámetros de entrada y salida de una acción se pueden mostrar usando las relaciones de flujo que conectan la acción y un estado de flujo de objeto.

En el diagrama, cuando un estado de actividad termina, la ejecución procede al siguiente estado de actividad dentro del diagrama. Una transición de terminación es activada en un diagrama de actividades cuando se completa la actividad precedente.

Un grafo de actividades puede contener también estados de acción, que son similares a los de actividad pero son atómicos y no permiten transiciones mientras están activos. Los estados de acción se deben utilizar para las operaciones cortas de mantenimiento.

Un diagrama de actividades puede contener bifurcaciones, así como divisiones de control en hilos concurrentes. Los hilos concurrentes representan actividades que se pueden realizar concurrentemente por los diversos objetos o personas. La concurrencia se representa a partir de la agregación, en la cual cada objeto tiene su propio hilo. Las actividades concurrentes se pueden realizar simultáneamente o en cualquier orden.

Un estado de actividad se representa como una caja con los extremos redondeados que contiene una descripción de actividad. Las transacciones simples de terminación se muestran como flechas. Las ramas se muestran como condiciones de guarda en transiciones o como diamantes con múltiples flechas de salida etiquetadas. Una división o una unión de control se representa con múltiples flechas que entran o salen de la barra gruesa de sincronización.

Cuando es necesario incluir eventos externos, la recepción de un evento se puede mostrar como un disparador en una transición, o como un símbolo especial que denota la espera de una señal.

A menudo las actividades son organizadas en un modelo según su responsabilidad. Esta clase de asignación puede mostrarse organizando las actividades en regiones distintas separadas por líneas en el diagrama que debido a su aspecto, se conocen como “calles”.

Un diagrama de actividades puede mostrar el flujo de objetos como valores. Para un valor de salida, se dibuja una flecha con línea discontinua desde la actividad al objeto. Para un valor de entrada, se dibuja una flecha con línea discontinua desde el objeto a una actividad.

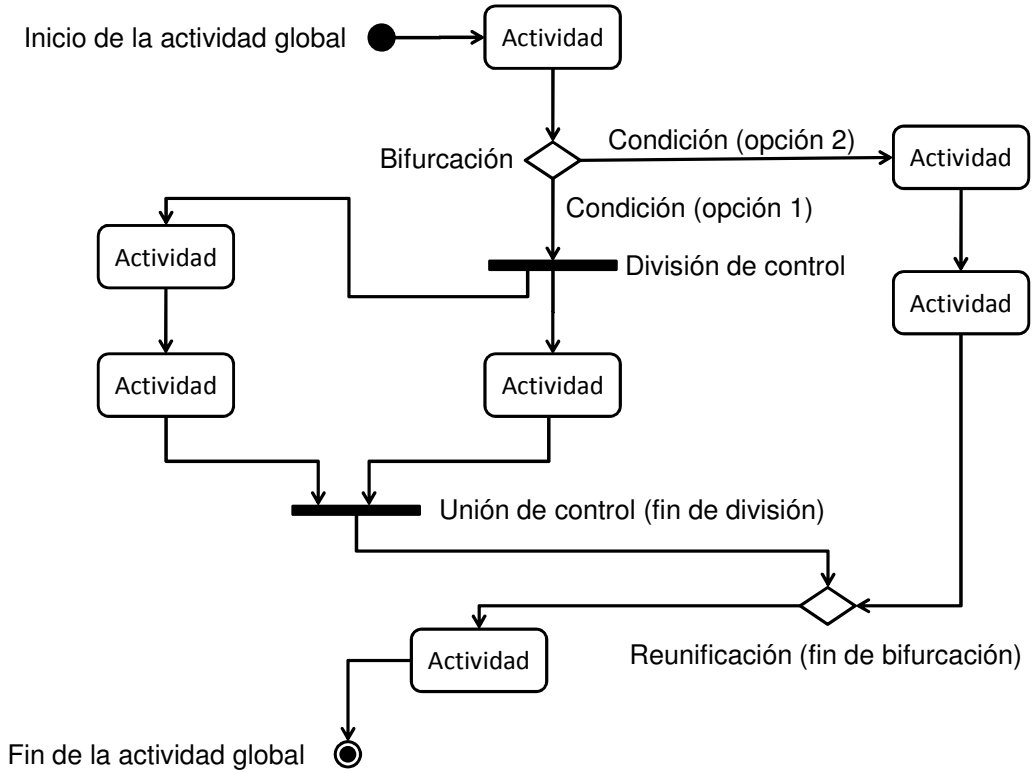


Figura 2—4 Ejemplo de un diagrama de actividades con una división de control y una bifurcación.

Diagramas de Interacción: Diagramas de Secuencia y Diagramas de Colaboración.

En UML, la vista de interacción describe secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Un rol clasificador es la descripción de un objeto, que desempeña un determinado papel dentro de una interacción, distinto de los otros objetos de la misma clase. Esta vista proporciona una vista integral del comportamiento del sistema, es decir, muestra el flujo de control a través de muchos objetos. La vista de interacción se exhibe en dos diagramas que se

centran en distintos aspectos pero complementarios. Uno se centra en los objetos individuales y el otro en objetos cooperantes.

Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones. Los diagramas de interacción muestran cómo se comunican los objetos dentro de estas interacciones. Existen dos tipos de diagramas de interacción: el Diagrama de Colaboración y el Diagrama de Secuencia.

El Diagrama de Secuencia es más adecuado para observar la perspectiva cronológica de las interacciones, muestra la secuencia explícita de mensajes y son mejores para especificaciones de tiempo real y para escenarios complejos. El Diagrama de Colaboración ofrece una mejor visión espacial mostrando los enlaces de comunicación entre objetos, muestra las relaciones entre objetos y son mejores para comprender todos los efectos que tiene un objeto y para el diseño de procedimientos. El diagrama de Colaboración puede obtenerse automáticamente a partir del correspondiente diagrama de Secuencia (o viceversa).

Características de los diagramas de secuencia:

- Muestran la secuencia de mensajes entre objetos durante un escenario concreto.
- Cada objeto viene dado por una barra vertical.
- El tiempo transcurre de arriba abajo.
- Cuando existe demora entre el envío y la atención se puede indicar usando una línea oblicua.

Características de los diagramas de colaboración:

Son útiles en la fase exploratoria para identificar objetos.
La distribución de los objetos en el diagrama permite observar adecuadamente la interacción de un objeto con respecto de los demás.
La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces.

Colaboración.

Una colaboración es la descripción de una colección de objetos que interactúan para llevar a cabo cierto comportamiento dentro de un contexto. Describe un conjunto de objetos cooperantes unidos para realizar un propósito. Una colaboración contiene ranuras que son rellenadas por los objetos y enlaces en tiempo de ejecución. Una ranura de colaboración se llama “rol” porque describe el propósito de un objeto o un enlace dentro de la colaboración.

Un rol clasificador representa una descripción de los objetos que pueden participar en una ejecución de la colaboración, un rol de asociación representa una descripción de los enlaces que pueden participar en una ejecución de colaboración. Un rol de clasificador es una asociación que está limitada por tomar parte en la colaboración. Las relaciones entre roles de clasificador y asociación dentro de una colaboración sólo

tienen sentido en ese contexto. En general fuera de ese contexto no se aplican las mismas relaciones.

Una colaboración tiene un aspecto estructural y un aspecto de comportamiento. El aspecto estructural es similar a una vista estática: contiene un conjunto de roles y relaciones que definen el contexto para su comportamiento. El comportamiento es el conjunto de mensajes intercambiados por los objetos ligados a los roles. Tal conjunto de mensajes en una colaboración se llama Interacción. Una colaboración puede incluir una o más interacciones.

Interacción.
Interacción es el conjunto de mensajes intercambiados por los roles clasificadores a través de los roles de asociación. Un mensaje es una comunicación unidireccional entre dos objetos, un flujo de objeto con la información de un remitente a un receptor. Un mensaje puede tener parámetros que transporten valores entre objetos. Un mensaje puede ser una señal (mensaje) o una llamada (la invocación una operación). Un patrón de intercambios de mensajes que se realizan para lograr un propósito específico es lo que se denomina una interacción.

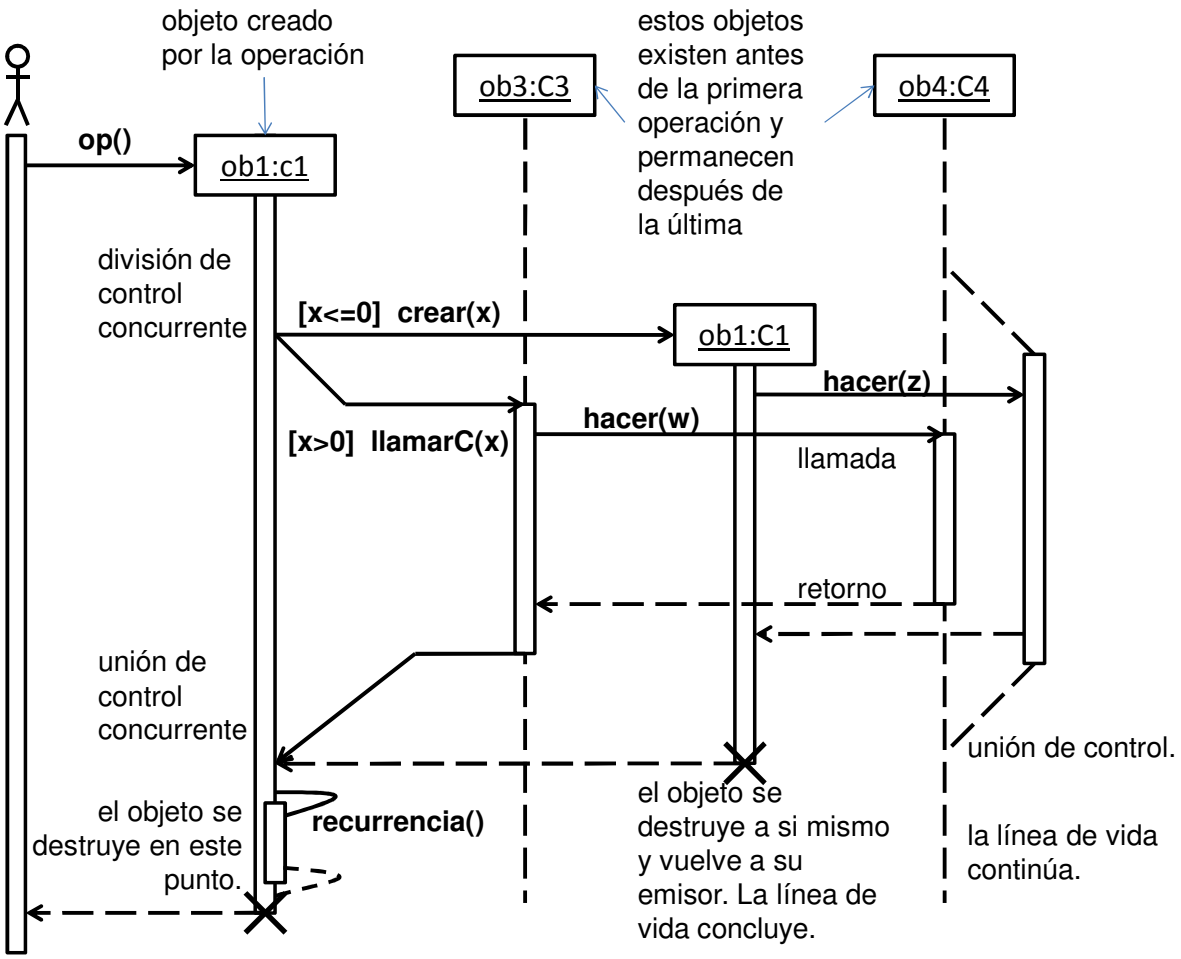


Figura 2—5 Ejemplo de diagrama de secuencia con división de control.

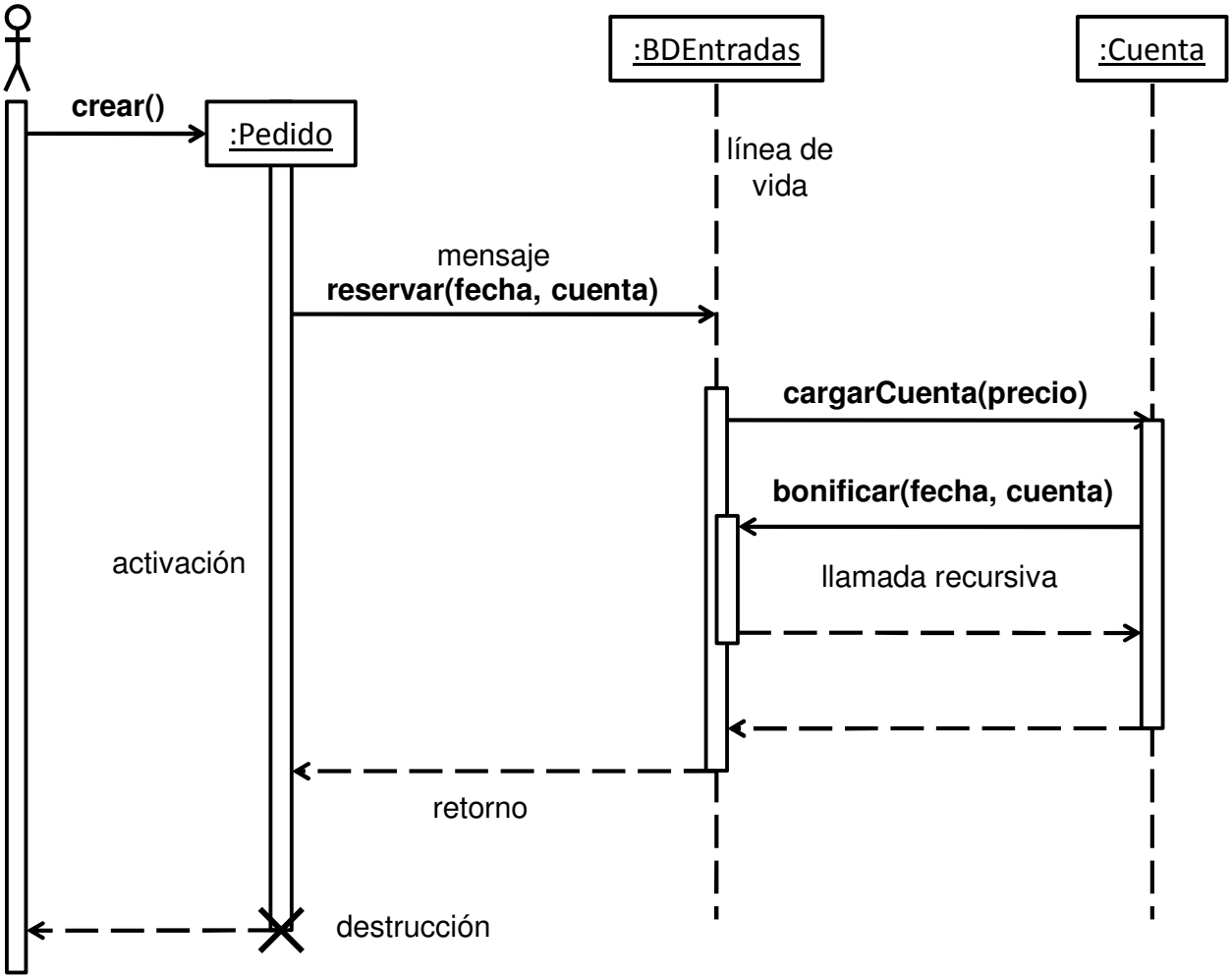


Figura 2—6 Ejemplo de diagrama de secuencia.

Diagramas de clase.

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones.

El modelado de un problema del mundo real puede llevarse a cabo de distintas maneras o formas de abstracción. Algunas formas de abstracción son:

- Clasificación / Instanciación.
- Composición / Descomposición.
- Agrupación / Individualización.
- Especialización / Generalización.

La clasificación es uno de los mecanismos de abstracción más utilizados. La clase define el ámbito de definición de un conjunto de objetos, y cada objeto pertenece a una clase; los objetos se crean por instanciación de las clases.

Cada clase se representa en un rectángulo con tres partes, mostrando cada una:

- Nombre de la clase.
- Atributos de la clase.
- Operaciones de la clase.

Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos. Por esta razón se crearon niveles de visibilidad para los elementos que son:

- (-) Privado : es el más fuerte. Esta parte es totalmente invisible (excepto para clases **friends** en terminología C++).
- (#) Los atributos/operaciones protegidos están visibles para las clases **friends** y para las clases derivadas de la original.
- (+) Los atributos/operaciones públicos son visibles a cualquier otra clase (aunque cuando se trata de atributos se está transgrediendo el principio de encapsulación).

Relaciones entre clases:
Los enlaces entre objetos pueden representarse entre las respectivas clases y sus formas de relación son:

- Asociación y Agregación (vista como un caso particular de asociación).
- Generalización/Especialización.

Las relaciones de Agregación y Generalización forman jerarquías de clases.

Asociación: expresa una conexión bidireccional entre objetos. Una asociación es una abstracción de la relación existente en los enlaces entre los objetos. Puede determinarse por la especificación de multiplicidad (mínima...máxima):

- Uno y sólo uno.
- 0..1 Cero o uno.
- M..N Desde M hasta N (enteros naturales).
- * Cero o más.
- 0..* Cero o más.
- 1..* Uno o más.

Agregación: representa una relación “parte de” entre objetos. En UML se proporciona una escasa caracterización de la agregación. Esta relación puede ser caracterizada con precisión determinando las relaciones de comportamiento y estructura que existen entre el objeto agregado y cada uno de sus objetos componentes.

Generalización: permite gestionar la complejidad mediante un ordenamiento taxonómico de clases, se obtiene usando los mecanismos de abstracción de Generalización y/o Especialización. La Generalización consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general. Los nombres usados: clase padre - clase hija. Otros nombres: superclase - subclase, clase base - clase derivada. Las subclases heredan propiedades de sus clases padre, es decir, atributos y operaciones (y asociaciones) de la clase padre están disponibles en sus clases hijas. La Generalización y Especialización son equivalentes en cuanto al resultado: la jerarquía y herencia establecidas. Generalización y Especialización no son operaciones reflexivas ni simétricas pero sí transitivas. La especialización es una técnica muy eficaz para la extensión y reutilización.

La noción de clase está próxima a la de conjunto. Dada una clase, podemos ver el conjunto relativo a las instancias que posee o bien relativo a las propiedades de la clase. Generalización y especialización expresan relaciones de inclusión entre conjuntos.

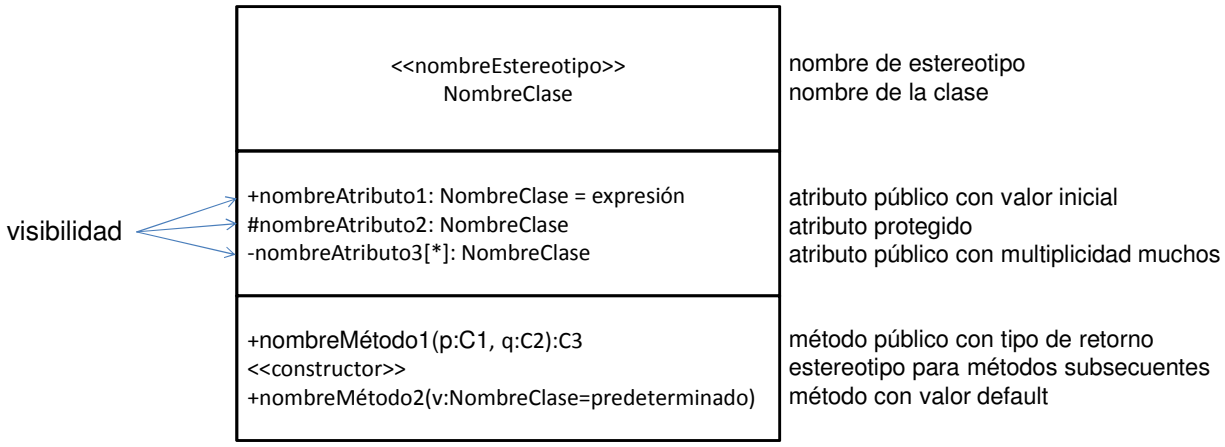


Figura 2—7 Definición de clase para un diagrama de clase.

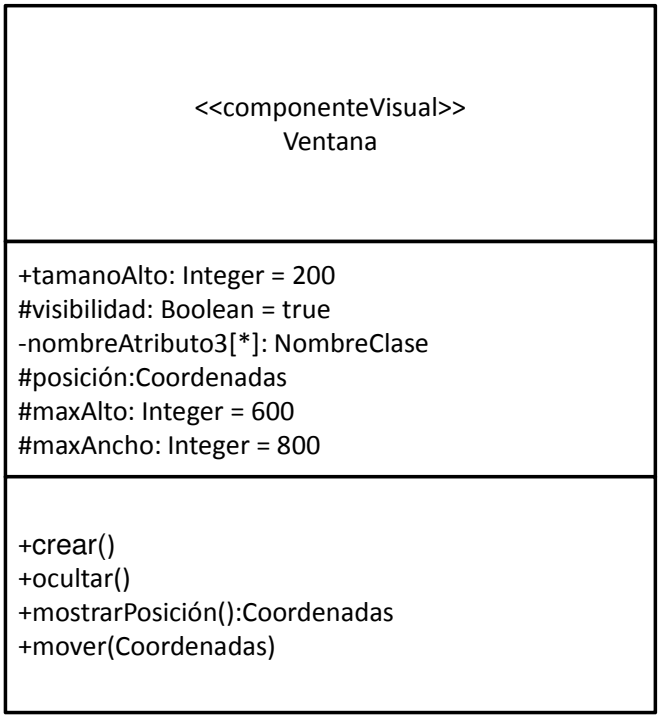


Figura 2—8 Ejemplo de clase en un diagrama de clases.

2.5 HTML.

HTML son las siglas de Hyper Text Markup Language – Lenguaje de Etiquetas de Hipertexto. Este lenguaje proporciona medios para dar formato a documentos de texto por medio de encabezados, listas, definiciones de párrafos, tablas, etc., y para incluir en esos documentos formularios interactivos, imágenes y otros elementos. Con este lenguaje se crean las páginas que son desplegadas por los navegadores web. Cuando un navegador web abre un documento HTML, lee el texto y aplica a éste el formato indicado por las etiquetas para mostrarlo finalmente como una página web.

Un archivo HTML está constituido por el texto de contenido y por etiquetas (*tags*) que dan formato al texto y que se denotan por estar entre paréntesis angulares (<>). Estas indican la manera en que debe mostrarse el texto. El texto que se ve afectado por una etiqueta HTML se encuentra entre los dos elementos principales de la etiqueta: la apertura y el cierre. Por ejemplo:

```
<HTML>
      <BODY>
            <FONT SIZE=”5”>hola, mundo!</FONT>
      </BODY>
</HTML>
```

Código 2—1 Código de una página que muestra el texto “hola, mundo!”

El código HTML mostrado define un documento que mostrará el texto “hola, mundo!” en el tamaño de letra número cinco. Se ven tres etiquetas con sus respectivos cierres en el documento:

- <HTML></HTML> Define que todo lo que está dentro de esta etiqueta es parte de un documento HTML.
- <BODY></BODY> Define que todo lo que está dentro de esta etiqueta es el cuerpo del documento, que será mostrado en pantalla por el navegador web.
- <FONT SIZE=”5”></FONT>

Elementos de una etiqueta HTML.
Los elementos de una etiqueta son: apertura, cierre, cuerpo, atributos. Aunque sólo los dos primeros son aplicables a la totalidad de las etiquetas HTML.

La apertura se indica abriendo paréntesis angular, escribiendo la etiqueta y cerrando el paréntesis. Además pueden especificarse los atributos antes de cerrar el paréntesis, como en el caso de la etiqueta <FONT> a la que pueden añadirse los atributos SIZE, COLOR y otros. Por ejemplo <FONT SIZE=”4” COLOR=”RED”>.

Entre la apertura y el cierre va el cuerpo de la etiqueta. Como en el caso de la etiqueta <HTML> que contiene como cuerpo muchos otros elementos que definen el documento HTML. O como la etiqueta <FONT> que contiene como cuerpo el texto al que va a modificar según estén definidos sus atributos.

En un documento HTML es muy frecuente que existan etiquetas que en su cuerpo contienen otras etiquetas. Las etiquetas también representan elementos y una gran cantidad de elementos HTML pueden incluir otros elementos dentro. Por ejemplo la etiqueta <TABLE> que sirve para definir un elemento tabla posee en su interior (es decir, antes de la etiqueta </TABLE>) elementos <TR> que definen filas de la tabla, y éstos, elementos <TD> que definen columnas de la fila. El siguiente código muestra cómo se define una tabla en HTML:

```
<HTML><BODY>
<TABLE>
      <TR>
            <TD>fila 1, columna 1</TD>
            <TD>fila 1, columna 2</TD>
</TR>
      <TR>
            <TD>fila 2, columna 1</TD>
            <TD>fila 2, columna 2</TD>
</TR>
</TABLE>
</BODY></HTML>
```

Código 2—2 Ejemplo de etiquetas anidadas en HTML.

El código anterior es interpretado por el navegador web para generar (*renderear*) una página que muestra una tabla como la siguiente:

fila 1, columna 1	fila 1, columna 2
fila 2, columna 1	fila 2, columna 2

Figura 2—9 Tabla de 2x2 desplegada en una página HTML.

Una etiqueta muy utilizada es la etiqueta <A> que sirve para crear textos que son ligas a otros documentos HTML u otros recursos. Para crear una liga debe especificarse el recurso al que apuntará la liga con el atributo HREF, y el texto que se mostrará en el navegador web dentro del cuerpo de la etiqueta. Por ejemplo:

```
<HTML><BODY>
      <A HREF="http://www.google.com">liga a Google</A>
</BODY></HTML>
```

Código 2—3 Uso de la etiqueta <A> para crear una liga.

Al interpretar el documento HTML el navegador web mostrará el texto "liga a Google" como un hipertexto al que puede darse clic y el navegador solicitará el recurso "http://www.google.com".

Versiones de HTML.

En sus inicios no existía un HTML estándar pero hoy en día la W3C (World Wide Web Consortium) que es un consorcio internacional para estándares para la World Wide Web ha contribuido a crear un estándar. Algunos navegadores web implementan sus propias etiquetas para posibilitar la adición de elementos vistosos a las página web que despliegan pero al desapegarse del estándar e incitar a la gente a usar sus etiquetas muchas veces provocan que una misma página se vea sumamente diferente dependiendo del navegador que la despliega. Es importante apegarse al estándar para que las páginas se desplieguen de manera muy similar en todos los navegadores.

A diciembre de 2007, HTML 4.01 y ISO/IEC 15445:2000 son las más recientes versiones de HTML. En el sitio del W3C pueden consultarse las etiquetas de dichas versiones y ejemplos de uso, así como tutoriales completos que explican el uso del lenguaje.

2.6 Conceptos básicos de un servidor web.

Un servidor web es un programa que atiende solicitudes provenientes de navegadores web (v.gr. MS IExplorer, Netscape, Opera, Safari, etc.) y les proporciona a éstos los recursos solicitados. El protocolo que es utilizado en estas transacciones es el HTTP o bien, la versión cifrada de este mismo protocolo, el HTTPS. Un navegador web envía una solicitud cuando se teclea una URL en la barra de direcciones, se da clic en una liga o se envía un formulario.

Para un servidor que ofrece contenidos estáticos (páginas que no cambian con frecuencia), básicamente, el ciclo de operaciones es el siguiente:

- Espera conexiones desde los navegadores.
- Establece una conexión con un navegador.
- Lee la petición de la conexión establecida.
- Con base en la petición, busca y lee el recurso solicitado (páginas HTML, imágenes, etc.)
- Por la misma conexión envía el recurso leído.
- Cierra la conexión y empieza de nuevo el ciclo.

El ciclo se ilustra en la Figura 2—10.

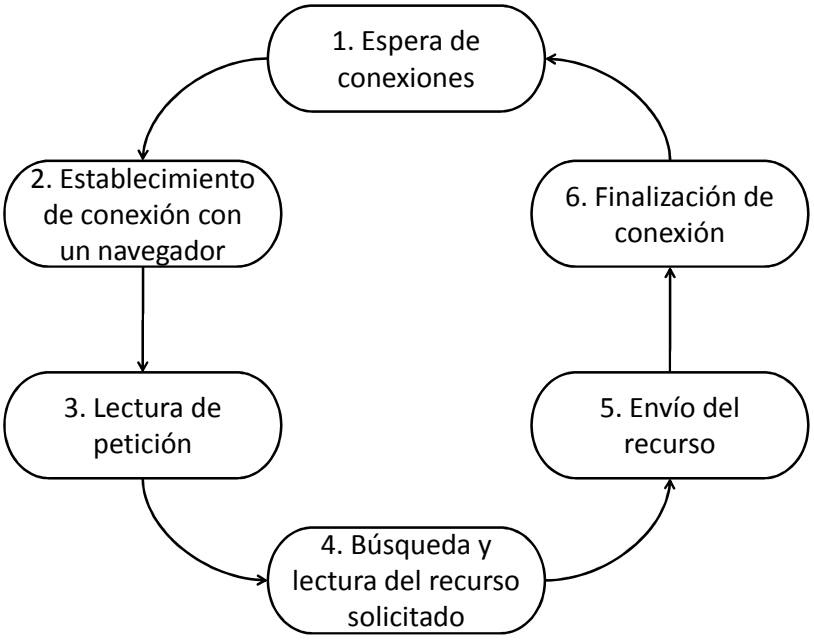


Figura 2—10 Ciclo de operaciones básicas de un servidor web que ofrece contenidos estáticos.

En general los servidores web funcionan de una manera similar a la explicada arriba porque fue así como fueron concebidos originalmente. Así podían servir recursos

estáticos, es decir, cualquier archivo que estuviera contenido en un dispositivo de almacenamiento: páginas HTML, imágenes, y en archivos en general. Más adelante fue necesario que los contenidos que despachaban los servidores fueran dinámicos, por ejemplo, una página que muestra la hora y fecha en que es desplegada, o una página que muestre datos actualizados sobre el clima, o la programación diaria de la televisión, etc. De esta manera se mantuvo el diseño original pero fueron añadidos distintos tipos de peticiones y sus respectivas maneras de despacharlos (así surgieron soluciones como CGI y los Servlets).

Servidores Web de contenido dinámico.

Ya que la mayor parte del contenido que se deseaba despachar en la web no provenía de páginas estáticas sino que era dinámico, es decir, debía ser generado al momento en que se solicitaba, los servidores web tuvieron que ser extendidos en sus capacidades de servir más que simples contenidos estáticos.

El método más antiguo que apareció para despachar contenidos dinámico fue CGI (siglas para Common Gateway Interface) que permitía pasar una solicitud HTTP a la línea de comandos del sistema operativo donde residía el servidor web. Fue una solución con una tecnología simple pero no muy portable ni muy segura. Se utilizó ampliamente por ser la única opción durante un lapso pero a la par empezaron a desarrollarse nuevas tecnologías más flexibles y confiables ya que la tendencia de despachar contenidos dinámicos no paró y hoy en día sigue al alza.

Más adelante se incorporaron a los servidores web capacidades para ejecutar código de lenguajes de programación. Esto fue un gran avance en seguridad ya que no se requería de acceso a la línea de comandos del sistema operativo sino que el mismo servidor web despachaba sus solicitudes. Y aunque algunos servidores web ofrecían soporte sólo para ciertos lenguajes, normalmente los hubo que ofrecían módulos que permitían extender el servidor para soportar más lenguajes. Estos módulos son servicios o aplicaciones contenidas por el servidor web, así que a éste se le llama “contenedor” o también, “servidor de aplicación” porque en realidad lo que maneja el servidor sólo son las peticiones HTTP pero los contenidos son elaborados por la aplicación contenida en él.

Un servidor web de contenido dinámico básicamente realiza las siguientes operaciones:

- Espera conexiones desde los navegadores.
- Establece una conexión con un navegador.
- Lee la petición de la conexión establecida.
- Establece el tipo de petición.
- Decide cómo despachar la petición y ejecuta el servicio o aplicación que generará el contenido solicitado con base al contenido de la petición.
- Por la misma conexión envía el contenido generado.
- Cierra la conexión y empieza de nuevo el ciclo.

EL ciclo se ilustra en la Figura 2—11.

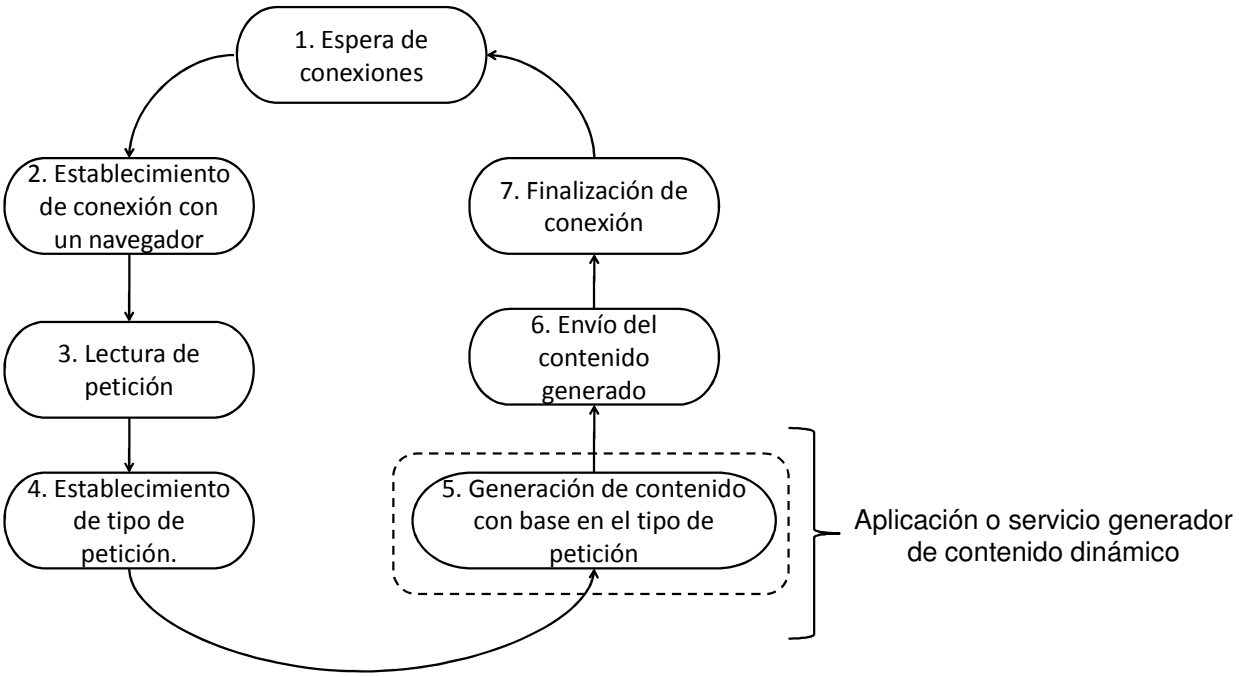


Figura 2—11 Ciclo de operaciones básicas de un servidor web de contenido dinámico.

Es importante señalar que la generación del contenido dinámico puede llevarse a cabo de distintas maneras. En el presente trabajo de tesis se explica y se utiliza la tecnología Servlet de Java.

2.7 Conceptos básicos del lenguaje Java.

En el presente trabajo de tesis se utiliza ampliamente la plataforma de desarrollo Java por lo que es de suma importancia conocer algunos de sus conceptos fundamentales y la terminología inherente. A continuación se proporciona un resumen general de los elementos que se manejarán durante el desarrollo de este trabajo y algunos otros conceptos que resulta útil tener en cuenta acerca de la plataforma Java.

Java es un lenguaje de programación desarrollando por Sun Microsystems y liberado en 1995 como el componente principal de la plataforma Java de Sun. La sintaxis del lenguaje proviene en gran medida de la de C y C++ pero tiene un modelo de objetos más simple y menos funcionalidad a bajo nivel. Las aplicaciones Java normalmente son compiladas a código byte que puede ser ejecutado en cualquier máquina virtual de Java (JVM) independientemente de la arquitectura de la computadora.

Características del lenguaje Java.

- Orientación a objetos. Java implementa algunas características de C++ con mejoras y suprime algunas otras que comprometerían la seguridad o complicarían la codificación con el lenguaje. Java manipula los datos como objetos e interfaces a esos objetos. Las plantilla de los objetos se llaman clases (igual que en C++) y las copias basadas en dichas plantillas se llaman instancias. De las características del POO utiliza principalmente la encapsulación, la herencia y el polimorfismo.

- Seguridad. La plataforma Java ofrece un lenguaje de programación de alto nivel con un gran potencial para cualquier tipo de aplicación que se desee desarrollar. Ya que es de alto nivel prescinde características que le permiten manipular direcciones de memoria directamente. Así, Java ofrece un entorno de ejecución seguro, ya que todo código Java es ejecutado dentro de la máquina virtual de Java (JVM) que es un programa que corre de manera aislada sobre un sistema operativo. Es imposible que el código ejecutado dentro de la máquina virtual acceda a localidades de memoria protegidas o realice cualquier otro tipo acceso ilegal por lo que el sistema operativo sobre el que corre la JVM no corre riesgos al ejecutar programas Java.

- Portabilidad. Ya que el código Java es ejecutado dentro de JVM y dicha máquina virtual puede ejecutarse en un gran número de sistemas operativos, Java no depende del sistema operativo sobre el que se ejecuta, y cualquier aplicación Java corre de la misma manera en cualquier sistema operativo.

- Código reutilizable. Debido a la orientación a objetos de Java, posee características como facilidad en el desarrollo, reutilización del código y mayor calidad del código.

Algunos de los conceptos del lenguaje Java que se utilizan en el presente trabajo se describen brevemente a continuación.

Clases.
Todo código Java forma parte de una clase, es una clase o describe como funciona una clase. El entendimiento de las clases es fundamental para hacer uso del lenguaje Java e interpretar código de programas Java.

Una clase es la descripción estática de un conjunto de atributos y de funciones que operan sobre ellos. Estas descripciones modelan conjuntos de objetos reales que poseen características en común. Por ejemplo, puede crearse la clase Coche que describe que un coche posee cuatro ruedas, volante, color, etc. La base de cualquier aplicación Java son las definiciones de las clases que lo constituyen.

A partir de una clase, que es como una plantilla, es posible crear objetos que son instancias dinámicas de esa clase y son colocadas en memoria con valores concretos en sus atributos. Por ejemplo, el atributo color de la clase Coche, en una instancia de dicha clase debe tener un valor concreto como “rojo” o “azul”.

Todas las acciones de un programa en Java se colocan dentro clases. Java no soporta funciones o variables globales, es decir, fuera del bloque de la clase. La única sentencia que se admite fuera del bloque de clase es **import**, que el compilador remplaza con el código que contiene la clase especificada a continuación de dicha sentencia.

En lenguaje C la unidad fundamental son los archivos con código fuente, en Java, como se ha dicho, son las clases, y cada definición de clase debe estar contenida en su propio archivo. Por ejemplo, la definición de la clase Coche debe estar contenida en un archivo de código fuente llamado Coche.java.

Paquetes.
Un paquete es un conjunto de clases Java. Los nombres de los paquetes son palabras en minúsculas separadas por puntos que indican los directorios en que están guardados los archivos de las clases.

Para que una clase pertenezca a un paquete, debe contener al principio de su código la sentencia package seguida del nombre del paquete al que pertenece. Por ejemplo:

```
package siidcb;

public class SiidcbUtil {

      public static String trimTo(final String target, int maxSize)
      {
                  return (target.length() > maxSize ? target.substring(0,
                  maxSize) : target);
      }
}
```

Código 2—4 Inclusión de una clase en un paquete.

En este caso la clase SiidcbUtil forma parte del paquete “siidcb”. Esto indica también que el archivo que contiene el código de la clase SiidcbUtil está dentro de un directorio llamado “siidcb”. Si el paquete “siidcb” a su vez perteneciera a otro paquete llamado “fi”, el nombre de este último se antepondría al de “siidcb” seguido de un punto. Es decir que el nombre completo del paquete “siidcb” sería “fi.siidcb”, y el nombre completo de la clase “SiidcbUtil” sería “fi.siidcb.SiidcbUtil”.

Identificadores.
Los identificadores son los nombres que se asigna a variables, métodos, clases y objetos; cualquier cosa que el programador necesite identificar o usar.

En Java, todo identificador debe comenzar con una letra, un guión bajo o un símbolo de pesos; los siguientes caracteres pueden ser letras o dígitos. Se distinguen las mayúsculas de las minúsculas y no hay longitud máxima.

Existen palabras clave y palabras reservadas que no pueden usarse como identificadores ya que tiene un significado especial para la gramática del lenguaje. Estas palabras son:

abstract	do	if	package	synchronized
boolean	double	implements	private	this
break	else	import	protected	throw
byte	extends	instanceof	public	throws
case	false	int	return	transient
catch	final	interface	short	true
char	finally	long	static	try
class	float	native	strictfp	void
const	for	new	super	volatile
continue	goto	null	switch	while
default	assert			

Herencia.

La Herencia es el mecanismo por el que se crean nuevas clases a partir de clases existentes. Por ejemplo, si se tiene la clase Persona, se puede crear la subclase Trabajador, que es una especialización de Persona:

```
class Trabajador extends Persona {
      int numero_de_trabajador;
      }
```

Código 2—5 Ejemplo de extensión de la clase Persona.

La palabra clave **extends** se usa para especificar que la clase Trabajador es una subclase (especialización) de la clase Persona. Cualquier cosa que contenga la definición de Persona (atributos y métodos) será copiada a la clase Trabajador. Además, en Trabajador se pueden definir métodos y atributos propios sólo de ella. Se dice que Trabajador deriva o hereda de Persona porque la clase Trabajador cuenta con todos los atributos y métodos que Persona más los suyos propios. También suele decirse que Trabajador extiende a Persona.

2.8 Java Servlet.

Servlet es una API (Application Programming Interface - Interfaz de Programación de Aplicaciones) que permite a los desarrolladores de software extender la funcionalidad de un servidor web de manera sencilla empleando la plataforma Java para añadir contenidos dinámicos a dicho servidor. El servidor web en este caso es llamado contenedor o servidor de aplicación y existen varios servidores web tanto comerciales como no comerciales contenedores de Servlets.

Un Servlet es la implementación de la Java Servlet API y es un objeto de Java que maneja peticiones HTTP provenientes de navegadores web (cliente) y genera respuestas basadas en esas peticiones: el Servlet recibe las peticiones, las procesa y finalmente envía los resultados a una página dinámica usualmente llamada *template* (plantilla) que es completada con esos resultados para luego ser enviada como respuesta al cliente. Los contenidos dinámicos casi siempre son generados como páginas HTML, pero pueden crearse también en otros tipos de formatos como XML.

Un Servlet es un conjunto de programas controladores que deciden qué lógica debe seguirse para proporcionar una respuesta al cliente que ha hecho una solicitud. Por ejemplo, si un usuario, por medio de su navegador web, envía al Servlet datos de acceso como su nombre de usuario y su contraseña, los controladores se encargan de validarlos contra un almacén de datos (como una base de datos) y de generar una respuesta que otro componente del Servlet enviará al cliente.

Por lo general una aplicación necesita llevar un seguimiento de las solicitudes que hace un mismo usuario (o sea, solicitudes desde un mismo navegador) durante un periodo de tiempo. Un ejemplo es cuando se desea controlar el acceso a ciertos recursos sólo a ciertos usuarios. Entonces un usuario debe enviar datos de autenticación y una vez que son validados, sus posteriores solicitudes se procesarán con base en dicha información de autenticación. Los Servlets pueden hacer este seguimiento de las solicitudes que hace un mismo navegador empleando *cookies* (pequeños archivos que se almacenan temporalmente en el sistema del cliente), variables de sesión (almacenadas en memoria del servidor web) o por medio de información añadida a la URL. Este seguimiento debe ser implementado en los controladores del Servlet.

Algunos servidores web contenedores de Servlets son:
No comerciales:

- Apache Tomcat (antes Jakarta Tomcat).
- Apache Geronimo (implementación totalmente hecha en J2EE).
- Jetty.
- Jaminid.
- Enhydra.
- Winstone.

- tjws spec 2.4.

comerciales:

- BEA WebLogic Server o Weblogic Express de BEA Systems.
- Borland Enterprise Server.
- GlassFish (*open-source*).
- Java System Application Server de Sun Microsystems.
- Java System Web Server de Sun Microsystems.
- JBoss (*open-source*).
- JRun de Adobe Systems (antes desarrollado por Macromedia).
- LiteWebServer (*open-source*).
- Oracle Application Server de Oracle Corporation.
- Orion Application Server de IronFlare.
- Resin Server de Caucho.
- ServletExec de New Atlanta Communications.
- WebObjects de Apple Inc.
- WebSphere de IBM.

Un ejemplo de implementación de la interfaz Java Servlet:

Una vez configurado en un servidor web, el siguiente Servlet genera y envía a un cliente una página web con la frase “Hola, mundo!”.

```
// Hola.java
import java.io.*;
import javax.servlet.*;

public class Hola extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        final PrintWriter pw = response.getWriter();
        pw.println(
            "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
            "Transitional//EN\">\n" +
            "<HTML>\n" +
            "<HEAD><TITLE>Hola mundo!</TITLE></HEAD>\n" +
            "<BODY>\n" +
            "<H1>Hola mundo!</H1>\n" +
            "</BODY></HTML>");
        pw.close();
    }
}
```

Código 2—6 Servlet que genera una página HTML con el texto "Hola mundo!".

Es importante remarcar que este sencillo Servlet de ejemplo sólo produce una página HTML que despliega el texto “Hola, mundo!”. Se puede ver que gran parte del código son etiquetas HTML (como podrían ser: <BODY>, <H1>, <font>, <B>, etc.) que se imprimen con el comando println("..."). La inclusión de código del formato HMTL para despliegue es una de las desventajas de utilizar la tecnología Servlet sin algún otro componente. Sin embargo hoy en día los Servlets son empleados en conjunto con la tecnología JSP y el patrón MVC para aplicaciones web; esto facilita mucho la generación de páginas HTML a partir de Servlets para crear el contenido y plantillas elaboradas con JSP para dar formato a dichos contenidos.

En el desarrollo del presente trabajo se utiliza Apache Tomcat como contenedor (servidor de aplicación) de Servlets.

2.9 Java Server Pages.

JSP es una tecnología Java que permite a los desarrolladores de software crear dinámicamente contenidos como páginas HTML, XML, etc. y enviarlos a un navegador web conforme éste lo solicite. Esta tecnología permite mezclar código HTML estático con código Java que genera contenidos generados dinámicamente. Así se reutiliza la parte estática del código HTML y con código Java se incluyen los contenidos que cambian frecuentemente antes de enviar la página al cliente (navegador web).

Las JSP fueron el paso siguiente de los Java Servlets ya que la codificación de un Servlet, incluso para generar una página sencilla, requería de una gran cantidad de trabajo. Puede decirse que antes de JSP el Servlet era principalmente código Java con fragmentos de código HTML y con JSP esta situación se invirtió: las JSP's son código HTML con fragmentos de código Java (llamados scriptlets) incrustados.

Ejemplo de una página JSP:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>Bienvenido a nuestra tienda</TITLE></HEAD>
<BODY>
<H1>Bienvenido a nuestra tienda</H1>
<SMALL>Hola,
<!--El nombre será "Nuevo Usuario" para los nuevos visitantes -->
<% out.println(Utils.getUserNameFromCookie(request)); %>
Para acceder a sus datos de cuenta, haga clic
<A HREF="Account-Settings.html">aquí.</A></SMALL>
<P>
HTML común para el resto de la página.
</BODY></HTML>
```

Código 2—7 Una página JSP simple con un scriplet.

Los elementos que se observan en este código son los siguientes:

Código descriptor del archivo:	<!DOCTYPE HTML (...)>
Código HTML:	entre las etiquetas <HTML>, <H1>, etc.
Comentarios:	<!—El nombre será (...) -->
Código Java:	<% out.println((...)); %>

Cuando la página es solicitada por primera vez, un Servlet principal se encarga de compilarla y generar un nuevo Servlet con la funcionalidad de dicha página. Una vez que es compilada se completa con el contenido dinámico (en este caso el resultado de la función out.println() que devuelve una cadena de caracteres) y se envía al cliente (navegador web). De modo que lo que se vería en el navegador es:

Figura 2—12 Página JSP mostrada en el navegador.

Algunas ventajas de las JSP son:

- Al ser provenir de Servlets y por tanto de Java, son multiplataforma.
- Son mucho más sencillas de escribir que un Servlet con muchas sentencias println() para generar código HTML.
- Cuentan con cierta separación entre el formato con el que es mostrado el contenido y el contenido mismo.

Sin embargo, una desventaja con que cuentan es que no separan por completo la lógica de negocio (el funcionamiento de una aplicación web) de la de presentación (generación de páginas que se envían al cliente). Es decir, el hecho de que pueda incluirse código Java en una página HTML resulta adecuado para sitios web pequeños con pocas páginas pero no para grandes proyectos que llegan a contar con cientos de páginas y lógica de presentación compleja.

Afortunadamente se han creado algunos marcos de trabajo que ayudan a separar claramente la lógica de negocio y la de presentación. Un ejemplo son las bibliotecas de etiquetas (*taglibs*) que permiten prescindir del uso de código Java en las JSP sustituyéndolo con etiquetas al estilo de HTML. Y también existen maneras de crear clases y funciones que pasen datos a páginas JSP para que éstas los desplieguen siguiendo el formato en ellas declarado y haciendo uso de etiquetas estilo HTML.

Capítulo 3. Análisis.

3.1 Definición y contenidos del sistema.

El sistema propuesto se llama SIIDCB que son las siglas para “Sistema Integral de Información de la División de Ciencias Básicas”. Es un sistema computacional accesible remotamente con un navegador web por medio de la red de la DCB.

Los contenidos del sistema son información del personal, de la organización de la DCB y de la programación semestral de actividades académicas. Almacena dichos contenidos en una base de datos segura y proporciona una interfaz gráfica para manipular los contenidos y permite el acceso sólo a usuarios que poseen una cuenta.

Los criterios a tomar en cuenta durante el desarrollo del sistema son la facilidad de uso y de mantenimiento, la extensibilidad y escalabilidad y el orden en la codificación.

Con el fin de facilitar el mantenimiento del sistema, éste se conforma por varios módulos o componentes principales responsables cada uno de diferentes funcionalidades requeridas. Los módulos son:

Módulo de manejo de personal.
Desde este módulo es posible dar de alta en la base de datos del sistema personal nuevo, así como modificar o eliminar sus registros. Cada registro guarda información personal y académica del sujeto. También con este módulo es posible hacer búsquedas de personal en la base de datos.

Módulo de manejo de entidades.
Este módulo maneja información acerca de la organización de entidades dentro de la División de Ciencias Básicas. Permite crear una estructura jerárquica como la que tiene la División. Cada nodo de la estructura representa una entidad que puede ser de diferente tipo (división, secretaría, coordinación, departamento, etc.) y puede asignarse nombre y jefe a cada una, guardar información sobre su ubicación y ver un listado del personal que pertenece a ella.

Módulo de currículums.
Cada miembro del personal registrado en la base de datos puede hacer uso de este módulo para ingresar su información de experiencia académica y profesional y puede crear un reporte para imprimirlo. Los administradores del sistema tienen acceso a todos los currículums y también a versiones imprimibles de éstos.

Módulo de programación académica semestral.
Por medio de este módulo los administradores pueden crear registros de los grupos de las asignaturas del semestre por comenzar y asignarles horarios y otros detalles como

cupo, bloque, sección, etc.. Los funcionarios pueden ver los grupos existentes de las materias a su cargo y asignar profesores a cada uno de dichos grupos.

Este módulo también maneja otras actividades académicas como talleres de ejercicios y asesorías. Los funcionarios crean estos registros y pueden asignar nombre y horario a cada actividad.

Otra parte de la programación académica que maneja este módulo son las actividades adicionales tales como ayudantías, licencias, comisiones, actividades en otras divisiones o dependencias, etc. En esta parte también los funcionarios crean los registros y les asignan tipo de actividad, nombre, horario, etc.

Módulo de administración de cuentas.
Ya que el sistema tendrá una interfaz web, este módulo facilita a los administradores modificar las cuentas de los usuarios existentes. Se puede definir el tipo de cuenta (administrador, funcionario o profesor), activarla o desactivarla y/o definir una fecha de expiración. Los administradores pueden cambiar la contraseña de cualquier cuenta desde este módulo en caso de que un usuario la olvide.

Módulo de avisos.
Con el fin de proporcionar información como noticias y avisos a los usuarios del sistema, este módulo permite a los administradores guardar textos y definir a qué tipo de usuarios éstos serán desplegados. También pueden definirse textos que se desplegarán en la página de inicio del sistema.

Módulo de catálogos.
Desde este módulo los administradores pueden modificar ciertas tablas de la base de datos que sirven como catálogos de opciones para una gran cantidad de campos en los formularios que son mostrados a los usuarios del sistema. Se puede crear, modificar o eliminar opciones de esos catálogos y los cambios se ven reflejados inmediatamente en los formularios que se despliegan una vez que algún cambio ha sido hecho.

Cada uno de los módulos que constituyen SIIDCB se presenta al usuario por medio de páginas web con menús que muestran las posibles acciones que pueden llevarse a cabo. También existe un menú principal siempre visible que despliegue los módulos accesibles por el usuario.

3.2 Casos de uso.

Diagramas de casos de uso.

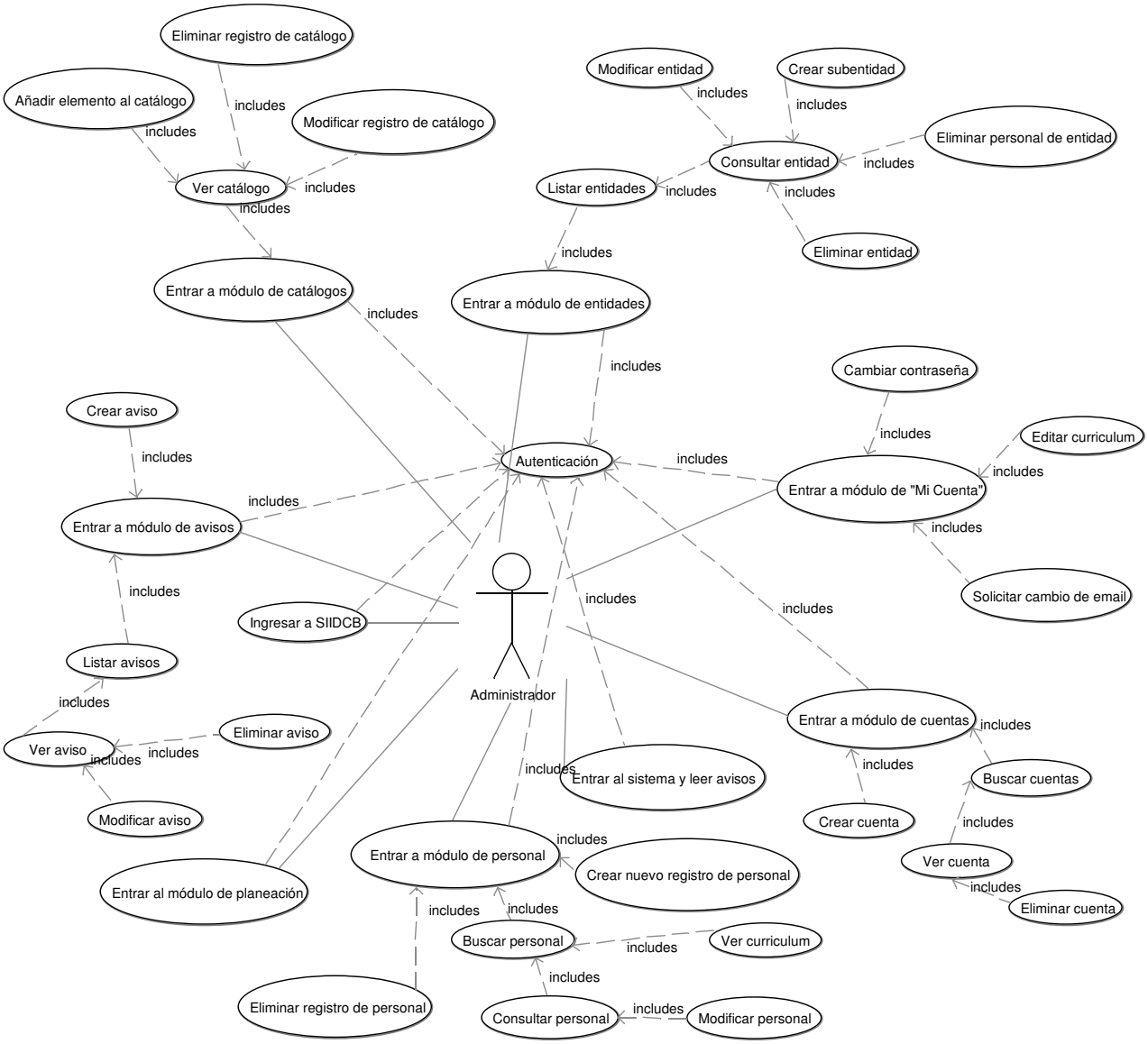


Figura 3—1 Diagrama de casos de uso para el actor "Administrador".

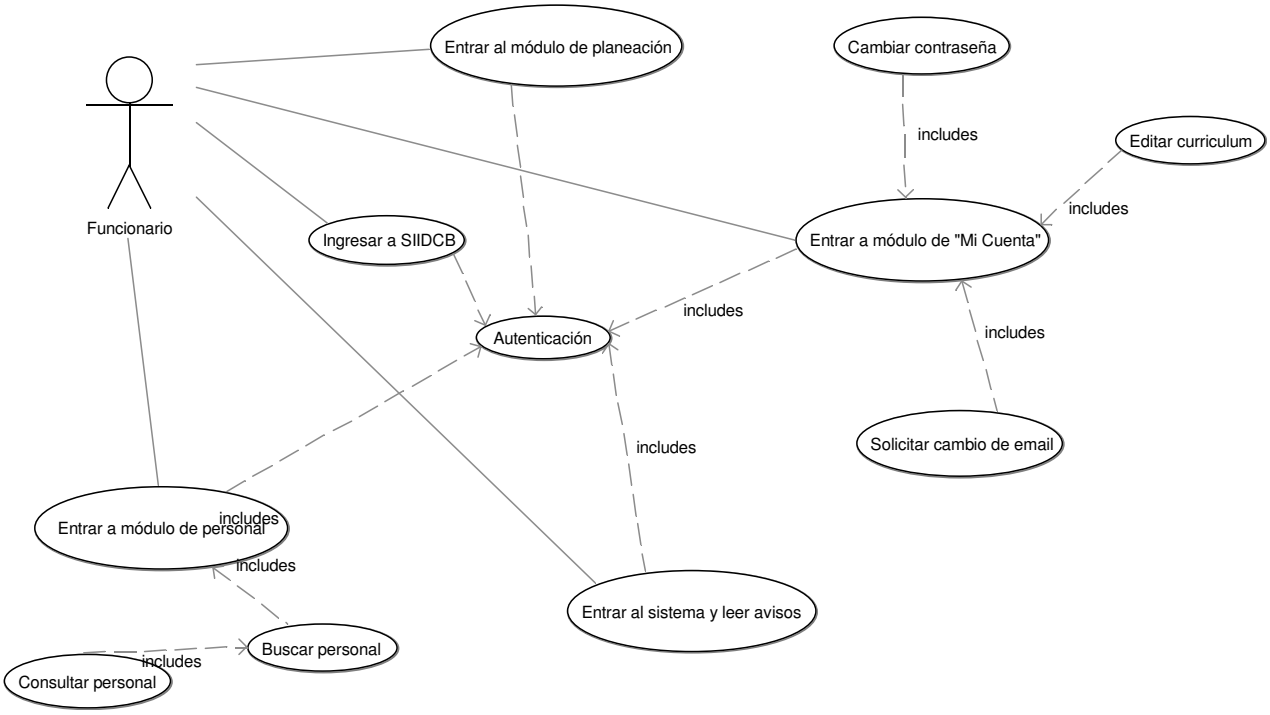


Figura 3—2 Diagrama de casos de uso para el actor "Funcionario".

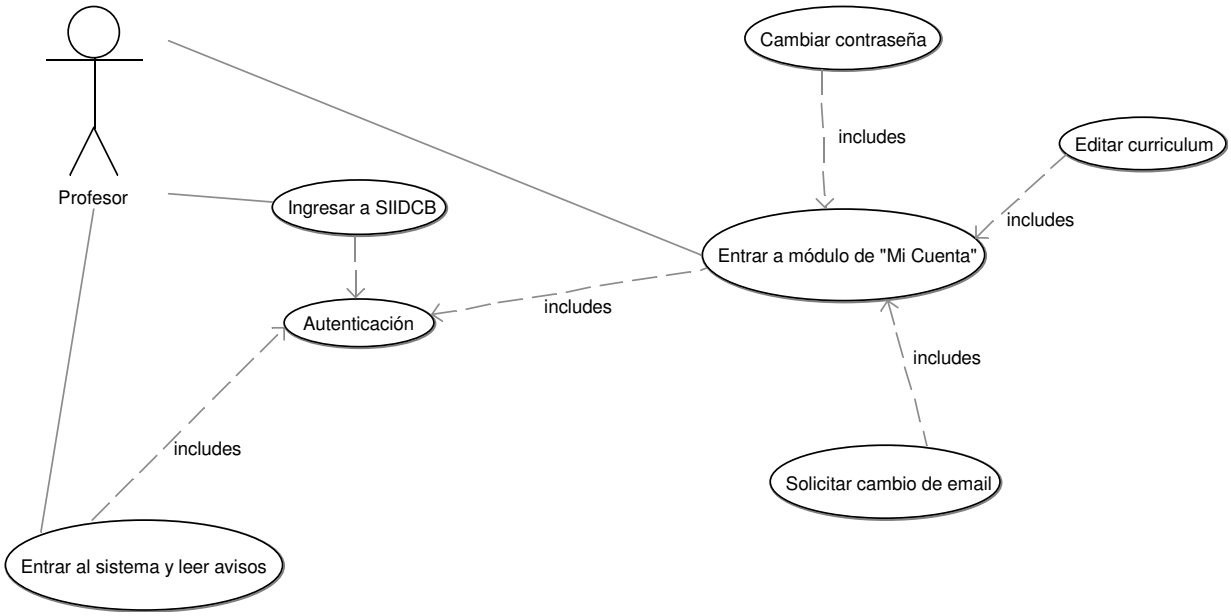


Figura 3—3 Diagrama de casos de uso para el actor "Profesor".

Definición de los casos de uso.

A continuación se muestran las definiciones de tres casos de uso mostrados en el diagrama de casos de usos para el usuario “Administrador”. Las definiciones completas de los casos de uso mostrados en los diagramas de este capítulo se incluyen como un anexo al final de presente trabajo.

Nombre:	Autenticación.
Descripción: Proceso que verifica si la sesión de un usuario es válida y autoriza el acceso a un recurso solicitado.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Ninguna.	
Flujo Normal: 1. El actor solicita un recurso del sistema. 2. El sistema verifica si su sesión es válida (ha ingresado nombre de usuario y contraseña) y no ha expirado. 3. El sistema envía al actor el recurso solicitado.	
Flujo Alternativo: Si el actor no ha ingresado al sistema nombre de usuario y contraseña válidos, se muestra un aviso de que es necesario hacerlo para acceder al recurso solicitado. También se muestra una liga a la página de ingreso al sistema.	

Nombre:	Entrar a módulo de catálogos.
Descripción: Permite ver los catálogos del sistema que pueden ser editados.	
Actores: Administrador.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita el módulo de catálogos.	

2. El sistema devuelve como respuesta un menú con las opciones del módulo.
Flujo Alternativo:

Nombre:	Ver catálogo.
Descripción: Permite ver un catálogo del sistema: nombre de los campos y valores.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de catálogos.	
Flujo Normal: 1. Desde el menú de catálogos, e actor da clic en la liga hacia un catálogo. 2. El sistema muestra el catálogo.	
Flujo Alternativo:	

Capítulo 4. Diseño.

4.1 Arquitectura del sistema.

Patrón de diseño utilizado: MVC (Model-View-Controller) para aplicaciones web.

Un sistema web debe ser fácil de mantener así que la separación de componentes resulta muy conveniente en su desarrollo. La arquitectura MVC para una aplicación web consigue esta separación en tres capas:

- **Modelo.** Es la representación de la información con la cual el sistema opera. Generalmente se trata de un sistema manejador de bases de datos y clases auxiliares para accederlo.

- **Vista.** Es la capa de presentación. Se encarga de entregar al usuario un listado de las acciones disponibles y la información necesaria para utilizarlas.

- **Controlador.** Es el puente entre el Modelo y la Vista. Interpreta las solicitudes del cliente y decide las modificaciones o manipulaciones sobre el modelo así como las vistas que se entregan al final de una solicitud.

Para entender un poco más este patrón y su utilidad en el desarrollo de una aplicación web, a continuación se explica un patrón usado anteriormente por los diseñadores web y los problemas que derivaron en la evolución al MVC.

Un antecesor del MVC en web: el Modelo 1.

Este patrón es la forma más sencilla de desarrollar aplicaciones web con presentación o vistas con páginas JSP. En el Modelo 1 el cliente (navegador web) accede directamente a las páginas JSP; es decir, las solicitudes del cliente son directamente manipuladas por la JSP.

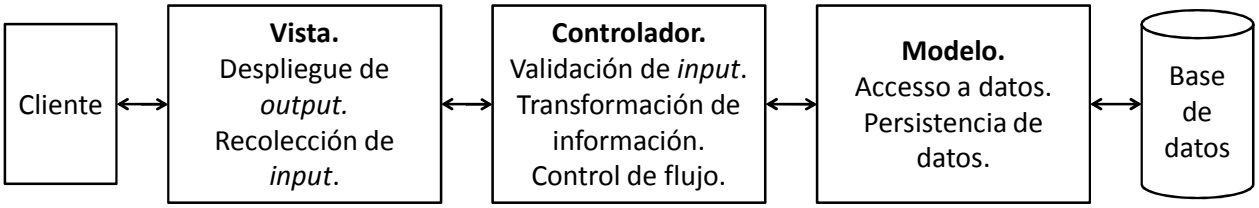


Figura 4—1 El patrón "Modelo 1".

Un ejemplo del Modelo 1 es una página web con un hipervínculo a una JSP. Cuando un usuario da clic al hipervínculo, en el servidor se manda llamar la página JSP, cómo se

muestra en la figura siguiente. El servidor de aplicación lee la JSP, la compila si es necesario (sólo se compila si es la primera vez que es accedida) y ejecuta el Servlet generado. La página JSP contiene código Java incrustado (scriptlets) y etiquetas para acceder los beans de Java de modelo que contienen la lógica para la conexión al modelo (base de datos) y extraer datos. Una vez que los beans contienen la información requerida, la página JSP es generada (*rendereada)* rellenándose con los datos de los beans, y enviada como respuesta al cliente para su despliegue.

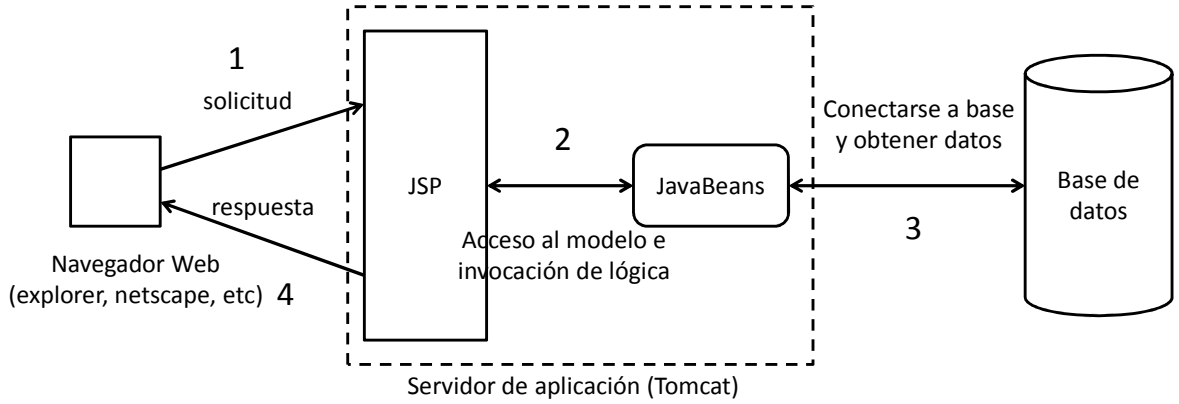


Figura 4—2 El patrón "El Modelo 1" implementando una aplicación web.

Problemas con la arquitectura Modelo 1.

El modelo 1 es sencillo y existe cierta separación entre el contenido (beans de Java de modelo) y la presentación (JSP), y esta separación puede ser suficiente para aplicaciones pequeñas. Las aplicaciones más grandes suelen tener mucha lógica de presentación. En el Modelo 1 la lógica de presentación usualmente implica una gran cantidad de código incrustado en las JSP en forma de scriptlets y esto vuelve difícil interpretar el código que se encuentra mezclado con etiquetas HTML. También vuelve las tareas de mantenimiento sumamente pesadas. En aplicaciones grandes las páginas de presentación son mantenidas por diseñadores y esta mezcla de fragmentos de código Java y etiquetas HTML impide una distribución clara del trabajo.

El control de la aplicación se encuentra descentralizado en el Modelo 1 ya que la siguiente página que se desplegará está determinada por la lógica en la página actual. Este tipo de navegación descentralizada suele provocar grandes dificultades conforme se desarrolla la aplicación y se hace más grande. Estos inconvenientes se resuelven con un patrón diferente:

El Modelo 2 o MVC.

El Modelo 2 para diseñar páginas JSP es en realidad la arquitectura MVC implementada a aplicaciones web. Por lo tanto Modelo 2 y MVC son términos intercambiables en el argot del desarrollo web. La diferencia principal entre los patrones Modelo 1 y Modelo 2 es que en el último un controlador manipula la petición del cliente

en lugar de que lo haga una página JSP. El controlador es implementado como un Servlet. Los pasos seguidos cuando un cliente envía una solicitud son:

1. El Servlet controlador recibe la solicitud (esto quiere decir que una liga en una página debe apuntar al controlador y no a una página JSP directamente).

2. El controlador entonces crea una instancia de los beans apropiados basándose en los parámetros de la solicitud (y a veces en atributos de la sesión).

3. El Servlet controlador se comunica con la base de datos para extraer la información solicitada.

4. El controlador pone los resultados o datos requeridos en beans de presentación en el entorno necesario (request, session o application).

5. El controlador pasa la solicitud a la siguiente vista (página) con base en la URL de la solicitud.

6. La capa de vistas o despliegue se encarga de *renderear* (generar) la página que se entrega al cliente.

Es importante notar que no existe código de lógica de presentación en la página JSP: las páginas JSP no son donde se decide cuál será la próxima página por desplegar. Esto es determinado por el controlador.

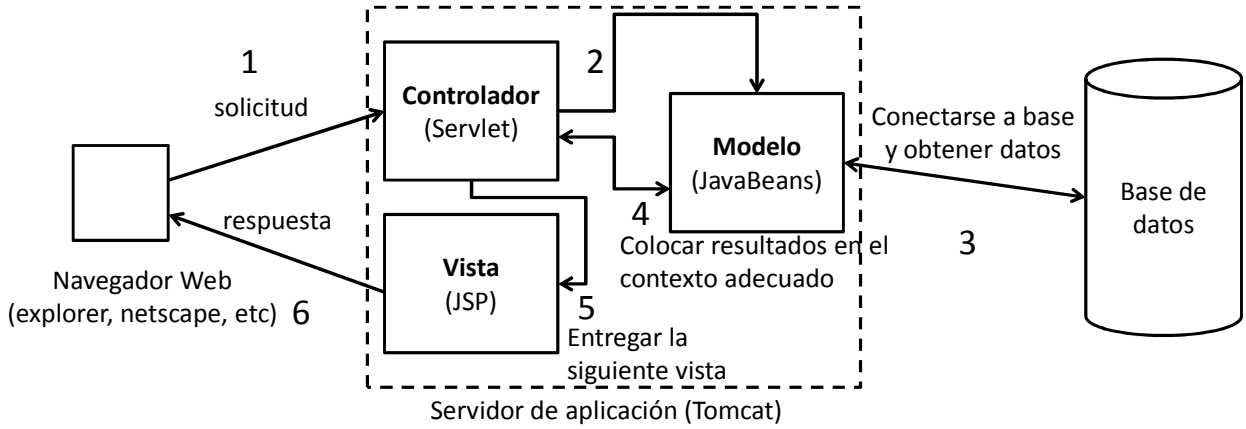


Figura 4—3 El patrón "Modelo 2" o MVC.

4.2 El Marco de trabajo para aplicaciones web Apache Struts 1.3.

Struts es un marco de trabajo *open-source* distribuido con licencia Apache para desarrollo de aplicaciones web sobre la Plataforma Java EE y está diseñado para seguir el patrón de diseño MVC. Hace uso y extiende la **API** Java Servlet y va dirigido a que los desarrolladores utilicen el patrón de diseño MVC que separa las capas de presentación, modelo y control, por lo cual permite una buena organización durante la creación de aplicaciones.

En una aplicación web estándar, el cliente (un navegador web) generalmente envía información al servidor por medio de formularios HTML. Esta información es entonces pasada a un Servlet que la procesa, interactúa con una base de datos y produce una página HTML que envía como respuesta; o bien, la información es pasada a una página JSP que tiene código Java incrustado junto con código HTML para obtener el mismo resultado. Estas dos metodologías resultan inadecuadas para proyectos medianos o grandes porque mezclan lógica de aplicación con lógica de presentación y por eso muestran una gran dificultad al realizar mantenimiento.

El marco Struts actúa como una fachada para aplicaciones Java, proporcionando una clase controladora (un Servlet conocido como ActionServlet) y facilitando la creación de plantillas para la presentación en web (generalmente Java Server Pages). El programador es responsable de programar dicho controlador y crear el archivo de configuración para el mismo, el cual une éste con el modelo y con las plantillas de presentación.

Struts proporciona los siguientes componentes para desarrollo:

- Un Servlet configurable (ActionServlet) que actúa como un controlador principal.
- Clases base que son extendidas para implementar la lógica de la aplicación web: Action y ActionForm.
- Un rico conjunto de etiquetas personalizadas JSP que cooperan con el controlador para su uso en componente de vistas (páginas JSP).
- Un marco de validación de datos de entrada desde formularios HTML. También puede implementarse la validación manualmente dentro de las clases que extienden ActionForm.
- Mecanismos para el manejo y reporte de errores.
- Soporte para la internacionalización (i18n) a través de archivos de recursos y Java Locales.

Se decidió trabajar con este marco debido a que es uno de los más utilizados para el desarrollo de aplicaciones web y es un marco estable y maduro. El proyecto comenzó en 2000 y las últimas versiones fueron liberadas en marzo (1.3.8) y julio (2.0.9) de 2007. Existe en internet una gran cantidad de documentación así como foros de discusión acerca de Struts. Y también existen muchas herramientas IDE para el desarrollo de aplicaciones web con Struts (v.gr. Eclipse, también gratuito).

4.3 Diseño general del sistema.

El sistema está desarrollado en el lenguaje Java y por lo tanto está integrado por varias clases y paquetes de clases. Ya que el sistema se llama SIIDCB, siglas de Sistema Integral de Información de la División de Ciencias Básicas, el paquete principal de la aplicación se llama “siidcb”.

SIIDCB se conforma por servlets que proporcionan tanto control para la lógica de negocio como para el despliegue de las vistas. El sistema está contenido en un servidor web así que debe ser accedido por medio de un navegador web. Dentro de las clases controladoras se realizan las consultas a la base de datos. En la Figura 4—4 se observa el funcionamiento general del sistema.

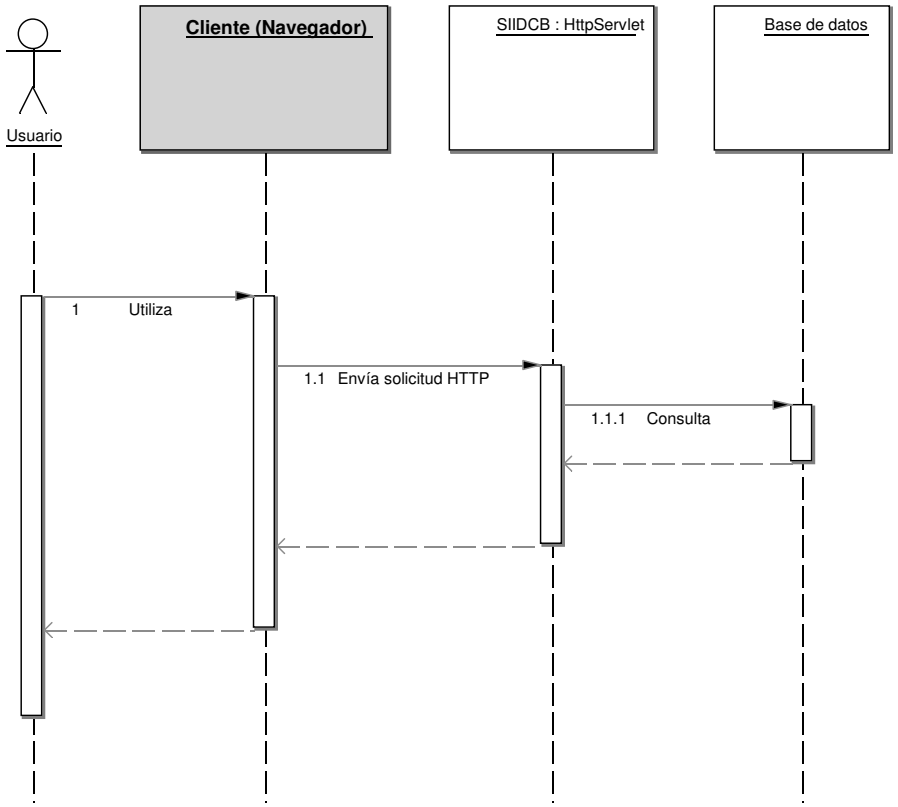


Figura 4—4 Diagrama de secuencia que muestra la operación general del sistema.

SIIDCB hace uso del marco de trabajo Apache Struts, así que un gran número de clases del sistema implementan o extienden clases o interfaces de Struts. Por lo general la clase org.apache.struts.action.Action las clases controladoras y org.apache.struts.action.ActionForm las clases para los formularios que se presentan al usuario en pantalla.

El sistema cuenta con una arquitectura que sigue el patrón MVC. Éste separa los componentes de control, despliegue y modelo. Cada uno de estos tres componentes está integrado por un conjunto de clases que desempeñan tareas respectivas. El control lo desempeñan las clases en el paquete “siidcb.struts.action”. Dentro de éste existen paquetes por módulos de funcionalidades. Así, las clases que integran el control en el módulo de personal están en el paquete “siidcb.struts.action.personal”.

4.4 Diagramas de clase.

Debido a que la aplicación SIIDCB cuenta con varios paquetes con un gran número de clases cada uno, en este apartado sólo se muestra el paquete principal de la aplicación donde se observan sus subpaquetes. Todos los diagramas de clase han sido anexados al final del presente documento.

Paquete siidcb.

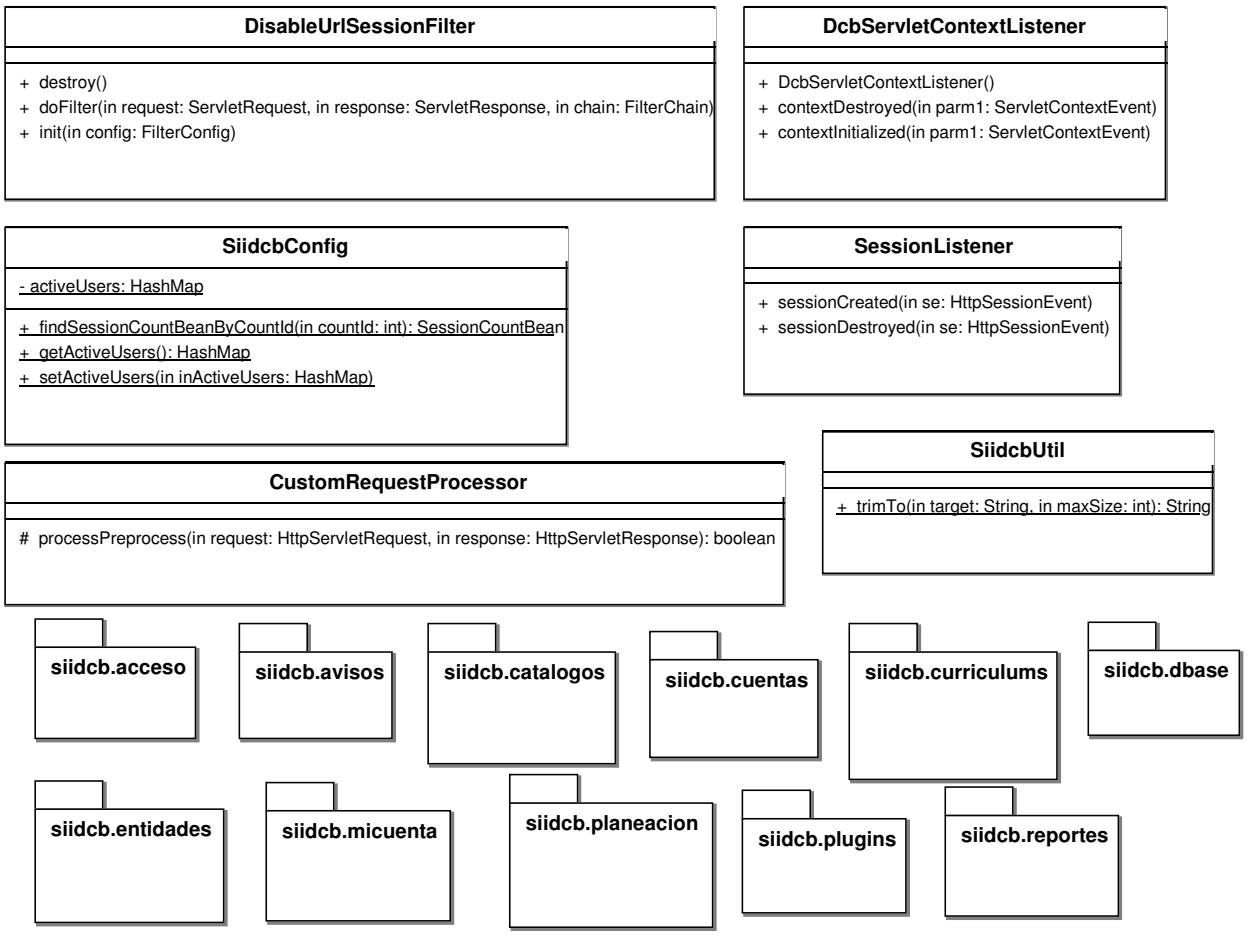


Figura 4—5 Diagrama de clases del paquete principal de la aplicación.

Capítulo 5. Desarrollo.

5.1 Selección de tecnologías y entorno de trabajo.

5.1.1 Manejador de base de datos: MySQL 5.1.

MySQL es un sistema manejador de bases de datos desarrollado en ANSI C y con licencia GNU GPL (Licencia Pública General, creada por la Free Software Foundation orientada a proteger la libre distribución, modificación y uso de software). Se trata de un manejador multiplataforma, de modo que éste puede ser montado en diferentes servidores sin cambios importantes en el resto de la aplicación.

Se ha seleccionado este manejador principalmente por ser software libre, por ser multiplataforma y porque posee la propiedad ACID para realizar transacciones seguras. **ACID** es un acrónimo de **A**tomicity, **C**onsistency, **I**solation and **D**urability: Atomicidad, Consistencia, Aislamiento y Durabilidad.

Atomicidad: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan únicamente aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información nunca generará ningún tipo de error.

Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Seguridad: un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Gran escalabilidad: soporte a grandes bases de datos. Puede manejar bases de datos que contienen miles de tablas y millones de registros. Estas cantidades dependen del tipo de información que contiene la tabla. Como referencia, la cantidad de información máxima por defecto es de 64TB y ésta es extensible de manera ilimitada.

Migración de Access a MySQL.

La decisión de migrar a MySQL se tomó con base en que MySQL es un manejador de bases de datos mucho más rápido y más estable que Microsoft Access. MySQL es software libre (licencia GPL), es gratis. Microsoft Access funciona óptimamente en un entorno local (base de datos personal) de aplicación de oficina. En cambio MySQL fue diseñado para funcionar muy eficientemente en un entorno multiusuario de cualquier tamaño.

Existen diversas interfaces gráficas (GUI) gratis para consultar bases de datos en MySQL directamente si es necesario, y también pueden ser consultadas por medio de Access con ODBC (estándar de Microsoft para acceder a una base de datos) si se quiere dar mantenimiento de esta manera.

Access no es usado frecuentemente para bases de datos de muchos megabytes y es muy probable que la aplicación propuesta deberá manejar información de muchos megabytes y si más adelante esa misma base de datos se usa en otra aplicación, es decir, sea consultada por la aplicación propuesta y por alguna o algunas más, MySQL soportará la carga extra con mucha eficiencia. En Access la escalabilidad es sumamente limitada.

Realizar respaldos de la base de datos de MySQL es sencillo y existen herramientas gratuitas (como MySQL Administrator) que permiten realizarlas periódicamente de manera automática. En el sistema actual, al estar la base de datos distribuida en varios archivos y cada uno en diferentes carpetas con diferentes permisos de acceso, el respaldo es más complicado. El mantenimiento a la base de datos también resulta complicado y debe llevarlo a cabo una persona con acceso a todas las carpetas donde la base se encuentra distribuida. Esto mismo complica la implementación de seguridad.

La migración de los datos de algunas tablas (como la de personal) se hizo por medio de programas en Java que hacen uso de la biblioteca Jackcess para leer archivos de Access 2000. Escribí estos programas usando Hibernate para insertar los datos extraídos directamente en los atributos de los objetos persistentes del modelo. Estos programas se encuentran en el paquete “siidcb.dbase.migration”.

5.1.2 Plataforma de programación y servidor de aplicación: J2EE y Apache Tomcat 5.0.

Java Platform, Enterprise Edition o Java EE, es una plataforma de programación, parte de la plataforma Java, para desarrollar y ejecutar software de aplicaciones escritas en lenguaje de programación Java con arquitectura de n niveles distribuida. Se basa en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

Java EE permite al desarrollador crear aplicaciones portables entre plataformas y escalables, a la vez que integrables con tecnologías anteriores. Otros beneficios añadidos son que el servidor de aplicaciones puede manejar transacciones, seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, así, los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de ocupar tiempo en tareas de mantenimiento de bajo nivel.

El lenguaje Java es la base de la Plataforma Java. Gran parte de su sintaxis es similar a la de C o C++ pero tiene un modelo de objetos más sencillo y menos capacidades a bajo nivel. Las aplicaciones Java son compiladas a código byte (**bytecode**) que puede ser ejecutado por una **máquina virtual Java** independientemente de la arquitectura de la computadora.

Tomcat es un servidor de aplicaciones o contenedor web que implementa las especificaciones de Servlet 2.4 y de JavaServer Pages (JSP) 2.0 de Sun Microsystems. Proporciona un entorno para que código Java se ejecute en cooperación con un servidor web, el cual también trae integrado.

Se ha decidido emplear esta plataforma de programación y servidor de aplicación porque ambos son multiplataforma. Tomcat implementa seguridad si se requiere y es ligero y fácilmente configurable, y el despliegue de aplicaciones es sumamente simple: una vez que sea ha creado la aplicación, el conjunto de archivos que la constituyen son almacenados en un archivo tipo WAR (Web ARchive). Este archivo es colocado en el directorio webapps dentro de la ubicación de Tomcat en el servidor, y al iniciar el servicio Tomcat, el archivo WAR es desempaquetado y la aplicación se ejecuta, quedando lista para su uso.

5.1.3 Herramienta de Mapeo objeto-relacional: Hibernate 3.1.

Hibernate es un servicio de persistencia y consultas objeto/relacional; una capa entre la aplicación y la base de datos. Es un auxiliar en los problemas que se presentan por la discrepancia entre dos modelos para organizar y manipular datos: el relacional, empleado por las bases datos, y el de objetos empleado por la Plataforma Java. Esta herramienta es gratuita y ofrecida como *open source* y distribuida bajo la licencia GNU LGPL (Lesser General Public License).

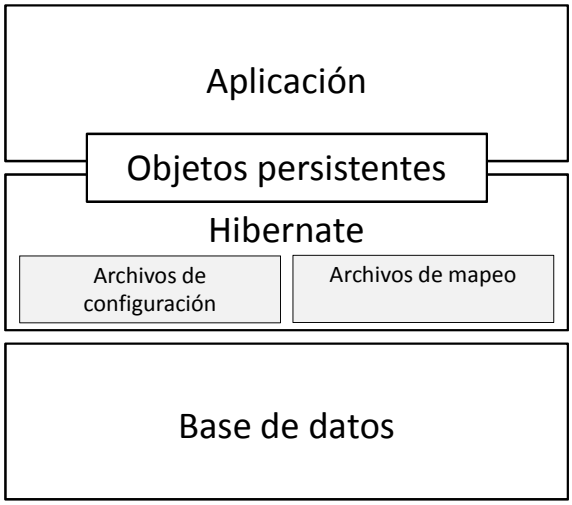


Figura 5—1 Hibernate es una capa intermedia entre la base de datos y la aplicación.

Hibernate permite crear asociaciones (*mappings*) entre las tablas en la base relacional y las clases en una aplicación desarrollada con el lenguaje orientado a objetos Java permitiendo implementar características como asociación, herencia, polimorfismo, composición y colecciones (el término para esta operación es ORM: object-relational mapping). Así, es posible hacer consultas a la base de datos a través de Hibernate y se obtendrán como resultado objetos manipulables en Java directamente sin tener que trabajar con sets de resultados.

Se decidió utilizar Hibernate porque con éste es posible explotar las capacidades del manejador de base de datos sin la necesidad de escribir código para consultas, manipulaciones e inserciones en la base de datos, y así concentrar esfuerzos en la lógica de negocio de la aplicación. Hibernate funciona en conjunto con la Plataforma Java como un componente modular; no es indispensable pero reduce el código y simplifica mucho la interacción entre la aplicación y los recursos persistentes.

Parte de la reducción del código de aplicación se debe a un aumento en el código de configuración de Hibernate, pero aún así, las ventajas de utilizar la capa de mapeo objeto/relacional son patentes durante la programación de la aplicación.

Un ejemplo de la reducción de código de aplicación que permite Hibernate es el siguiente:
Se desea hacer una consulta sobre la tabla personal en la base de datos que devuelva todos los registros.

Con Hibernate:	Sólo con JDBC
Session session = sessionFactory.openSession( );	Driver d =(Driver) Class.forName("com.MySQL.jdbc.Driver").newInstance( );
List personal = new ArrayList();	DriverManager.registerDriver(d);
try {	try {
personal = session.find("from Personal");	Connection con = DriverManager.getConnection("jdbc:MySQL://basicas/test","salo","");
Iterator it = personal.iterator();	Statement stmt = con.createStatement();
while (it.hasNext()) {	String select = "SELECT * from personal";
Personal p = (Personal)it.next();	ResultSet res = stmt.executeQuery(select);
...	List personal = new ArrayList();
}	while (res.next() == true) {
} finally {	String pID = res.getString("id");
session.close();	String pNombres = res.getString("nombres");
}	Personal p = new Personal(pID,pNombres);
	(.....)
	list.add(p);
	}
	stmt.close();
	con.commit();
	con.close();
	} catch (Throwable ex) {
	System.out.println(" Error ");
	}

Código 5—1 Comparación de código con y sin Hibernate.

En este ejemplo, en el caso de JDBC se han extraído solamente los campos ‘id’ y ‘nombres’ de cada registro de personal. En cambio, en el ejemplo con Hibernate, al inicializar la variable ‘p’ con un registro, ‘p’ adquiere todos los atributos mapeados a la tabla personal en la base de datos (v.gr. apellidos, RFC, etc.).

5.1.4 Entorno de desarrollo (IDE): Eclipse 3.2.

Eclipse es un entorno de desarrollo *open-source* para Java y otras plataformas que provee herramientas y asistencia útiles durante el ciclo de desarrollo de una aplicación. Eclipse incluye herramientas de diseño, ejecución, *debugging*, despliegue y muchas otras tereas relacionadas con el desarrollo de una aplicación. También le pueden ser agregados componentes *plugin* para frameworks específicos y otras herramientas.

5.2 Funcionamiento de SIIDCB.

El sistema recibe y atiende solicitudes HTTP con ayuda del servidor web. Por medio del archivo de configuración principal del Struts se asocian URL's a clases que despachan las solicitudes de dichas URL's. Por ejemplo, es posible asociar la URL /login.do a la clase "siidcb.acceso.action.Login" y así las solicitudes a la URL serán despachadas por la lógica de negocio contenida en la clase Login, dentro de la cual también se decide qué vista enviar al terminar el procesamiento de los datos enviados por el cliente.

Dentro de las clases controladoras de la aplicación se realizan las consultas a la base de datos por medio de las clases de modelo; éstas utilizan el servicio Hibernate configurado para acceder a la base de datos. El conjunto de clases de modelo se encuentra en el paquete "siidcb.dbase".

Las vistas que son enviadas al cliente cuando finaliza la lógica de negocio una clase controladora son páginas JSP que se utilizan como plantillas que son rellenadas con datos extraídos de las base de datos.

La Figura 5—2 es un diagrama de secuencia que muestra el funcionamiento general del sistema.

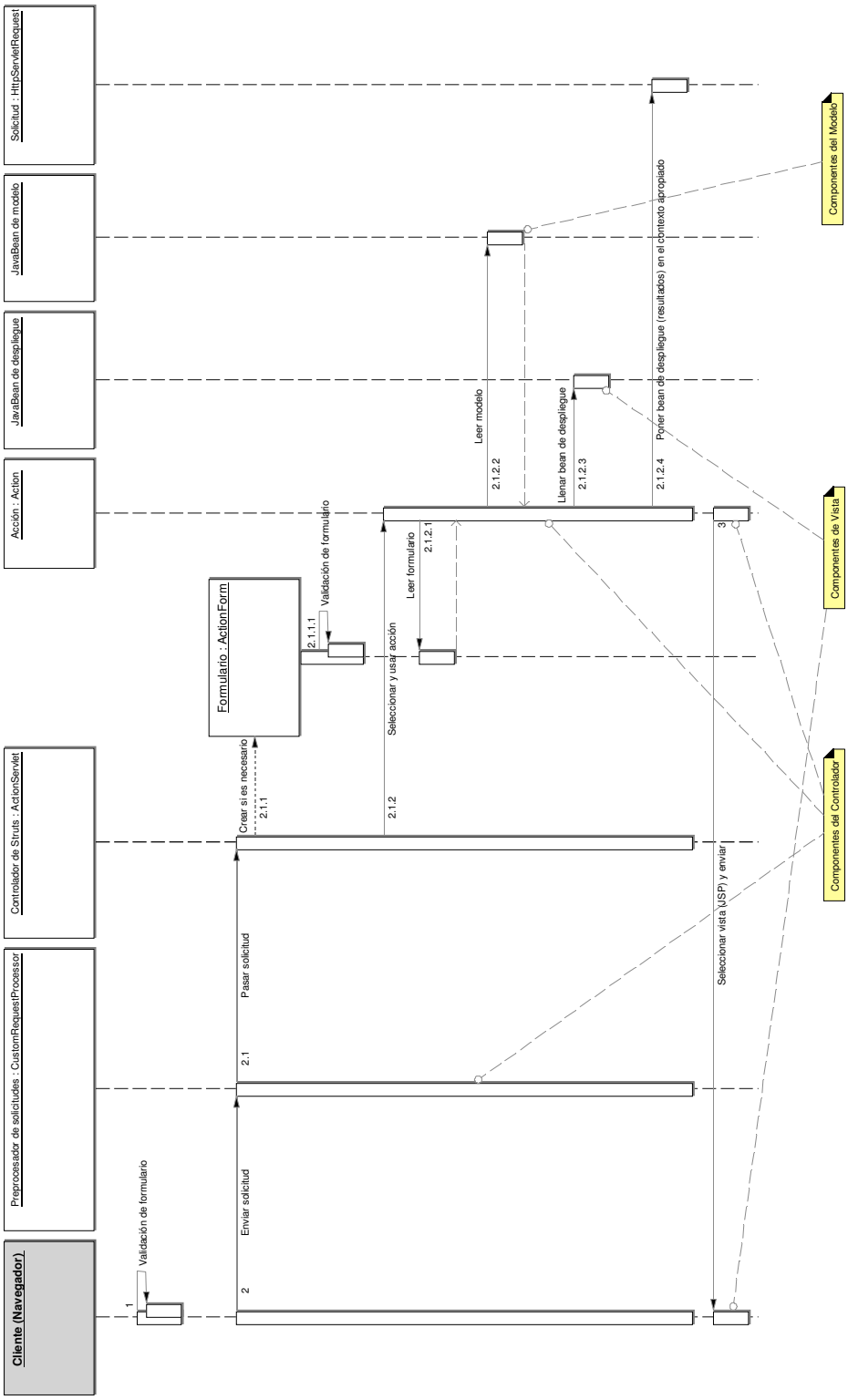


Figura 5—2 Diagrama de secuencia que muestra la operación general del sistema.

Los elementos mostrados en el diagrama se explican a continuación.

Clase preprocesadora de solicitudes.
En esta clase, que funciona como un filtro, es posible implementar diversas tareas de revisión de una solicitud antes de procesarla. Si se requiere llevar registro total de las solicitudes entrantes, sus direcciones IP, hora en que son recibidas, etc., esta clase, que debe extender a org.apache.struts.tiles.TilesRequestProcessor proporciona los medios para hacerlo. Dentro del preprocesador de solicitudes se ha implementado la autenticación y autorización para acceder a acciones que la requieren.

Clase ActionServlet.
Este es el Servlet de Struts y provee a la aplicación del componente controlador en el patrón MVC. Este Servlet recibe solicitudes a acciones (URL's con la extensión .do) y la pasa a la clase correspondiente asignada en un archivo de configuración (/WebRoot/WEB-INF/struts-config.xml).

Clase Action.
Dentro de clases que extienden a ésta se lleva a cabo la lógica de negocio de la aplicación. Pueden verse como partes del controlador que llevan a cabo determinadas tareas. También es desde estas clases desde donde se lee el modelo generalmente, y esto se hace con la utilización de las clases de modelo del paquete "siidcb.dbase" y que acuden al servicio Hibernate.

Clase ActionForm.
La primera función de esta clase es mapear los formularios que son desplegados al usuario (contenidos en páginas JSP) a JavaBeans fácilmente manipulables. Esta clase también simplifica la validación de los datos ingresados por el usuario por los formularios mencionados. De modo que dentro de la clase Acción, que implementa la lógica de negocio, podemos tomar los datos del JavaBean del formulario validado sin problemas. Se puede implementar tanto validación simple (no necesita consultar el modelo) como compleja (lee el modelo, es decir, la base de datos) con esta clase, y la simple puede además llevarse a cabo del lado del cliente si éste es un navegador que cuenta con JavaScript. Hay que resaltar que la validación de datos, se lleve o no a cabo del lado del cliente, se realiza en el servido también. Más capas de validación contribuyen a una mayor robustez e integridad en los datos almacenados.

JavaBeans de modelo.
Estas son clases que tienen atributos mapeados al modelo físico (base de datos) por medio de archivos de configuración. Estos beans utilizan el servicio de persistencia Hibernate para acceder a la base de datos. Como cualquier JavaBean, cuentan con métodos *set* y *get* para cada atributo.

JavaBeans de despliegue.
Son beans auxiliares que se utilizan para colocar en ellos los datos que serán pasados a la vista (JSP) y ésta proporcione información al usuario. Por lo general, cuando una acción ha terminado su lógica, coloca uno o varios beans de despliegue en el contexto de sesión (session context) o de solicitud (request context) para que la página JSP extraiga de ellos los datos y los imprima en una página que se envía al cliente (navegador). Pueden usarse como beans de despliegue las clases que extienden ActionForm ya que son beans también; o bien, beans simples creados en su totalidad para el despliegue. En el paquete siidcb.beans se encuentran los de este último tipo.

Acceso al sistema y control de acceso a URL's.

El acceso al sistema se lleva a cabo por medio de un formulario con campos para nombre de usuario y contraseña. Una vez que el cliente ha enviado estos datos, el sistema los verifica consultando la base de datos y si son válidos coloca en la sesión HTTP asociada a ese cliente un JavaBean que contiene, entre otros datos, el tipo de cuenta que ha accedido al sistema. Esto se lleva a cabo en la clase siidcb.acceso.action.UserLoginAction.

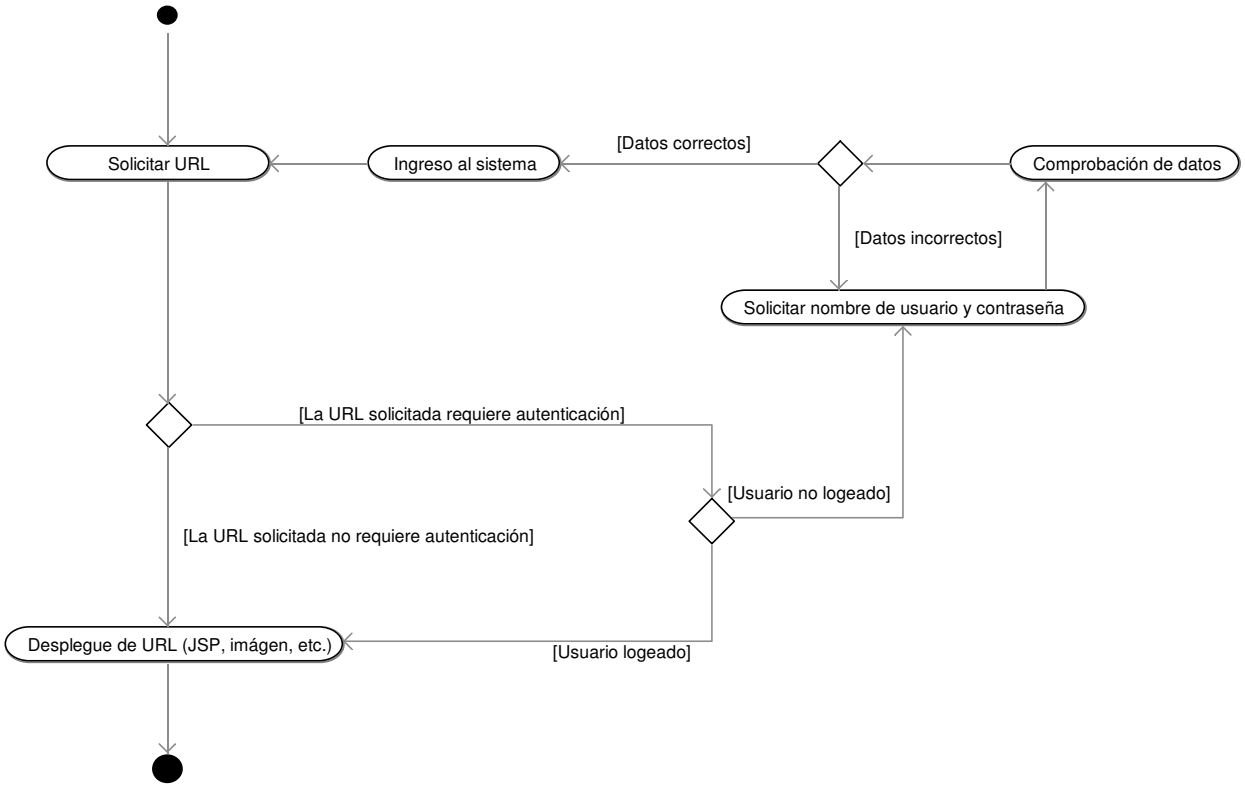


Figura 5—3 Diagrama de secuencia que muestra cuando se solicitan datos de acceso.

La clase preprocesadora de solicitudes (siidcb.CustomRequestProcessor) verifica si un usuario ha iniciado sesión y si ésta es válida antes de proporcionar el recurso que solicite. De este modo también es posible limitar los contenidos a los que puede acceder cada tipo de cuenta. Para esto se han utilizado las URL's de la manera siguiente:

- /administrador/* Sólo puede acceder las cuentas de tipo "Administrador".
- /funcionario/* Pueden accederlas los tipos "Administrador" y "Funcionario".
- /profesor/* Pueden accederlas los tres tipos de cuenta.

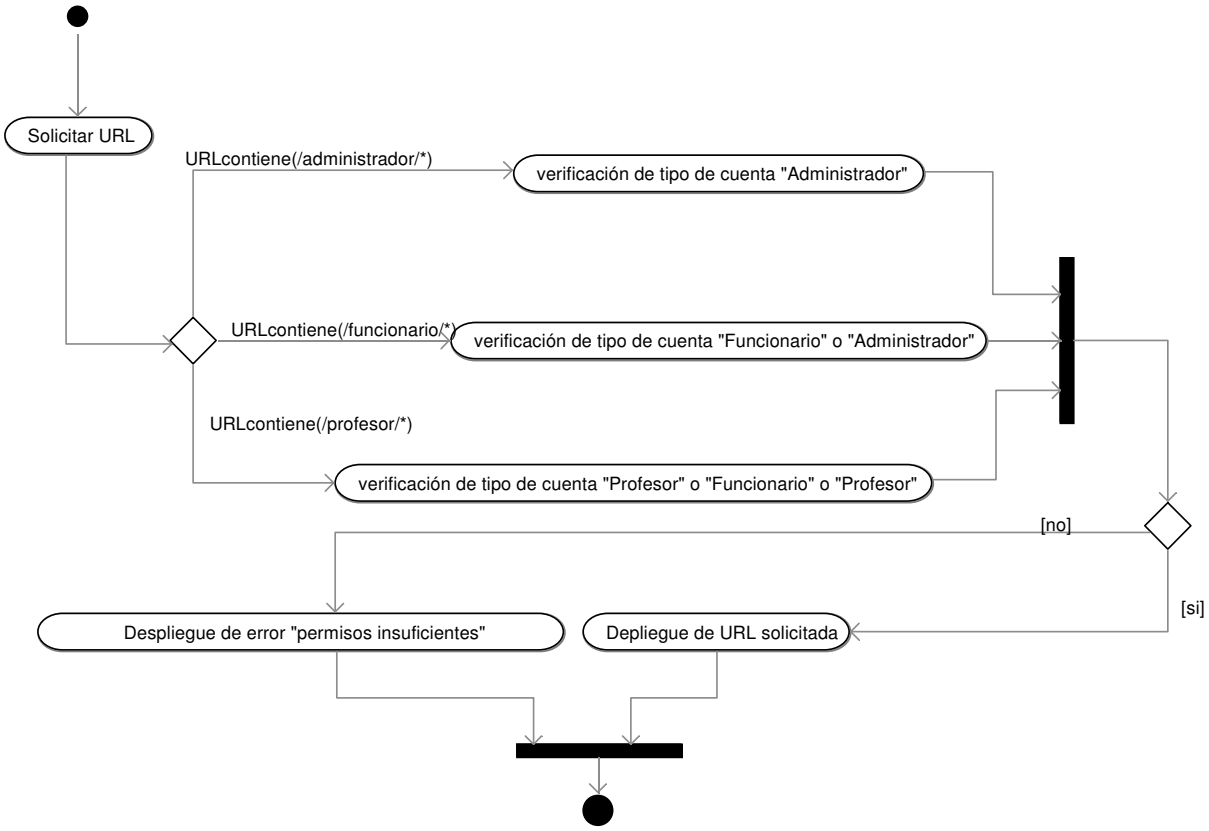


Figura 5—4 Diagrama de secuencia que muestra el algoritmo de control de acceso a URL's según el tipo de cuenta.

Validación de datos de entrada en formularios.

La validación de los datos que ingresan los usuarios en los formularios se hace con el marco de validación que proporcional Struts y se lleva a cabo tanto del lado del cliente con JavaScript como del lado del servidor en las clases controladoras. Se lleva a cabo de ambos lados de la aplicación por si el cliente no cuenta con JavaScript pero no hay duplicación de código de validación porque el marco de validación de Struts genera código JavaScript automáticamente basado en las validaciones que se emplean también del lado del servidor y que son definidas en el archivo /WebRoot/WEB-INF/validation.xml.

5.3 Estructura de archivos de la aplicación.

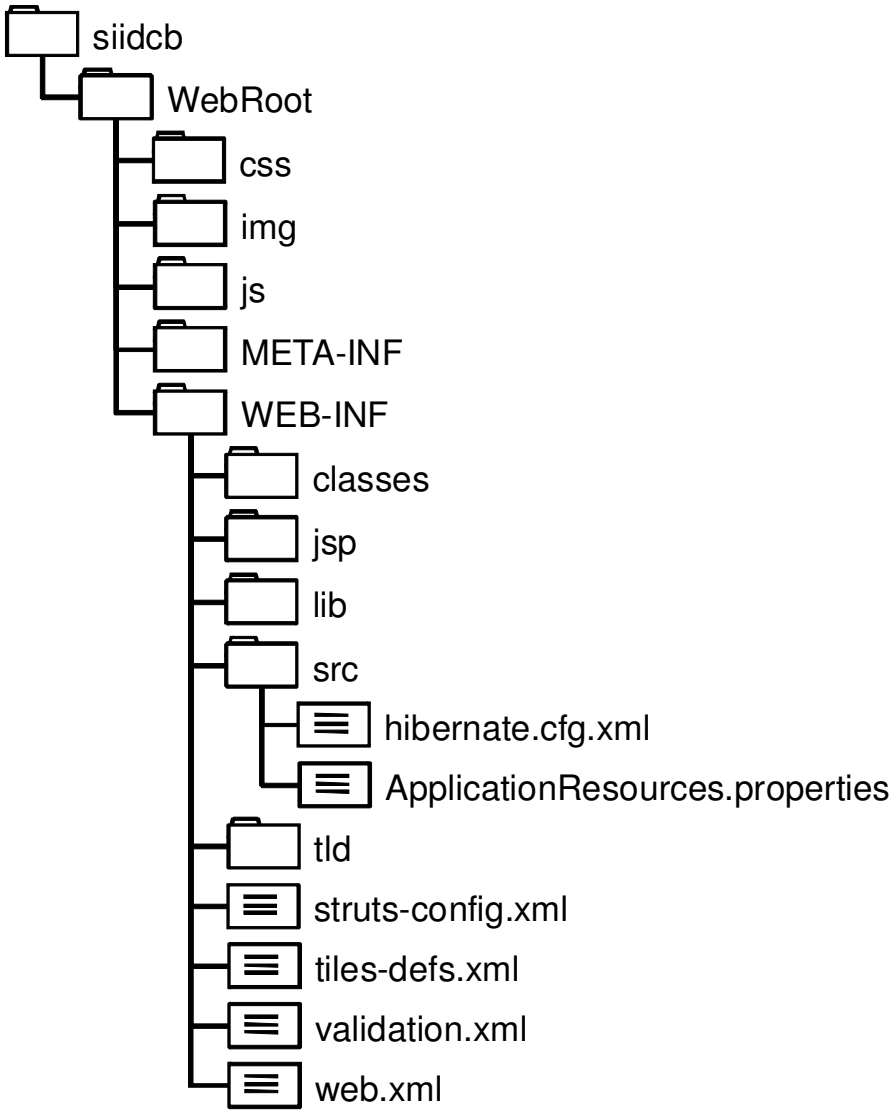


Figura 5—5 Estructura de archivos dentro de la aplicación SIIDCB.

- /WebRoot/ – Toda aplicación web de Tomcat debe tener una raíz que es este directorio; dentro de él Tomcat busca los recursos que la constituyen: clases, archivos jsp, html, imágenes, etc. Los archivos colocados dentro de este directorio, con excepción de los que están dentro de WEB-INF y META-INF, son accesibles directamente sin intervención de los controladores de la aplicación. Por eso aquí pueden colocarse archivos estáticos como en el caso de las

imágenes en img. Pueden ponerse también páginas HTML o JSP para despliegue al público en general.

- /WebRoot/css/ – Contiene los archivos *.css (cascading style sheets) para dar formato a las páginas HTML y JSP que hacen uso de ellos.

- /WebRoot/img/ – Contiene archivos de imagen *.jpg y *.gif para despliegue en las páginas HTML y JSP.

- /WebRoot/js/ – Contiene los scripts escritos en JavaScript. Archivos *.js para añadir algunas funcionalidades a páginas HTML y JSP.

- /WebRoot/META-INF/ – Aquí pueden guardarse archivos descriptores del archivo de despliegue .WAR. Estos archivos pueden contener información como la versión entregada en el WebArchive. No es un directorio público.

- /WebRoot/WEB-INF/classes/ – Aquí se encuentran las clases compiladas de la aplicación organizadas por directorios de paquetes. Se trata de archivos binarios *.class.

- /WebRoot/WEB-INF/jsp/ – Todos los archivos jsp que emplea la aplicación como vistas se encuentran aquí ordenados en directorios de acuerdo al módulo al que corresponden. Se trata de archivos *.jsp.

- /WebRoot/WEB-INF/lib/ – Directorio que contiene todas las librerías que utiliza la aplicación. Son archivos *.jar (Java ARchive) que contienen clases compiladas y archivos descriptores.

- /WebRoot/WEB-INF/src/ – Directorio contenedor de todos los archivos de código fuente de la aplicación, organizados por directorios de paquetes. Archivos *.java. También se encuentran en este directorio algunos archivos de configuración como log4j.properties para el *logging* (bitácoras) y otros archivos *.xml.

- /WebRoot/WEB-INF/src/hibernate.cfg.xml – Archivo de configuración del *plugin* de Hibernate.

- /WebRoot/WEB-INF/src/ApplicationResources.properties – Este archivo/clase contiene parámetros estáticos que pueden ser empleados por Struts. Es usado para guardar configuración de la aplicación y para guardar textos que aparecen en diversas paginas y sólo cambian ocasionalmente (como mensajes de bienvenida) y así evitar duplicación de código.

- /WebRoot/WEB-INF/tld/ – Directorio contenedor de los archivos descriptores de bibliotecas de etiquetas (Tag Library Descriptor). Archivos *.tld que permiten utilizar las librerías de etiquetas en las páginas jsp.

- /WebRoot/WEB-INF/struts-config.xml – Este archivo contiene la configuración del Servlet de Struts. El controlador principal de la aplicación.

- /WebRoot/WEB-INF/tiles-defs.xml – Archivo que describe plantillas para reutilización de código de presentación jsp o html. Lo utiliza el plugin de Tiles de Struts.

- /WebRoot/WEB-INF/validation.xml – Archivo que describe la validación de los formularios en la aplicación. Lo utiliza el plugin de validación de Struts.

- /WebRoot/WEB-INF/web.xml – El archivo descriptor de despliegue de la aplicación web que contiene la configuración de ésta.

5.4 Estándares para nombres de archivos de la aplicación.

Para archivos de código Java se ha seguido la convención sugerida por Sun Microsystems en la Sun Developer Network (consultar las referencias). En resumen, las convenciones seguidas son:

- Las extensiones de archivos de clases son .java

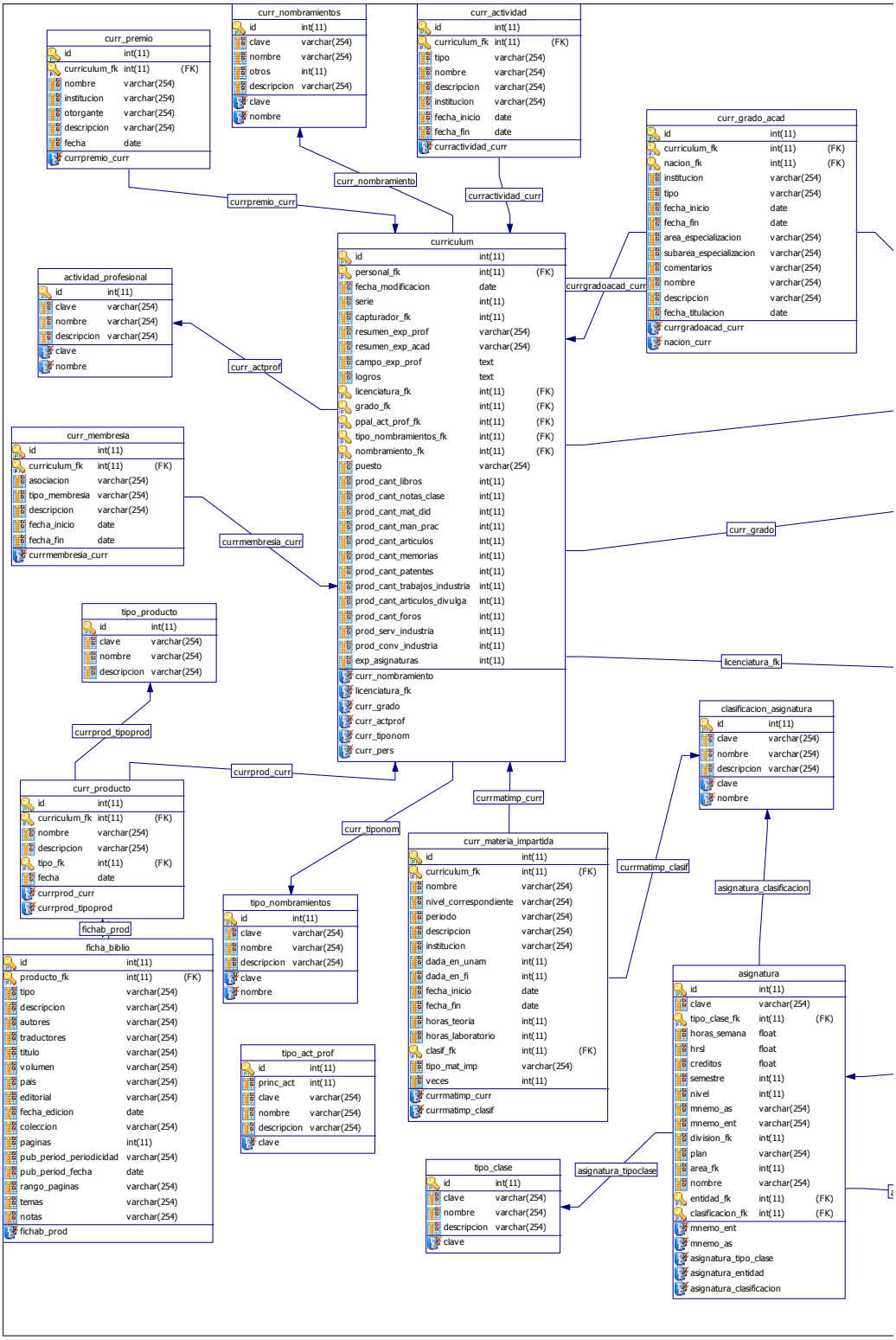
- Los nombres de paquetes siempre son con minúsculas y se usa punto para indicar sub paquetes. Por ejemplo, si el paquete “java” contiene al subpaquete “awt”, el nombre completo del paquete “awt” es “java.awt”, y si éste tiene otro paquete llamado “color”, “java.awt.color” es el nombre completo (*cualified name*) del paquete “color”.

- Los nombres de las clases deben ser descriptivas, empezar con mayúscula y para separar palabras se usa mayúscula en la primera letra de cada palabra. Por ejemplo Personal o FichaBibliografica. No se usa guión ni guión bajo.

Para el resto de archivos de la aplicación (archivos de imagen, jsp, html, xml, etc) se han utilizado nombres sólo en minúsculas y se usa guión bajo para separar palabras. Por ejemplo, personal.gif, inicio.jsp, etc.

Finalmente, los archivos están ordenados en carpetas que indican el módulo o la función a la que pertenecen.

5.5 Diagrama Entidad Relación



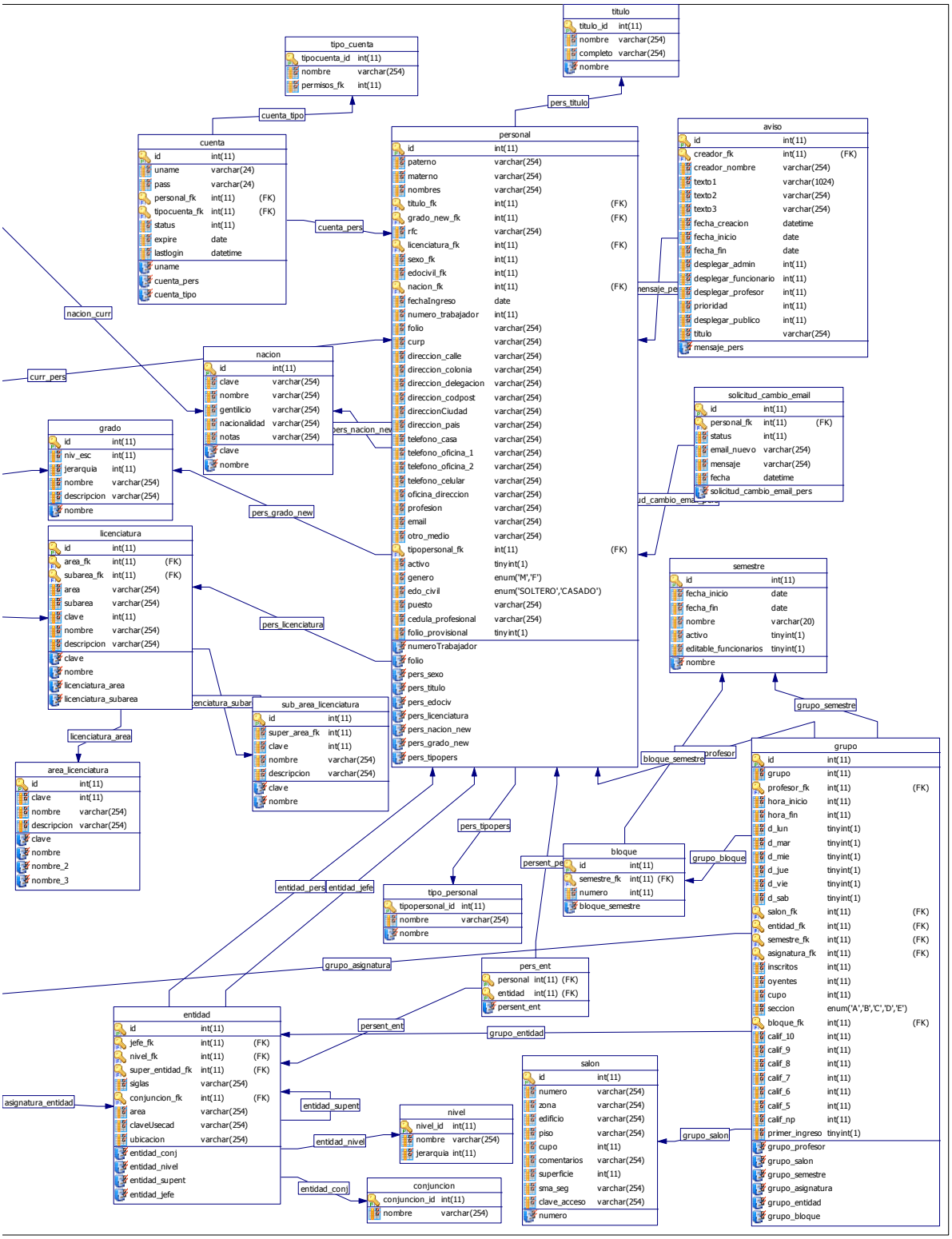


Figura 5—6 Diagrama Entidad-Relación de la base de datos de la aplicación SIIDCB.

5.6 Diccionario de datos.

Debido a que el diccionario de datos para la base de datos de la aplicación SIIDCB es muy extenso, éste ha sido incluido como un anexo al final del presente trabajo.

5.7 Conexión a la base de datos.

Para el acceso a la base de datos se utiliza JDBC, Hibernate y la biblioteca c3p0. Con esta última se prepara un conjunto de conexiones (*conection pool*) a la base de datos que utilizan JDBC y que se mantienen abiertas mientras el servicio Hibernate está activo. Por medio de Hibernate, desde la aplicación es posible hacer uso de dichas conexiones. Hibernate funciona como capa intermedia entre nuestra base de datos y la aplicación.

Cuando la aplicación inicia, un *plugin* crea el *pool* de conexiones. El plugin es una instancia de la clase siidcb.struts.HibernatePlugin que al iniciar lee datos de configuración desde el archivo /WEB-INF/src/hibernate.cfg.xml y coloca en contexto de aplicación una fábrica de sesiones que puede proporcionar sesiones para trabajar la base de datos.

Para obtener un objeto que tiene acceso a la base de datos por medio del servicio Hibernate sólo es necesario crear un objeto de la clase ServletContext que haga referencia al contexto del Servlet de la aplicación. En contexto de sesión existe un atributo llamado SESSION_FACTORY_KEY que hace referencia a una única fábrica de sesiones de Hibernate para toda la aplicación.

Un ejemplo de obtención de una sesión de Hibernate para trabajar la base de datos:

```
/* obtener el contexto de sesión: */
ServletContext context =  request.getSession().getServletContext();
/* obtener la fábrica de sessiones que existe en el contexto: */
SessionFactory _factory = (SessionFactory)
context.getAttribute(HibernatePlugIn.SESSION_FACTORY_KEY);
/*Abrir la sessión hibernate: */
Session hibernateSession = _factory.openSession();
```

Código 5—2 Acceso a la base de datos desde la aplicación SIIDCB.

En este ejemplo, por medio del objeto hibernateSession se puede hacer cualquier operación sobre la base de datos. Cuando ha terminado de emplearse la conexión, es necesario cerrarla para que esté nuevamente disponible en el depósito de conexiones. Esto se logra con el método *close* de la clase HibernateSession:

```
/*Cerrar la sessión hibernate: */
hibernateSession.close();
```

Código 5—3 Finalización de una sesión de Hibernate desde una clase.

5.8 Ejemplo de clase controladora.

La clase controladora para este ejemplo se encuentra en el paquete siidcb.ejemplos. Las clases controladoras accesibles al cliente por medio de una URL están configuradas en el archivo de configuración del Servet de Struts (el controlador principal de la aplicación). El archivo /WebRoot/WEB-INF/struts-config.xml contiene varios apartados dentro de la sección principal <struts-config>. Los URL asociados a acciones (clases controladoras) se encuentran en el apartado <action-mappings>. A continuación se muestra la configuración que mapea la URL /LoginAction.do a la clase controladora siidcb.struts.action.LoginAction.

```
<action path="/LoginAction"
      type="siidcb.ejemplos.LoginAction"
      name="UserLoginForm"
      scope="request">
<forward name="failure" path="/login.jsp "/>
<forward name="success" path="/bienvenido.jsp" />
</action>
```

Código 5—4 Configuración de la clase que despacha la URL /LoginAction.

“path” especifica la URL asociada a la clase controladora.
“type” es la clase controladora.
“name” es el ActionForm que utiliza esta clase controladora para obtener datos de un formulario.

Los elementos <forward> son las vistas asociadas a este controlador. En este caso fueron llamados “failure” y “success”. Como veremos, el controlador decide cuál de estas dos vistas será desplegada al final de realizar la lógica.

Lo que se indica al Servlet de Struts con la configuración mostrada significa que cuando un usuario solicite o envié datos a la URL /LoginAction.do a través de su navegador, la solicitud será procesada por la clase controladora siidcb.struts.action.UserLoginAction que decidirá, con base en los datos enviados en la solicitud por medio de un formulario, qué vista mostrar: la página login.jsp o bienvenido.jsp.

Veamos ahora una página por medio de la cual se puede acceder a la URL /LoginAction.do:

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html-el" %>
<html><head><title>Login</title></head><body>
<html-el:form action="/LoginAction.do">
            Nombre de usuario:<html-el:text property="userName"/><br>
            Contraseña: <html-el:password property="password"/><br>
            <html-el:submit value="Iniciar sesión"/>
            ${UserLoginForm.aviso}
</html-el:form>
</body></html>
```

Código 5—5 Página JSP que solicita la URL /LoginAction.

La primera línea del código especifica una librería de etiquetas que se usa en la página. El código restante es para crear un formulario con dos campos de texto y un botón de envío de datos. Es importante notar que el atributo “action” del formulario indica a cuál URL mandará los datos del formulario al presionar el botón. En este caso envía los datos a la URL /userLogin.do. El código produce la siguiente página web:

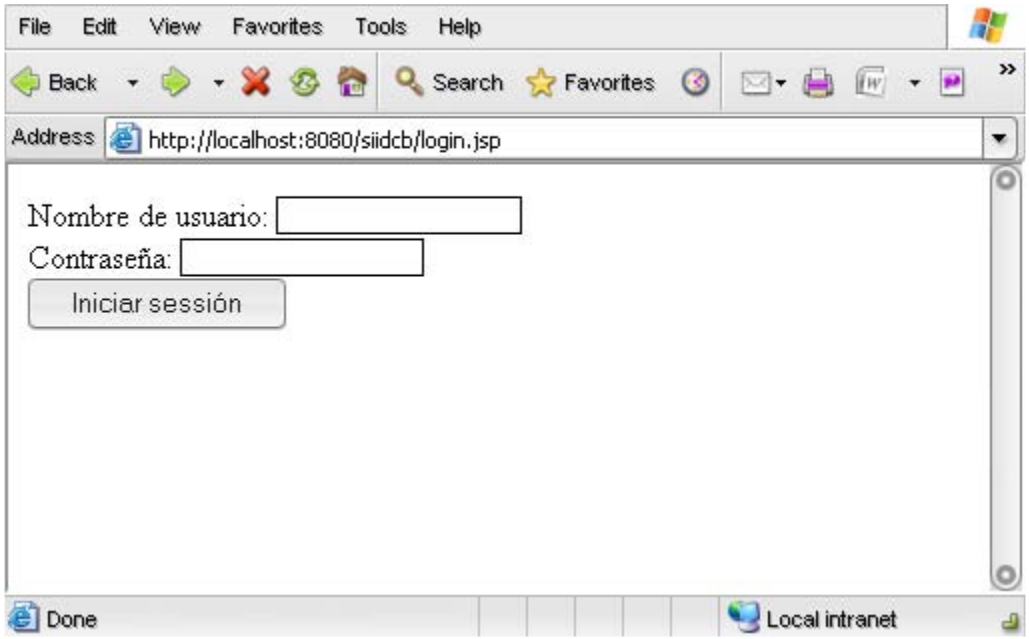


Figura 5—7 Página JSP mostrada por el navegador.

Al presionar el botón los datos contenidos en el formulario son enviados a la URL /LoginAction.do. El controlador principal de Struts, basado en su configuración, crea un *bean* de formulario userLoginForm y asigna a los atributos “userName” y “password” los valores contenidos en el formulario. Este objeto formulario es pasado a la clase controladora mapeada a la URL /LoginAction.do: siidcb.struts.action.LoginAction.

Veamos ahora código del controlador siidcb.ejemplos.LoginAction:

```
package siidcb.struts.action;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletContext;
import org.apache.struts.action.*;
import org.hibernate.*;
import siidcb.struts.HibernatePlugIn;
import siidcb.struts.form.UserLoginForm;
import siidcb.dbase.*;

public class LoginAction extends Action {

      public ActionForward execute(ActionMapping mapping, ActionForm form,
                  HttpServletRequest request, HttpServletResponse response) {
            //Obtenemos el objeto que tiene la información del formulario:
            UserLoginForm userLoginForm = (UserLoginForm) form;

            //extraemos los datos de ese formulario:
            String uname = userLoginForm.getUserName();
            String password = userLoginForm.getPassword();

            //si no se ingresó username o password mandar a la vista
"failure"
            if(esVacio(uname)){
                  userLoginForm.setAviso("ingrese nombre de usuario");
                  return mapping.findForward("failure");
            }

            if(esVacio(uname)||esVacio(password)){
                  userLoginForm.setAviso("ingrese password");
                  return mapping.findForward("failure");
            }

            //Obtenenos una sesión del servicio Hibernate para acceder a la
base de datos
            ServletContext context =
request.getSession().getServletContext();
            SessionFactory _factory = (SessionFactory)
            context.getAttribute(HibernatePlugIn.SESSION_FACTORY_KEY);
            Session hibernateSession = _factory.openSession();

            //Creamos un objeto de acceso a datos (DAO) para objetos de la
clase Cuentas
            CuentaDAO dao = new CuentaDAO();
            dao.setSession(hibernateSession );

            //creamos un objeto de la clase Cuenta
            Cuenta c = null;

            //buscamos la cuenta por medio del username:
            if(dao.findByUname(uname).isEmpty()) //si no existe la cuenta:
```

```
            {
                  userLoginForm.setAviso("cuenta inexistente");
                  hibernateSession .close();
                  return mapping.findForward("failure");
            }
            else //si existe la cuenta:
            {
                  c = (Cuenta)dao.findByUname(uname).iterator().next();

                  //revisamos si el password ingresado coincide con el
password de la cuenta:
                  if(c.getPass().equals(password))//si el password es
correcto:
                  {
                        //ponemos el nombre del dueño de la cuenta en la
variable nombre
                        //para que pueda ser leida por la JSP
                        request.setAttribute("nombre",
c.getPersonal().getNombres());
                        hibernateSession .close();
                        return mapping.findForward("success");

                  }
                  else//si el password es incorrecto:
                  {
                        hibernateSession .close();
                        userLoginForm.setPassword("");
                        userLoginForm.setAviso("datos incorrectos");
                        return mapping.findForward("failure");
                  }
            }
      }

      private boolean esVacio(String value){
            return((value==null||(value.trim().equals(""))));
      }
}
```

Código 5—6 Ejemplo de código de una clase controladora.

Como se ve en el código, la clase posee un método llamado “execute”. Este método es donde se lleva a cabo la lógica que decide qué vista será desplegada a continuación. Para explicar la lógica de esta clase controladora se proporciona un diagrama de actividad:

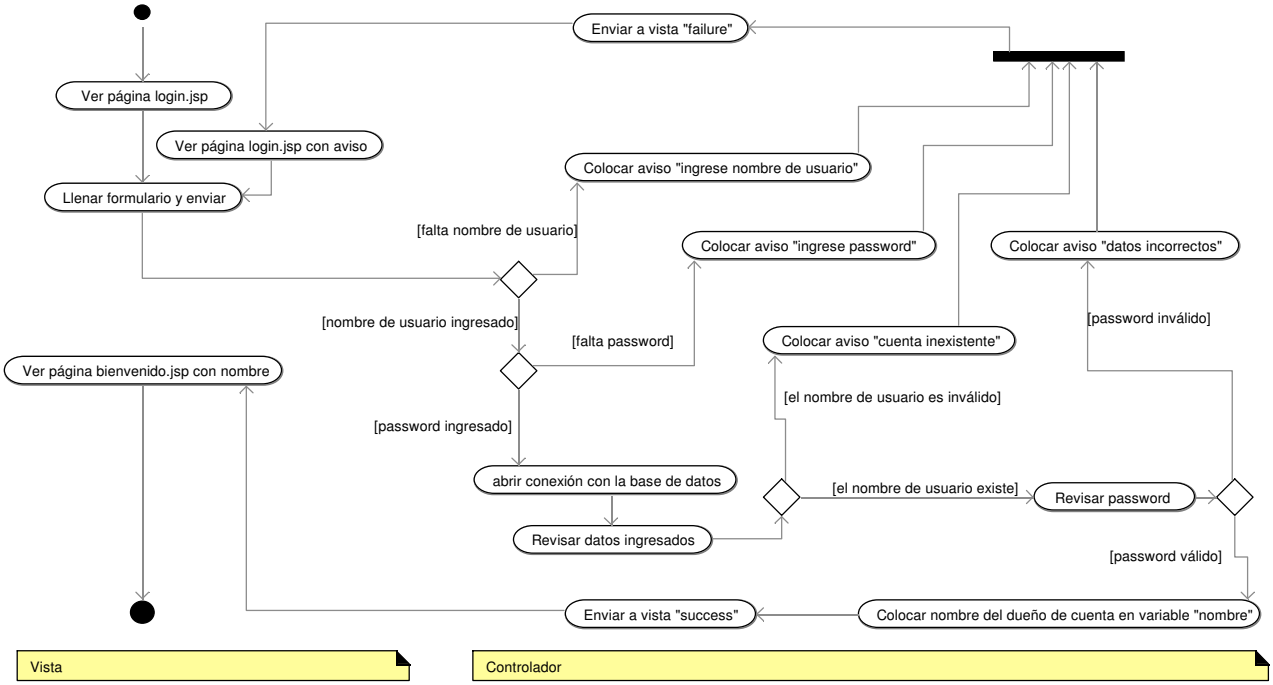


Figura 5—8 Diagrama de actividades de un ejemplo de controlador.

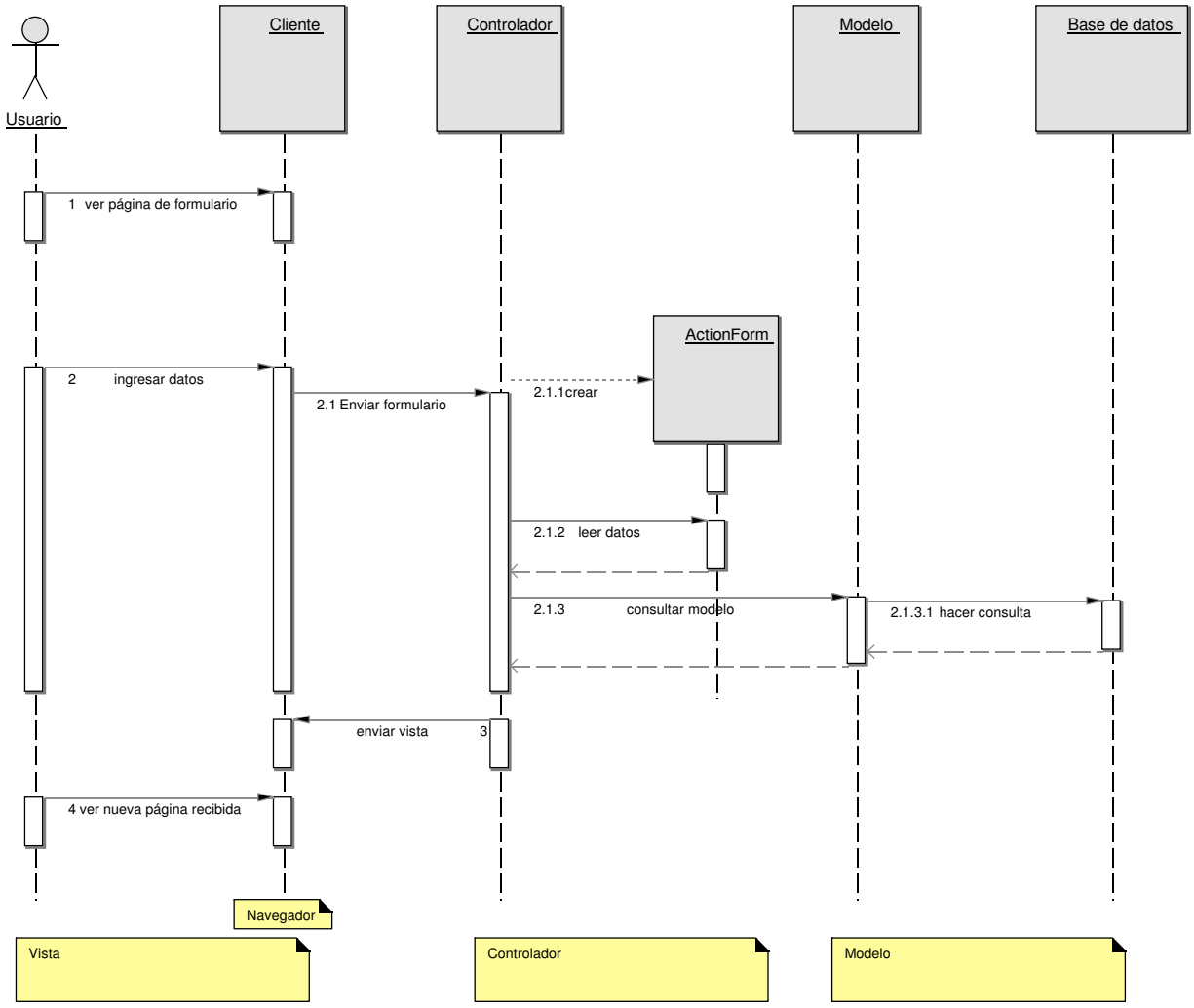


Figura 5—9 Diagrama de secuencia que ilustra el funcionamiento del controlador.

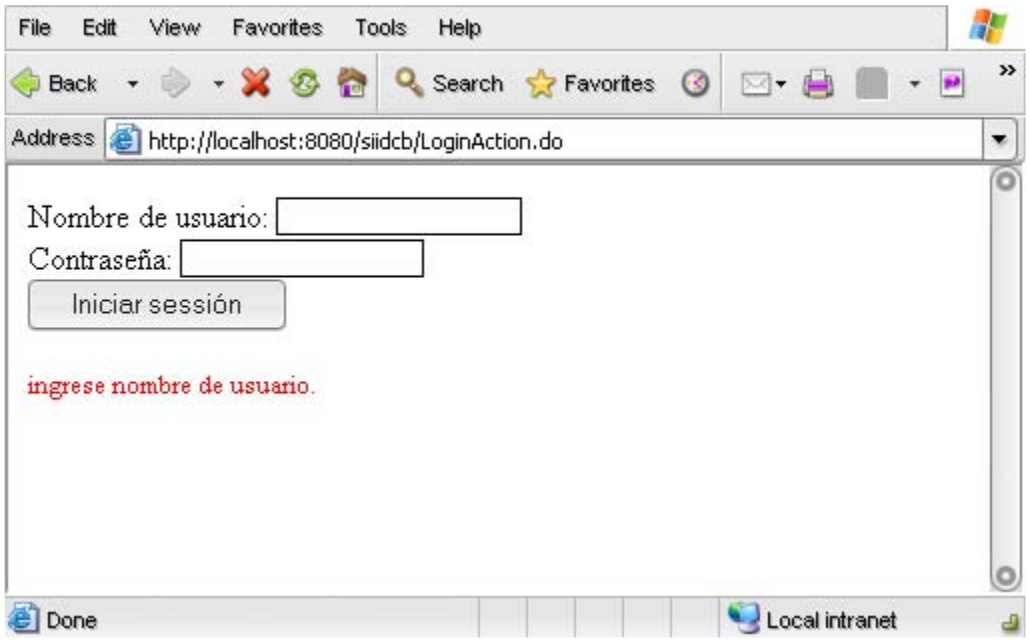


Figura 5—10 Vista entregada cuando no se ingresa nombre de usuario.

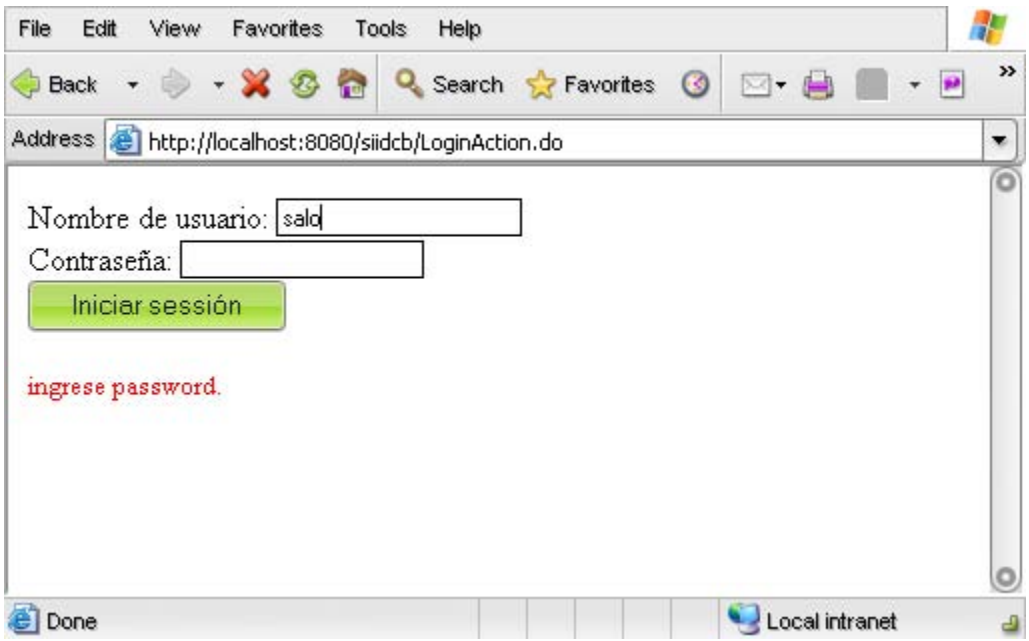


Figura 5—11 Vista entregada cuando no se ingresa contraseña.

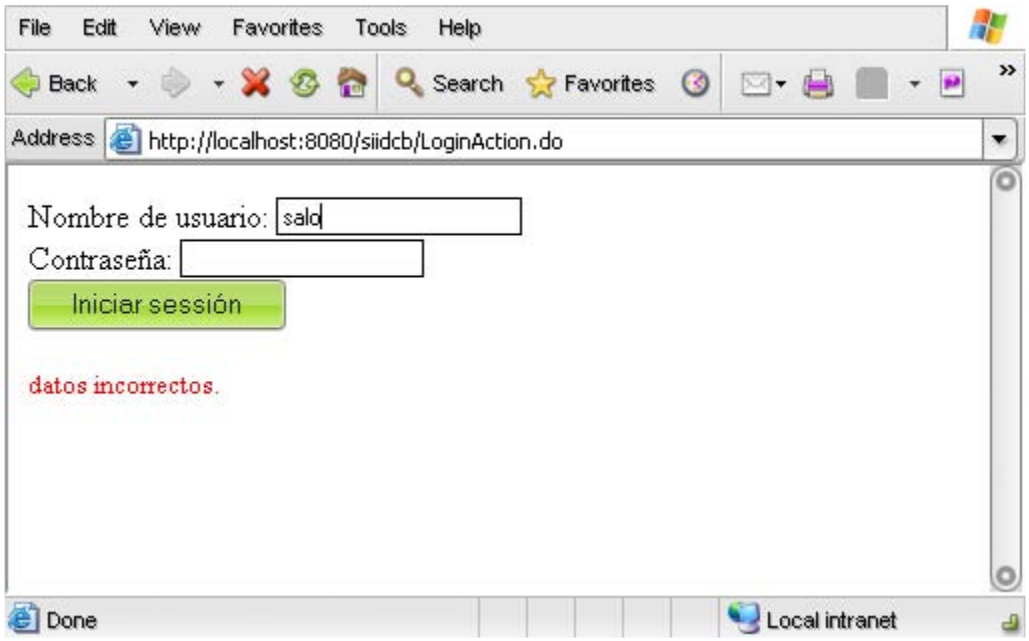


Figura 5—12 Vista entregada cuando la contraseña es incorrecta.

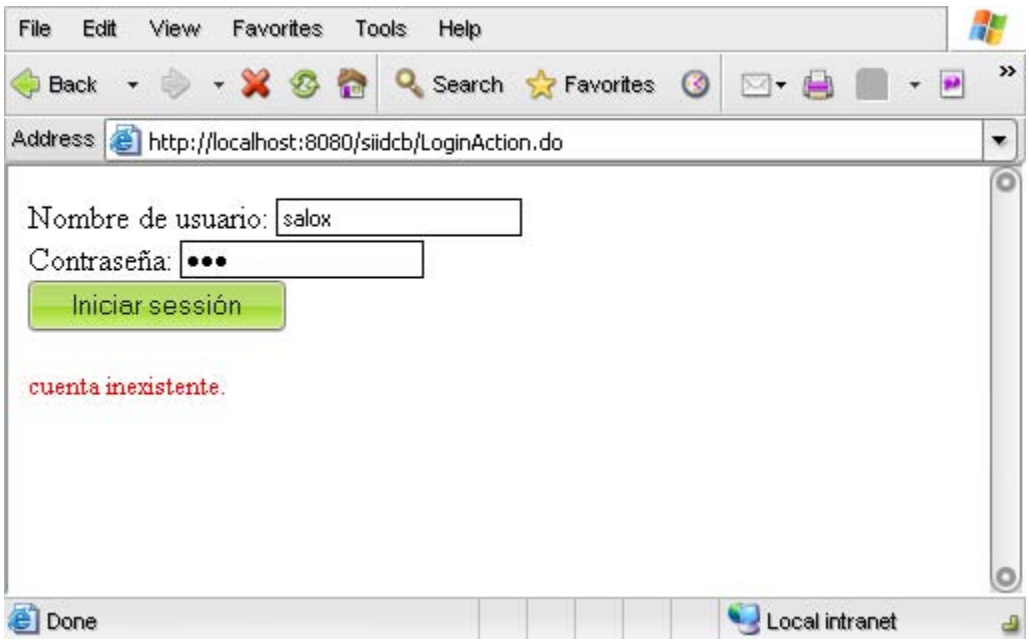


Figura 5—13 Vista entregada cuando no existe el nombre de usuario ingresado.

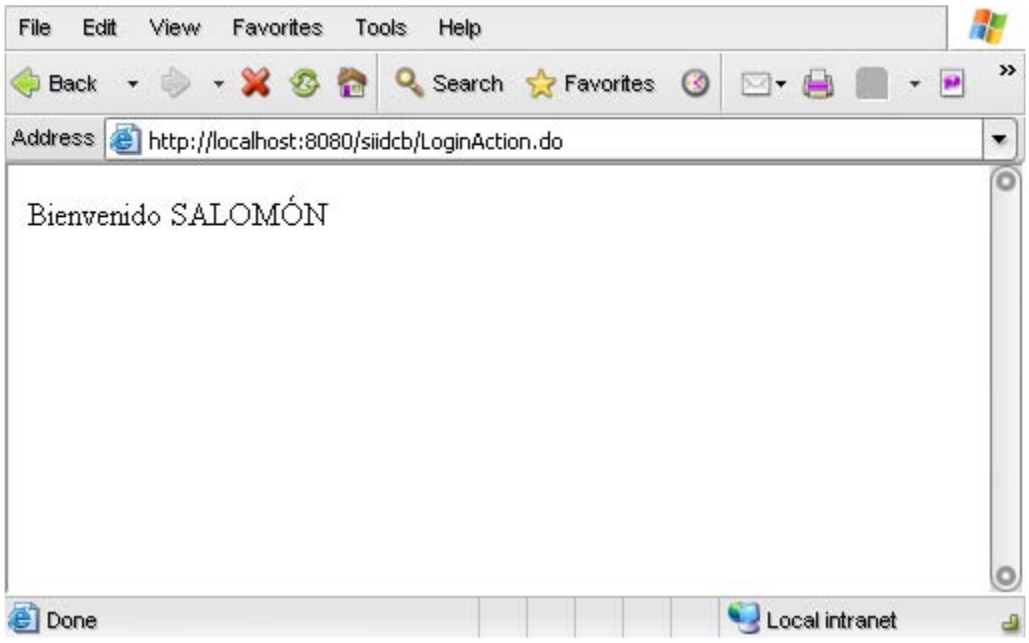


Figura 5—14 Vista entregada cuando los datos son correctos.

Código de la vista “success” (bienvenido.jsp) del controlador del ejemplo:

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html-el" %>
<html><head><title>Bienvenido</title></head><body>
Bienvenido ${requestScope.nombre}
</body></html>
```

Código 5—7 Código de la vista "success" del controlador de ejemplo.

5.9 Ejemplo de funcionamiento de una vista.

La manera en que la presente aplicación despliega las vistas se muestra en el diagrama de secuencia de la Figura 5—15.

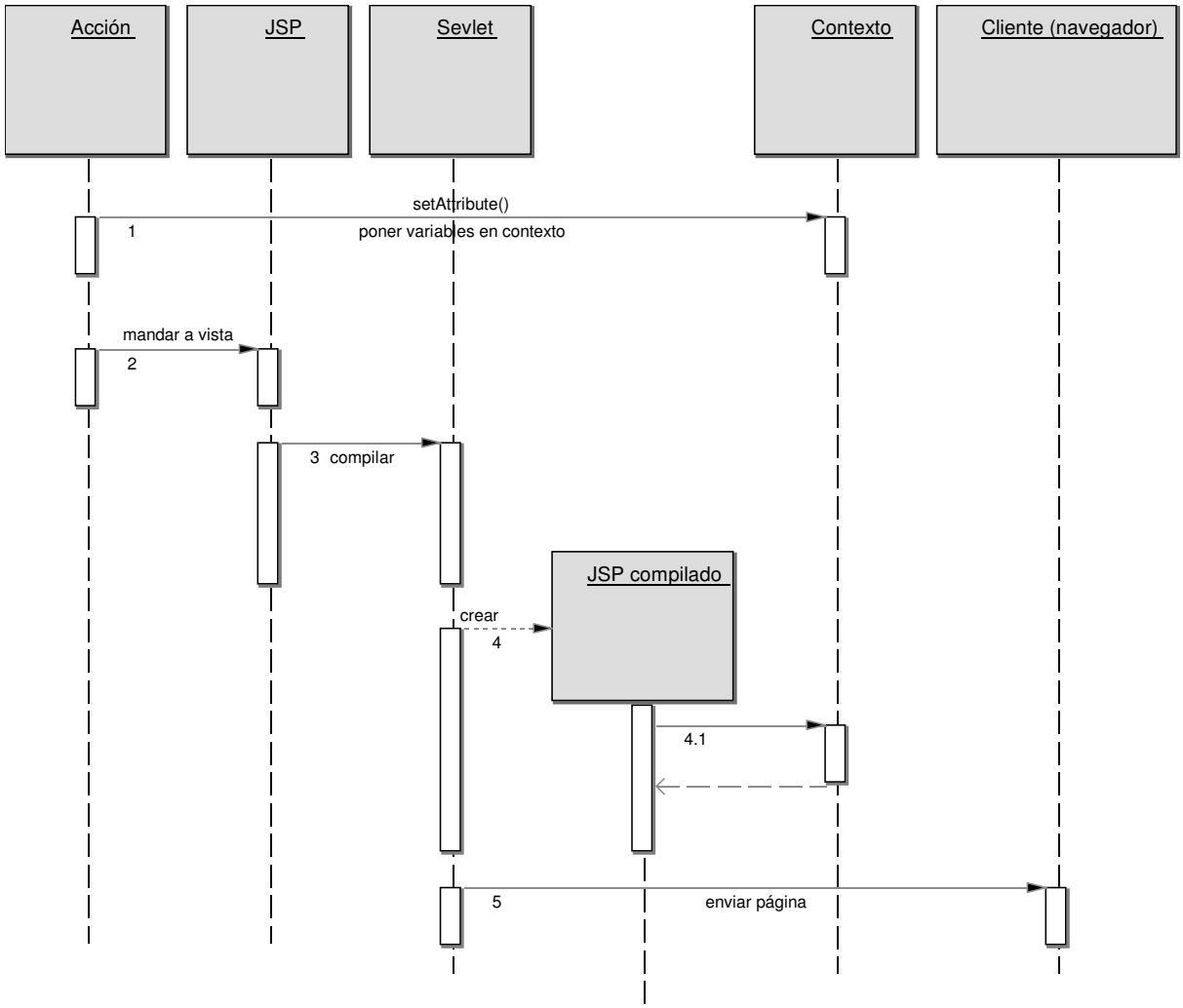


Figura 5—15 Diagrama de secuencia del funcionamiento de un componente de vista.

El Código 5—8 es un fragmento de una página JSP. Una vez que una clase acción (controlador) ha puesto en el contexto los beans de despliegue y/o otras variables, pasa (hace un *forward*) a esta página JSP, que es compilada por el servlet remplazando las etiquetas especiales con los datos contenidos en los beans de despliegue.

```
<display:table name="resultados">
  <display:column property="numeroTrabajador" title="No.Trab."/>
  <display:column property="titulo" title="Título"/>
  <display:column property="paterno" title="Paterno"/>
  <display:column property="materno" title="Materno"/>
  <display:column property="nombres" title="Nombres"/>
</display:table>
```

Código 5—8 Código de vista que genera una tabla con resultados.

Por ejemplo, la etiqueta <display:table> lee la variable indicada por la propiedad “name” e imprime una tabla con los datos en ella. En este caso se trata de la variable “resultados”, que es una lista de beans de resultados que poseen los atributos “numeroTrabajador”, “titulo”, “paterno”, “materno” y “nombres”. Este código produce la tabla a partir de una lista de beans de resultados en la variable “resultados”:

No.Trab.	Título	Paterno	Materno	Nombres
	Ing.	CONTRERAS	CASTRO	JAIME
	M.C.	CONTRERAS	CELEDÓN	CLAUDIA ARACELI
	Ing.	LÓPEZ	CONTRERAS	JULIAN
115371	Ing.	PÉREZ	CONTRERAS	EDUARDO
26466	Ing.	PÉREZ	CONTRERAS	AGUSTÍN
832183	Sr.	RAMÍREZ	CONTLA	SALOMÓN

Figura 5—16 Tabla generada por el código de ejemplo.

Capítulo 6. Pruebas.

6.1 Pruebas de volumen.

Para las pruebas de volumen se ha utilizado la versión de evaluación del programa WAPT (Web Applications Testing) versión 5.0. Éste puede simular una situación de muchos usuarios que acceden al sistema y solicitan páginas, envían datos, etc.

La prueba estudia el desempeño del sistema conforme los usuarios aumentan desde uno hasta veinte. Cuatro administradores y dieciséis profesores simulados ingresan al sistema en intervalos regulares. Cada uno solicita páginas y realiza operaciones que involucran lectura y escritura en la base de datos. Una vez que están logeados el total de veinte usuarios, éstos se mantienen trabajando concurrentemente por algunos minutos hasta que la prueba dure treinta minutos.

Inicialmente el sistema está pensado para una concurrencia de usuarios baja (2 a 4 usuarios). El objetivo de esta prueba es determinar la variación de tiempo de respuesta del sistema conforme aumenta el número de usuarios ya que en un futuro el sistema podría requerir una mayor concurrencia de usuarios. Es necesario que el tiempo de respuesta no aumente demasiado con relación al número de usuarios logrados.

Test execution parameters:
Test status: finished
Test started at: 28/11/2007 11:50:19 a.m.
Test finished at: 28/11/2007 12:20:59 p.m.
Scenario name: siidcb.wps
Test run comment:
Test executed by: Salomón Ramírez
Test executed on: CIRUELA
Test duration: 0:30:00
Virtual users: 1 - 20

Summary

Profile	Sessions performed	Sessions with errors	Pages performed	Pages with errors	Hits performed	Hits with errors	Total KBytes sent	Total KBytes received
Profesor	111	0	3,636	0	13,458	0	6,001	193,493
Admin	55	0	5,072	0	28,440	0	12,703	305,266
Total	166	0	8,708	0	41,898	0	18,704	498,759

Number of active users

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00
Profesor	1	1	1	2	5	8	10	10	10	10
Admin	2	5	8	10	10	10	10	10	10	10
Total	3	6	9	12	15	18	20	20	20	20

Sessions

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	1	3	2	2	8	13	20	21	20	21	111
Admin	0	1	3	6	8	7	7	9	7	7	55
Total	1	4	5	8	16	20	27	30	27	28	166

Sessions per second

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	0.006	0.02	0.01	0.01	0.04	0.07	0.11	0.12	0.11	0.12	0.06
Admin	0	0.006	0.02	0.03	0.04	0.04	0.04	0.05	0.04	0.04	0.03
Total	0.006	0.02	0.03	0.04	0.09	0.11	0.15	0.17	0.15	0.16	0.09

Pages

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	62	66	69	88	273	472	632	652	667	655	3,636
Admin	64	252	420	607	617	614	626	617	628	627	5,072
Total	126	318	489	695	890	1,086	1,258	1,269	1,295	1,282	8,708

Hits

Profile	0:00:00 -	0:03:00 -	0:06:00 -	0:09:00 -	0:12:00 -	0:15:00 -	0:18:00 -	0:21:00 -	0:24:00 -	0:27:00 -	Total

	0:03:00	0:06:00	0:09:00	0:12:00	0:15:00	0:18:00	0:21:00	0:24:00	0:27:00	0:30:00	
Profesor	230	240	255	325	999	1,741	2,333	2,416	2,488	2,431	13,458
Admin	251	1,359	2,271	3,344	3,502	3,603	3,493	3,398	3,652	3,567	28,440
Total	481	1,599	2,526	3,669	4,501	5,344	5,826	5,814	6,140	5,998	41,898

Pages per second

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	0.34	0.37	0.38	0.49	1.52	2.62	3.51	3.62	3.71	3.64	2.02
Admin	0.36	1.4	2.33	3.37	3.43	3.41	3.48	3.43	3.49	3.48	2.82
Total	0.7	1.77	2.72	3.86	4.94	6.03	6.99	7.05	7.19	7.12	4.84

Hits per second

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	1.28	1.33	1.42	1.81	5.55	9.67	13.0	13.4	13.8	13.5	7.48
Admin	1.39	7.55	12.6	18.6	19.5	20.0	19.4	18.9	20.3	19.8	15.8
Total	2.67	8.88	14.0	20.4	25.0	29.7	32.4	32.3	34.1	33.3	23.3

Pages with errors

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	0	0	0	0	0	0	0	0	0	0	0
Admin	0	0	0	0	0	0	0	0	0	0	0
Total	0	0	0	0	0	0	0	0	0	0	0

Hits with errors

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profeso	0	0	0	0	0	0	0	0	0	0	0

r											
Admin	0	0	0	0	0	0	0	0	0	0	0
Total	0	0	0	0	0	0	0	0	0	0	0

KBytes sent

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	32.7	34.3	35.8	47.0	141	249	335	343	355	349	1,922
Admin	30.0	121	201	291	296	297	301	295	303	302	2,437
Total	62.7	156	237	338	437	546	636	637	658	651	4,359

KBytes received

Profile	0:00:00 - 0:03:00	0:03:00 - 0:06:00	0:06:00 - 0:09:00	0:09:00 - 0:12:00	0:12:00 - 0:15:00	0:15:00 - 0:18:00	0:18:00 - 0:21:00	0:21:00 - 0:24:00	0:24:00 - 0:27:00	0:27:00 - 0:30:00	Total
Profesor	2,882	2,893	2,911	4,295	11,579	21,811	28,724	29,856	31,572	31,582	168,105
Admin	2,729	11,747	20,251	28,673	30,113	32,963	32,188	30,409	33,328	33,031	255,432
Total	5,611	14,641	23,162	32,968	41,692	54,774	60,912	60,265	64,900	64,613	423,537

Desempeño general (Overall performance).

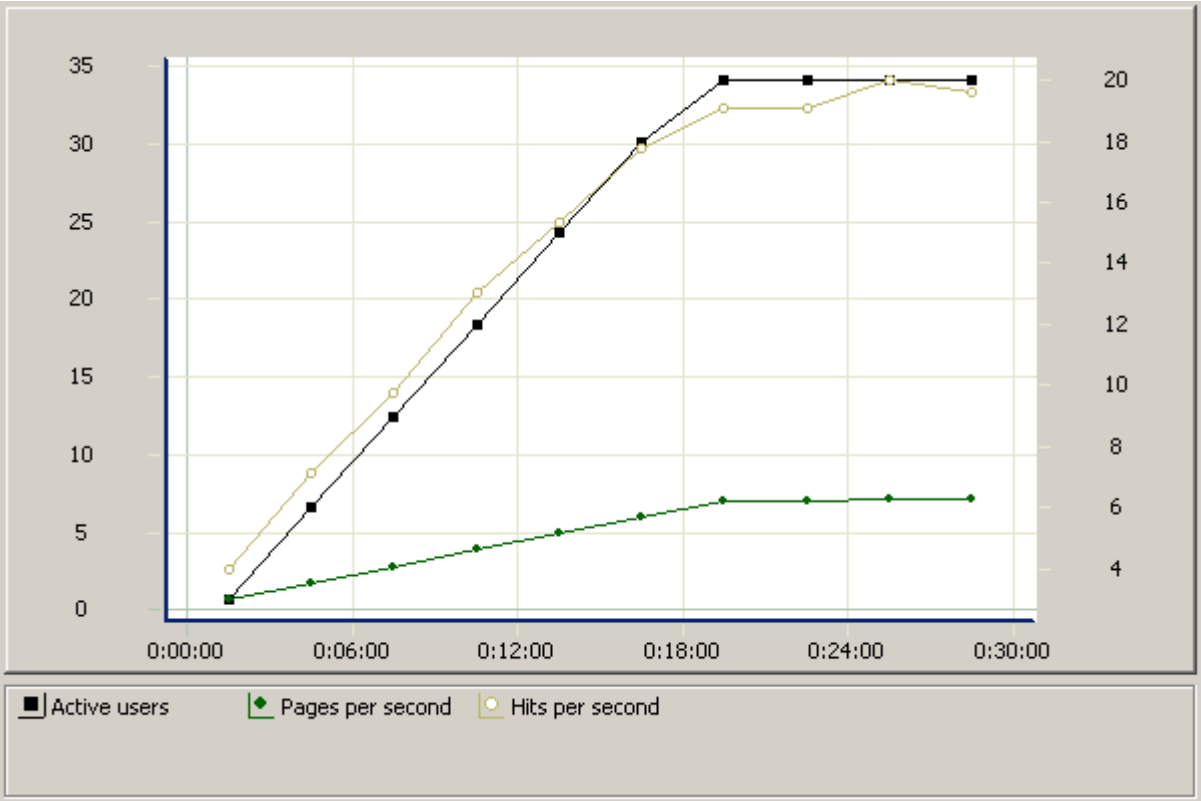


Figura 6—1 Gráfica de desempeño general durante la prueba.

Errores

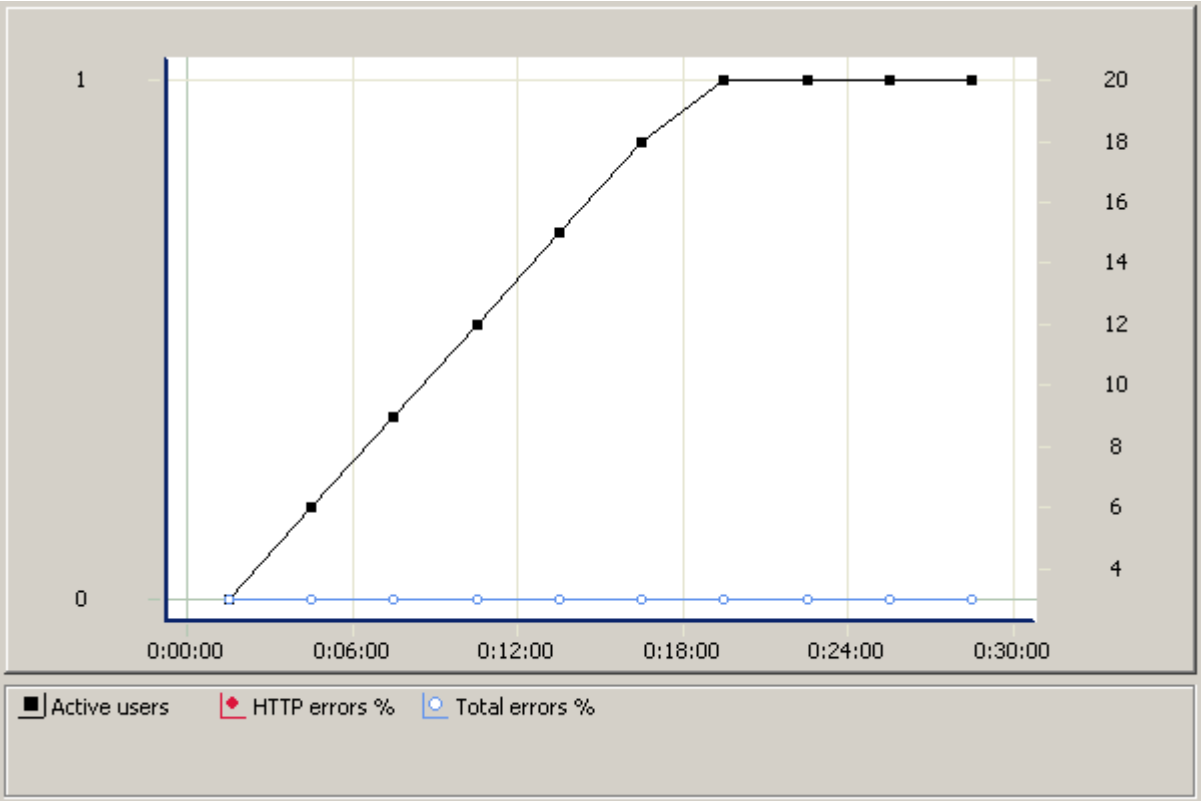


Figura 6—2 Gráfica de errores durante la prueba.

Ancho de banda promedio (Average bandwidth).

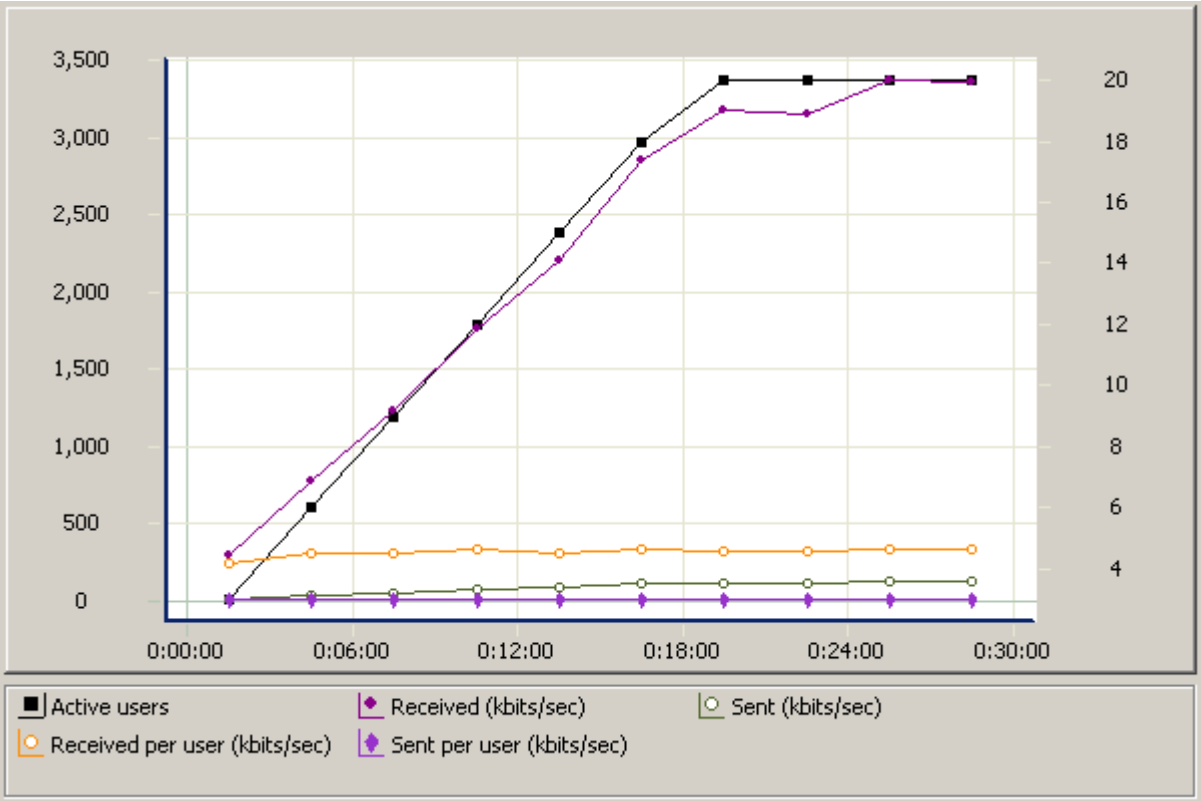


Figura 6—3 Gráfica de ancho de banda utilizado durante la prueba.

Desempeño.

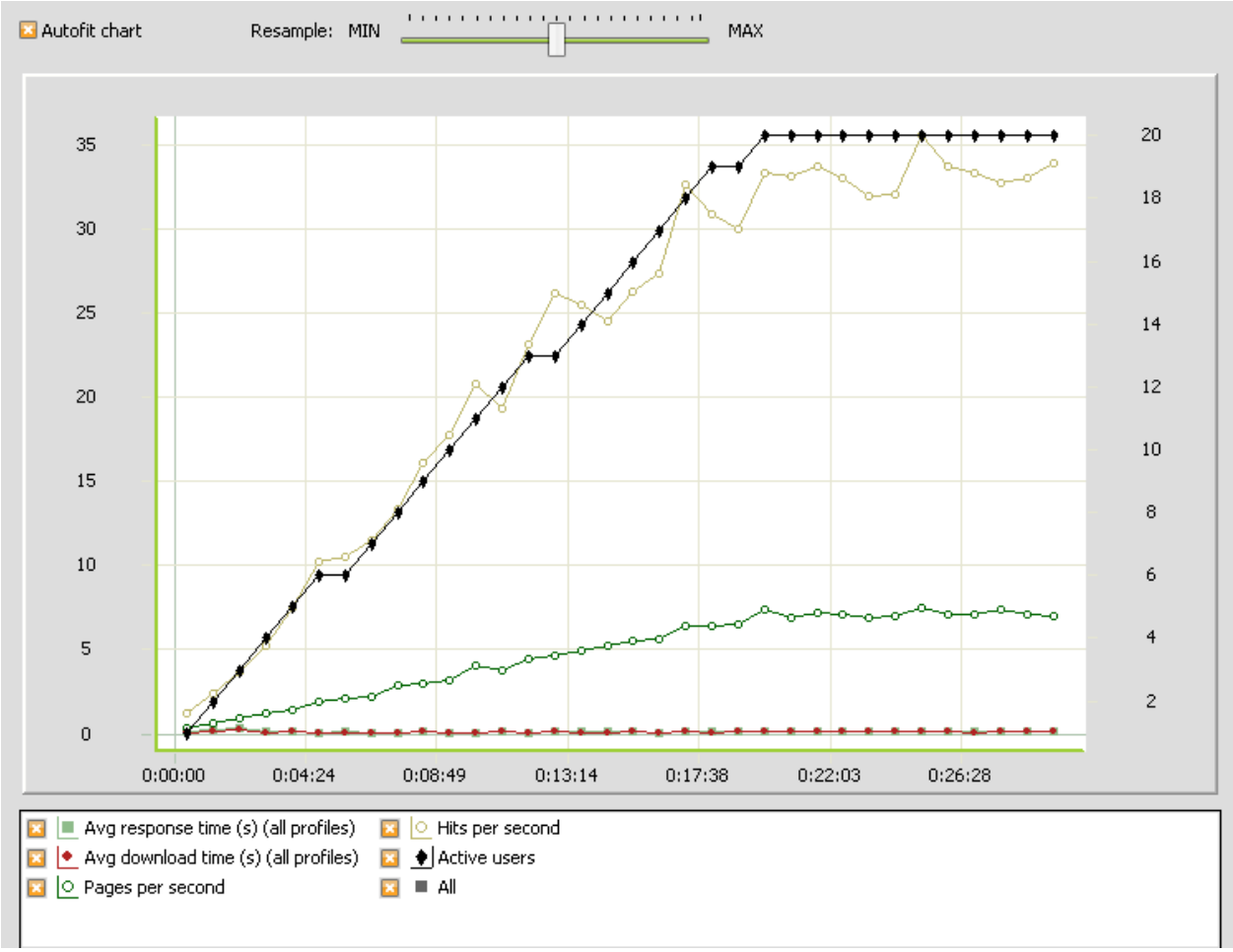


Figura 6—4 Gráfica con mayor detalle del desempeño durante la prueba.

Ancho de banda.

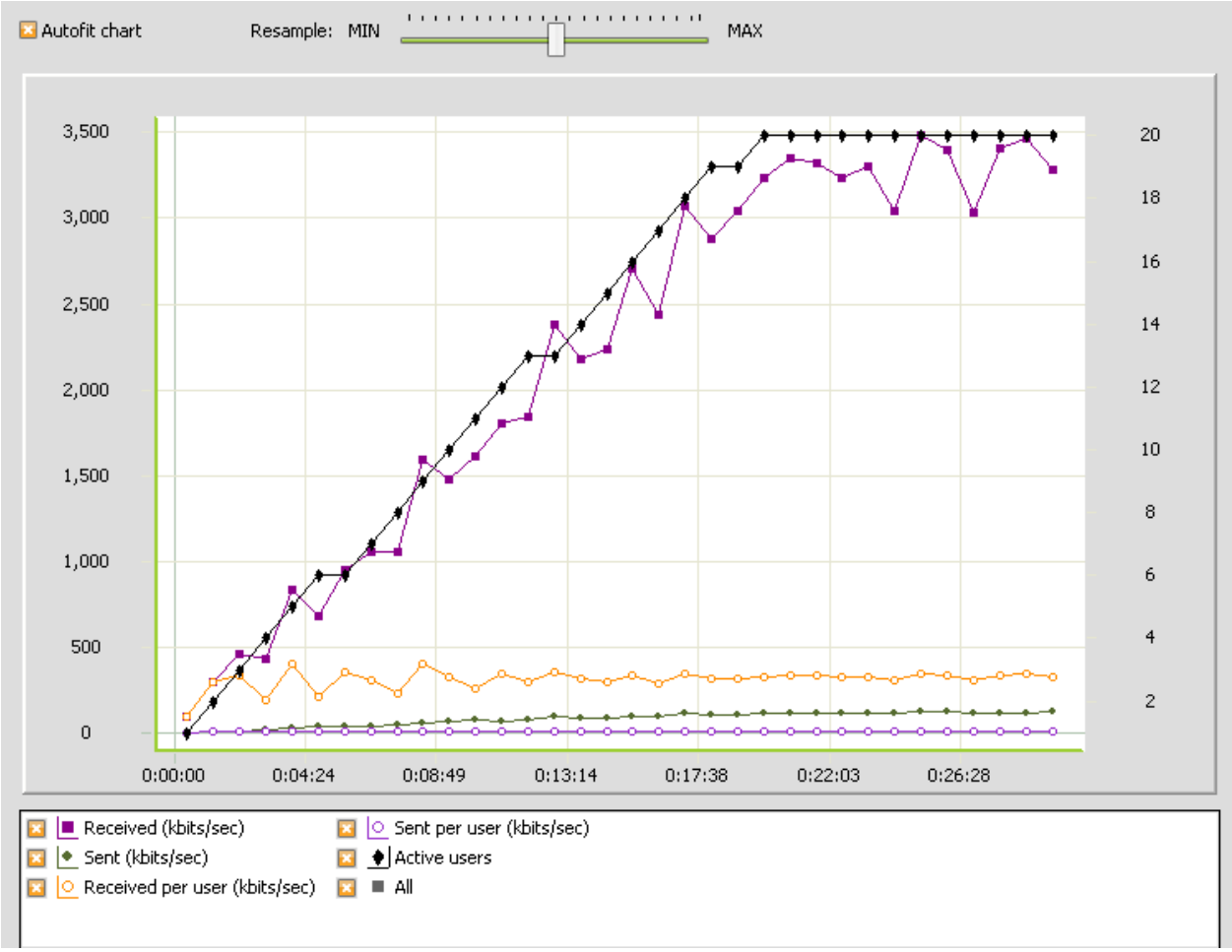


Figura 6—5 Gráfica con mayor detalle del ancho de banda utilizado durante la prueba.

Resultados de las pruebas.

En las gráficas puede observarse que el desempeño de la aplicación disminuye muy poco con relación al incremento de usuarios concurrentes. Este bajo impacto es un buen indicador del poco consumo de recursos de la aplicación por cada usuario que accede y hace solicitudes al sistema. En un futuro la aplicación podría aceptar una cantidad de usuarios concurrentes mucho mayor a veinte si es necesario.

6.2 Casos de prueba.

Los siguientes casos de prueba han sido seleccionados para las pruebas ya que son representativos de la funcionalidad general del sistema y porque son casos muy frecuentes en el uso cotidiano de la aplicación.

- **Ingresar al sistema.**

  Propósito:
  Acceder al menú de acciones de SIIDCB.

  Prerrequisitos:
  Solicitar la URL de la página de acceso.

  Datos de prueba:
  username=”salo”, password=”123456”

  Pasos:
  - Ingresar el nombre de usuario y contraseña en el formulario.
  - Presionar el botón “Iniciar sesión”.
  - Ver la página de inicio con el menú correspondiente al tipo de cuenta.

  Resultados:
  El sistema valida correctamente los datos y muestra la página de inicio.

  El sistema verifica que se hayan ingresado ambos campos antes de consultar la base de datos.

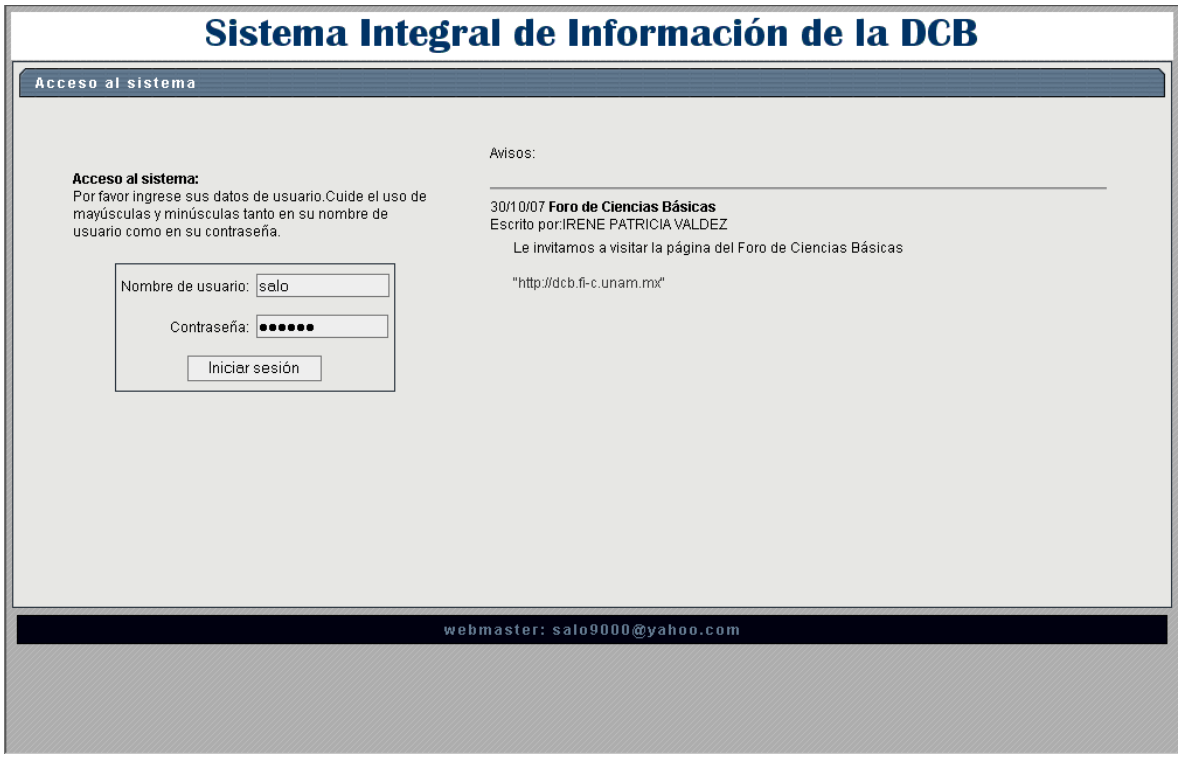


Figura 6—6 Página de ingreso a SIIDCB.

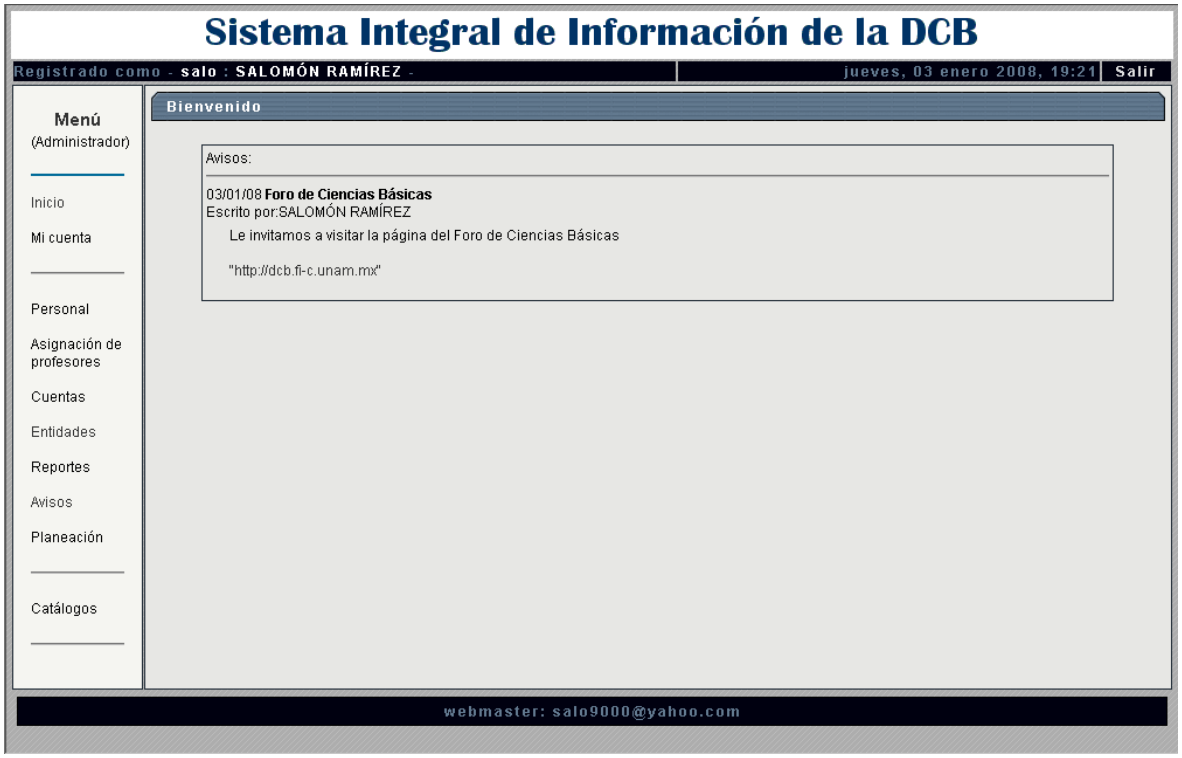


Figura 6—7 Página de inicio.

- **Consultar datos de un miembro del personal.**

Propósito:

Ver datos personales registrados en el sistema.

Prerrequisitos:

Haber iniciado sesión en SIIDCB e ingresado al módulo de personal.

Datos de prueba:

texto_de_busqueda=”salo”

Pasos:

- Ingresar el texto de búsqueda en el campo mostrado.
- Presionar el botón “Buscar”.
- Ver los resultados de búsqueda.
- Dar clic en la liga “Consultar” que corresponda al registro que se busca.
- Ver los datos personales mostrados.

Resultados:

El sistema realiza una búsqueda en los registros de personal con base en el texto proporcionado y muestra los que coinciden con él. Al dar clic en “Consultar” para uno de los resultados, se muestran los datos personales de éste.

El resultado es el esperado. Se muestran los datos que se requieren.

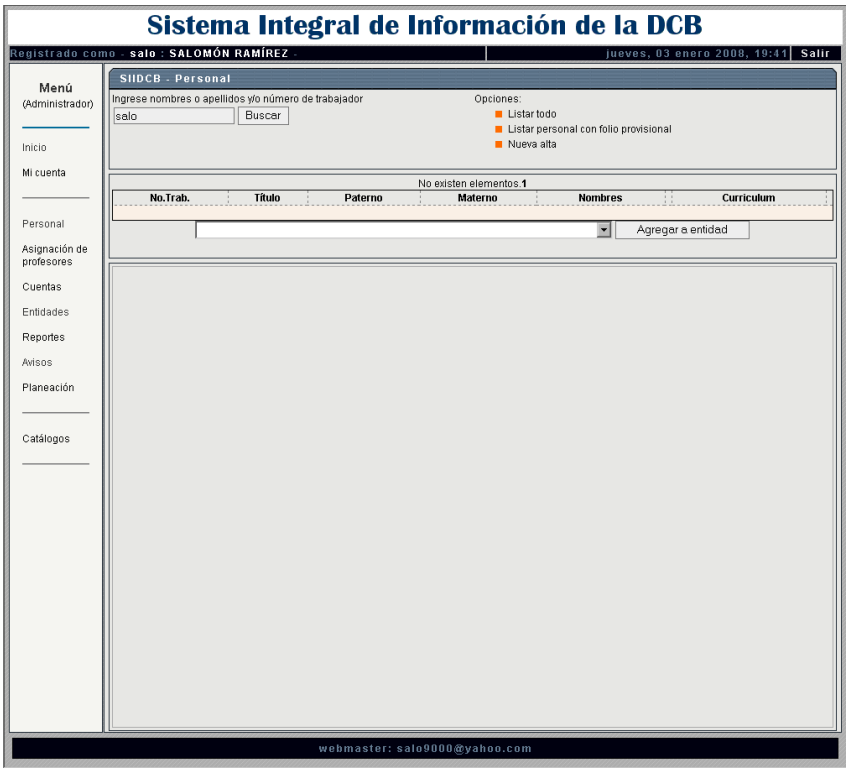


Figura 6—8 Página del módulo de personal e ingreso de patrón de búsqueda.

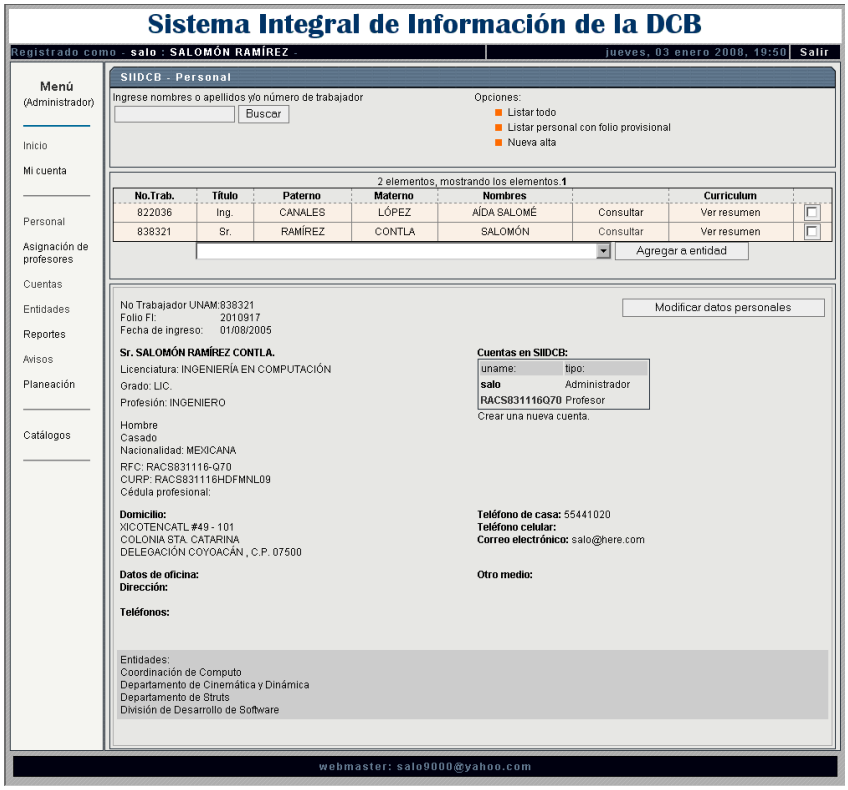


Figura 6—9 Resultados de búsqueda y despliegue de datos de un registro de personal.

- **Modificar datos de un registro de personal.**

Propósito:

Realizar cambios en los datos de un registro de personal.

Prerrequisitos:

Consultar los datos de un miembro del personal.

Datos de prueba:

calle=”Enrique Rébsamen #104”
colonia=”Narvarte Poniente”
cp=”03100”
delegación=”Benito Juárez”

Pasos:

- Presionar el botón “Modificar datos personales”.
- Ver el formulario con los datos personales.
- Hacer cambios en los campos editables.
- Dar clic en el botón “Guardar”.
- Ver resultados de la operación.

Resultados:

El sistema guarda el registro editado y muestra las opciones de “hacer más cambios” o “regresar a la página de personal”.

El resultado es el esperado. Los cambios son almacenados en la base de datos.

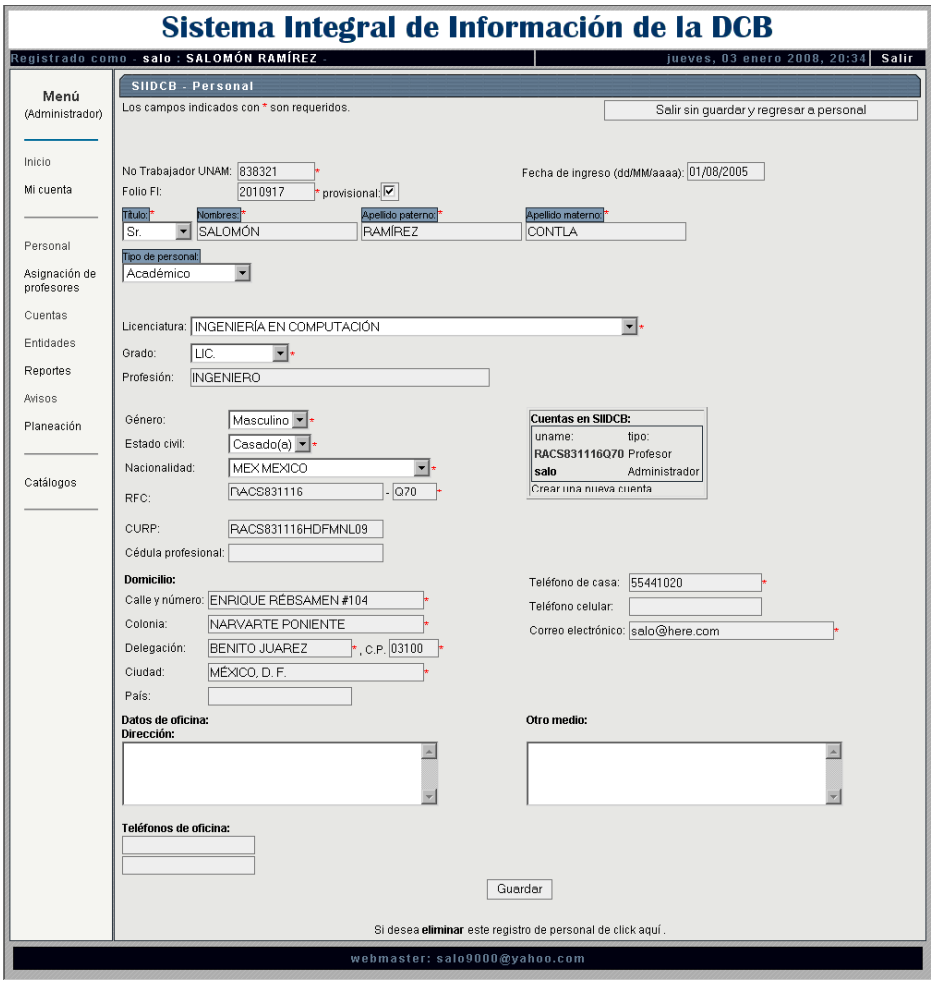


Figura 6—10 Formulario para realizar cambios de datos personales.

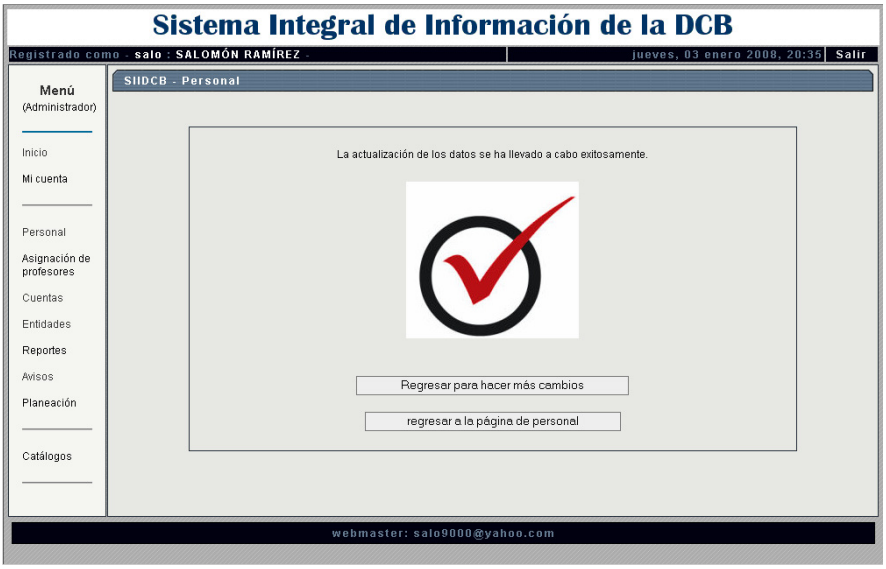


Figura 6—11 Resultados de la operación de modificación de datos personales.

- **Alta de un nuevo registro de personal.**

Propósito:

Guardar en la base de datos un nuevo registro de personal.

Prerrequisitos:

Inicio de sesión como administrador, ingreso al módulo de personal.

Datos de prueba:

calle=”Enrique Rébsamen #104”
colonia=”Narvarte Poniente”
cp=”03100”
delegación=”Benito Juárez”

Pasos:

- Dar clic en la liga “Nueva alta” que se muestra en la página de personal.
- Ver el formulario con los datos personales.
- Ingresar todos los datos marcados como requeridos.
- Dar clic en el botón “Guardar”.
- Ver resultados de la operación.

Resultados:

El sistema verifica que todos los campos requeridos sean ingresados y valida los datos de entrada antes de proceder al guardado del registro.

Pasada la validación, el sistema guarda el nuevo registro y muestra las opciones de “hacer más cambios” o “regresar a la página de personal”.

El resultado es el esperado. El nuevo registro es guardado en la base de datos.

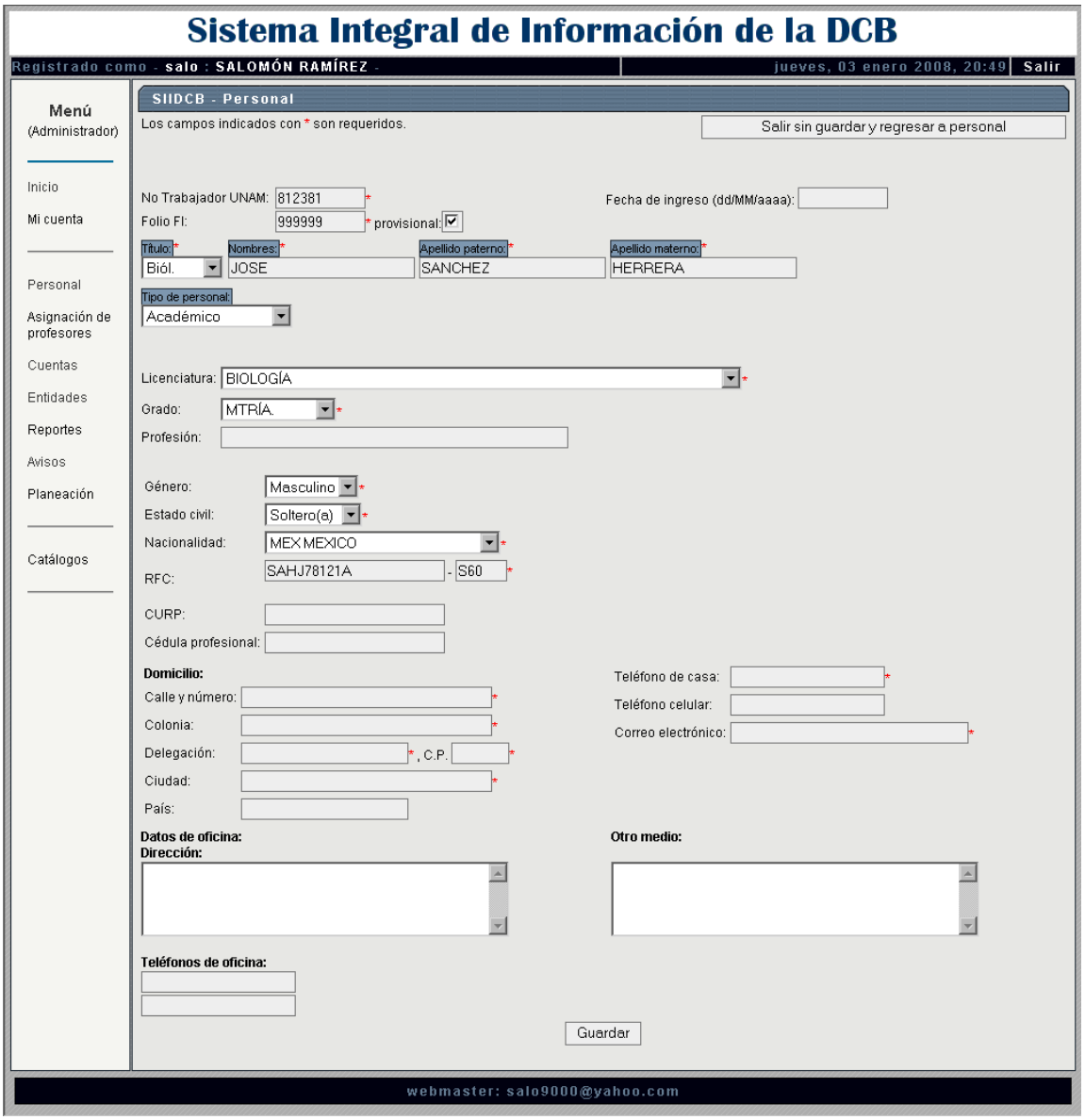


Figura 6—12 Formulario para nuevo guardar registro de personal.

Figura 6—13 Aviso de falta de datos requeridos.

- **Creación de una nueva cuenta de acceso.**

Propósito:

Proporcionar a un registro de personal una cuenta que permita el acceso a un usuario asociado con el registro.

Prerrequisitos:

Inicio de sesión como administrador, ingreso al módulo de personal y consulta de datos de un registro de personal.

Datos de prueba:

username=”salomonrc”
password=”contrasena”

Pasos:

- En la página de consulta de datos personales ver las cuentas de usuario asociadas al registro.
- Dar clic en la liga “Crear una cuenta nueva”.
- Ingresar username y contraseña para la cuenta y confirmar la contraseña.
- Dar clic en el botón “Guardar”.
- Ver resultados de la operación.

Resultados:

El sistema verifica que no exista otra cuenta con el mismo username que el ingresado y que la contraseña y su confirmación coincidan.

Pasada la validación, el sistema crea la cuenta con el rol “profesor” y muestra los resultados de la operación.

El resultado es el esperado. La nueva cuenta para ingresar al sistema ha sido creada.

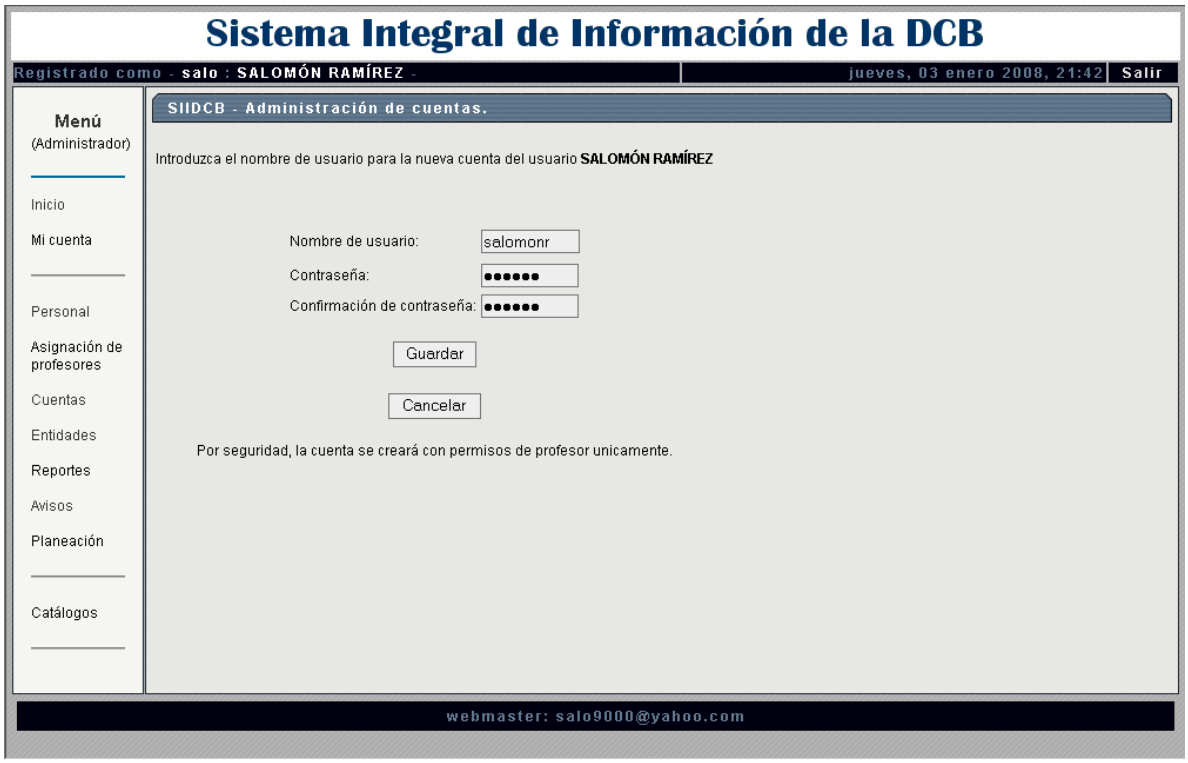


Figura 6—14 Formulario para creación de una nueva cuenta de acceso a SIIDCB.

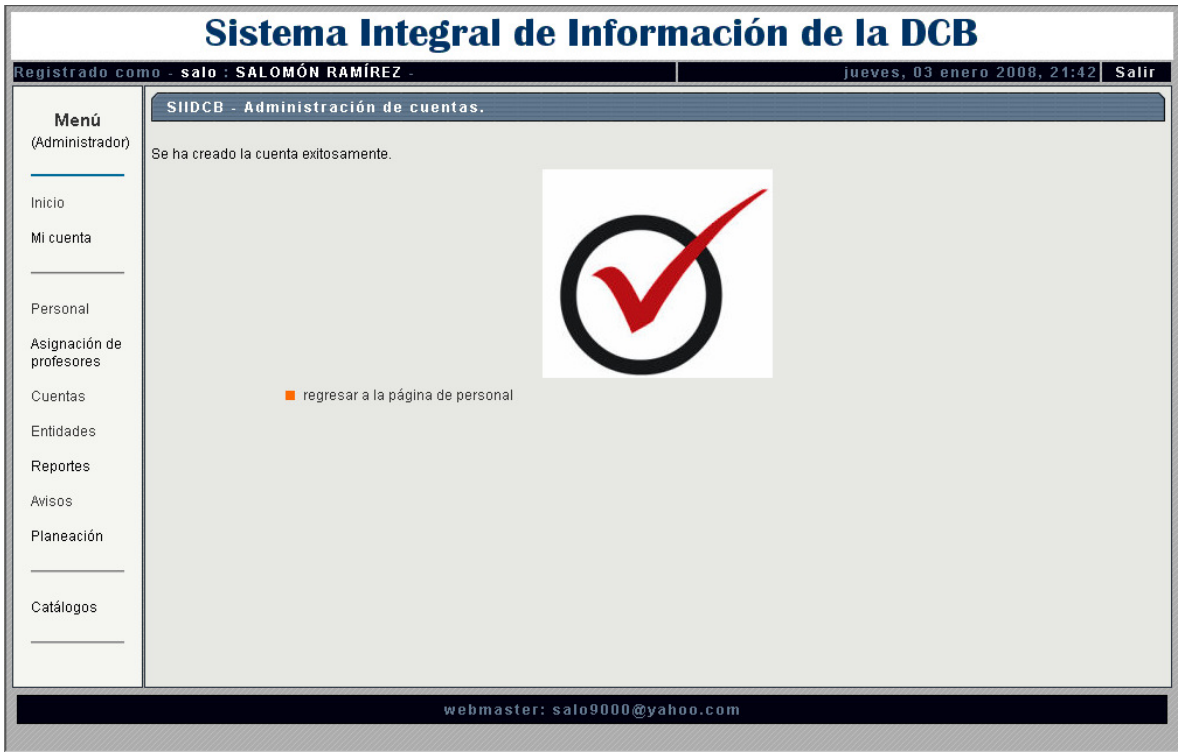


Figura 6—15 Muestra del resultado de la creación de cuenta.

- **Consulta de datos de una cuenta de acceso existente.**

Propósito:

Ver los permisos y datos generales de una cuenta de acceso a SIIDCB.

Prerrequisitos:

Inicio de sesión como administrador, ingreso al módulo de cuentas.

Datos de prueba:

patron_de_busqueda=”salo”

Pasos:

- En la página del módulo de cuentas se ingresa el patrón de búsqueda en el campo mostrado.
- Dar clic al botón “Buscar”.
- Ver los resultados de la búsqueda en la sección de la página indicada como “Resultados de búsquedas y listados”.
- Dar clic en la cuenta que se desea consultar mostrada en la lista.
- Ver los datos de la cuenta en la sección de la página indicada como ”Datos de la cuenta seleccionada”.

Resultados:

El sistema realiza una búsqueda basada en el texto ingresado y muestra las cuentas coincidentes con éste y una vez seleccionada una de las cuentas, se muestran los datos solicitados.

El resultado es el esperado. Los datos requeridos son mostrados en pantalla.

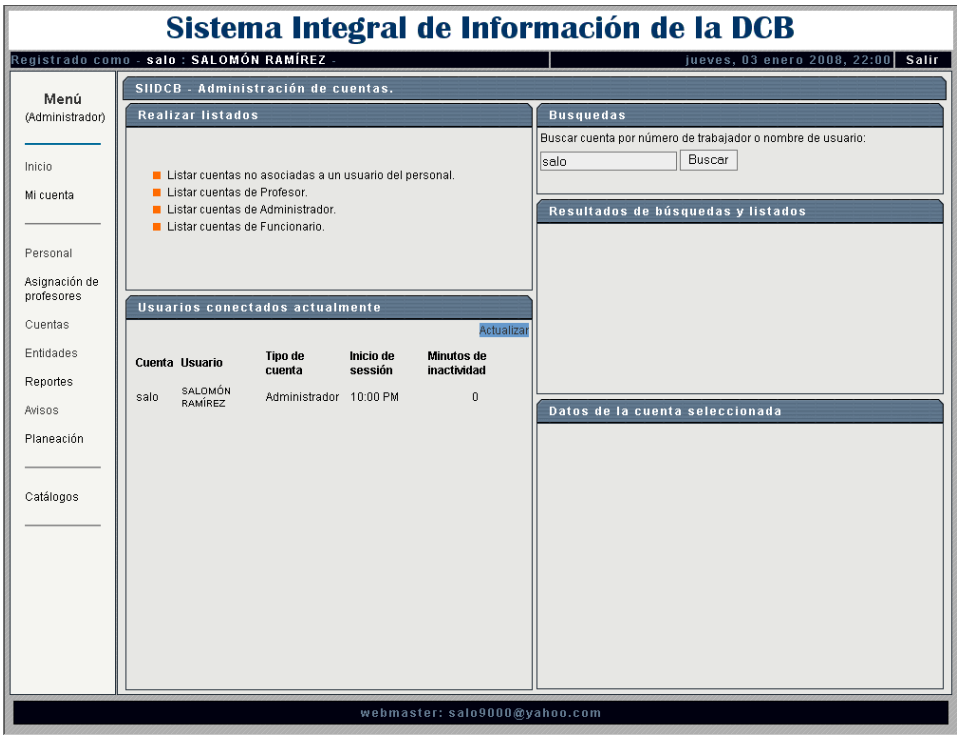


Figura 6—16 Página inicial del módulo de cuentas. El patrón de búsqueda ha sido ingresado en el campo.

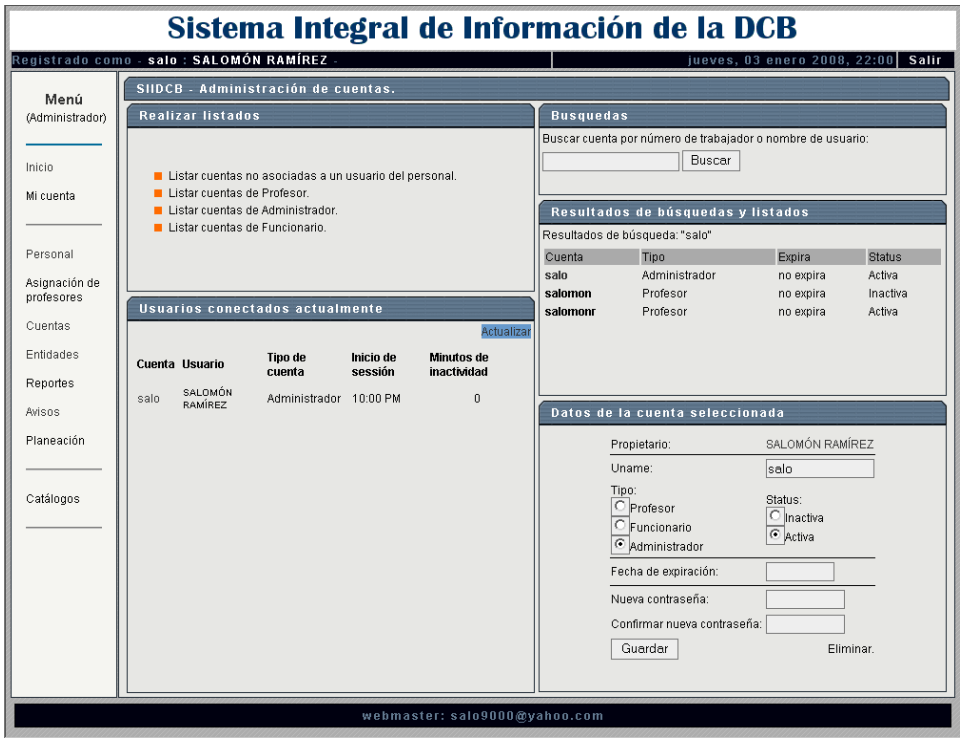


Figura 6—17 Despliegue de los datos de la cuenta seleccionada de entre los resultados obtenidos.

- **Modificación de una cuenta de acceso existente.**

Propósito:

Cambiar datos de una cuenta de acceso a SIIDCB (tipo de cuenta, nombre de usuario, status, contraseña, etc.).

Prerrequisitos:

Inicio de sesión como administrador, consulta de datos de una cuenta.

Datos de prueba:

Botones de radio: seleccionar “Administrador”

Pasos:

- Modificar los datos de la cuenta en el formulario mostrado en la sección de la página indicada como ”Datos de la cuenta seleccionada”.
- Dar clic al botón “Guardar”.
- Ver el resultado de la operación en la misma sección.

Resultados:

Se ha cambiado el tipo de cuenta de “Profesor” a “Administrador” exitosamente. El resultado es el esperado y el resultado se muestra en pantalla.

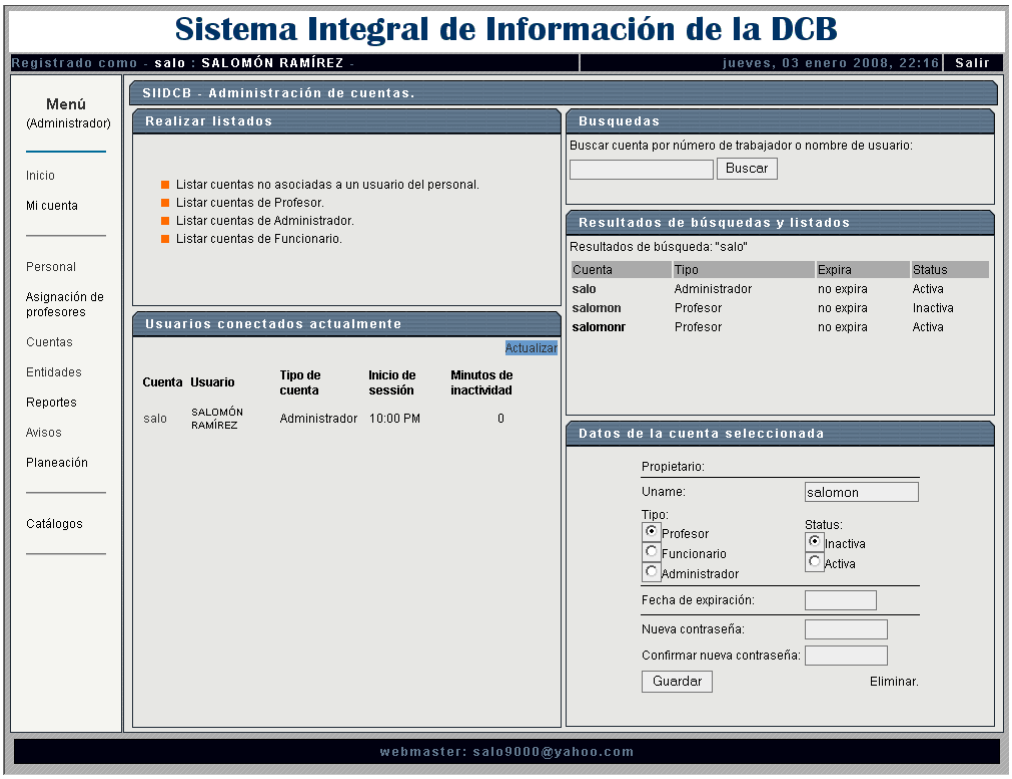


Figura 6—18 Formulario con datos de cuenta.

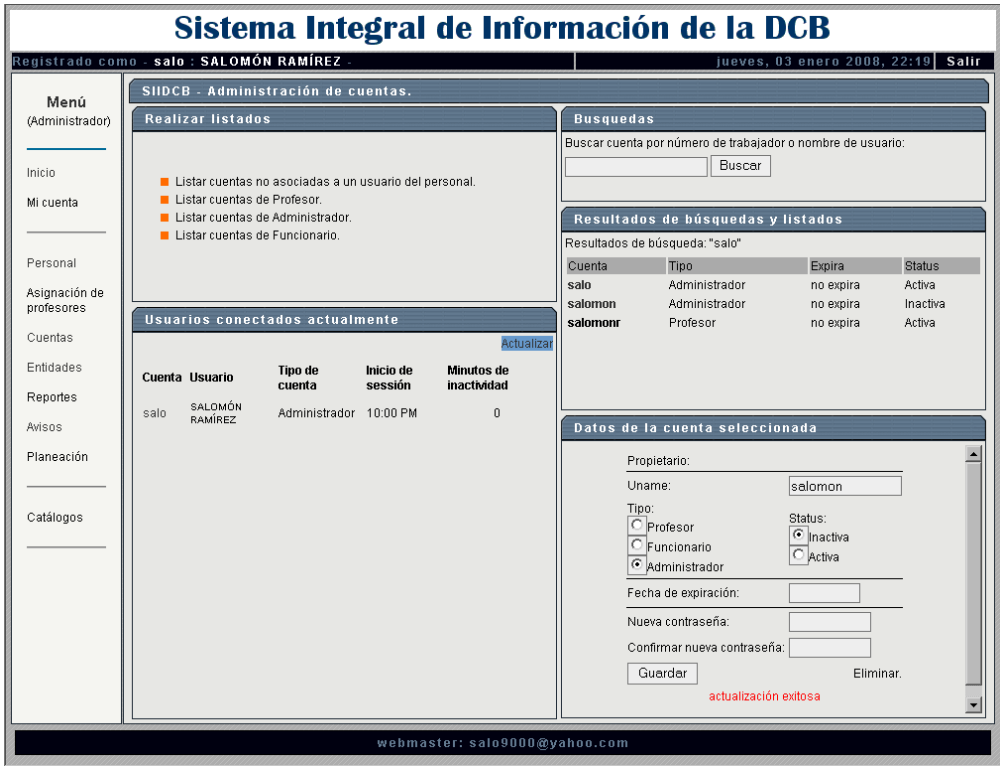


Figura 6—19 Formulario de modificación de cuenta mostrando el resultado de los cambios.

- **Consulta de los datos de una entidad de la DCB.**

Propósito:

Ver los datos de una de las entidades que conforman la organización de la División.

Prerrequisitos:

Inicio de sesión como administrador, ingreso al módulo de entidades.

Datos de prueba:

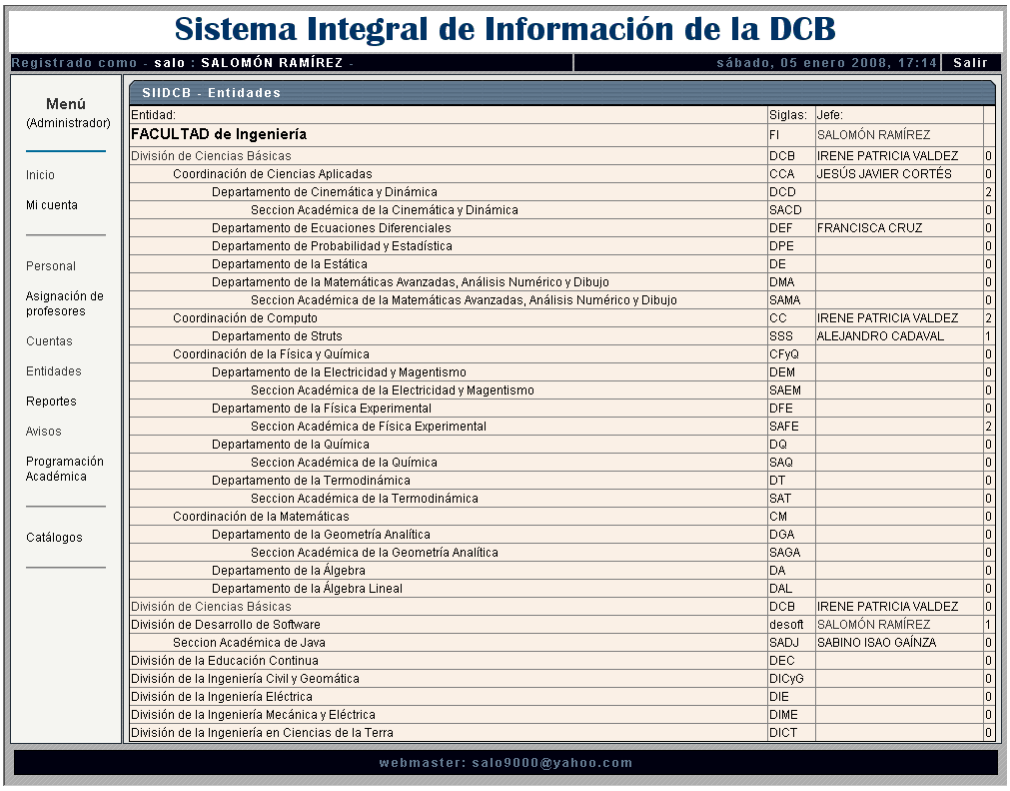
Ninguno.

Pasos:

- Ver la lista de entidades desplegada en la página del módulo de entidades.
- Dar clic al nombre de una de las entidades.
- Ver los datos en la página desplegada como respuesta.

Resultados:

Se han desplegado los datos de la entidad solicitada. El resultado es el esperado.



Entidad:	Siglas:	Jefe:	
FACULTAD de Ingeniería	FI	SALOMÓN RAMÍREZ	
División de Ciencias Básicas	DCB	IRENE PATRICIA VALDEZ	0
Coordinación de Ciencias Aplicadas	CCA	JESÚS JAVIER CORTÉS	0
Departamento de Cinemática y Dinámica	DCD		2
Seccion Académica de la Cinemática y Dinámica	SACD		0
Departamento de Ecuaciones Diferenciales	DEF	FRANCISCA CRUZ	0
Departamento de Probabilidad y Estadística	DPE		0
Departamento de la Estática	DE		0
Departamento de la Matemáticas Avanzadas, Análisis Numérico y Dibujo	DMA		0
Seccion Académica de la Matemáticas Avanzadas, Análisis Numérico y Dibujo	SAMA		0
Coordinación de Computo	CC	IRENE PATRICIA VALDEZ	2
Departamento de Struts	SSS	ALEJANDRO CADAVAL	1
Coordinación de la Física y Química	CFyQ		0
Departamento de la Electricidad y Magentismo	DEM		0
Seccion Académica de la Electricidad y Magentismo	SAEM		0
Departamento de la Física Experimental	DFE		0
Seccion Académica de Física Experimental	SAFE		2
Departamento de la Química	DQ		0
Seccion Académica de la Química	SAQ		0
Departamento de la Termodinámica	DT		0
Seccion Académica de la Termodinámica	SAT		0
Coordinación de la Matemáticas	CM		0
Departamento de la Geometría Analítica	DGA		0
Seccion Académica de la Geometría Analítica	SAGA		0
Departamento de la Álgebra	DA		0
Departamento de la Álgebra Lineal	DAL		0
División de Ciencias Básicas	DCB	IRENE PATRICIA VALDEZ	0
División de Desarrollo de Software	desoft	SALOMÓN RAMÍREZ	1
Seccion Académica de Java	SADJ	SABINO ISAO GAÍNZA	0
División de la Educación Continua	DEC		0
División de la Ingeniería Civil y Geomática	DICyG		0
División de la Ingeniería Eléctrica	DIE		0
División de la Ingeniería Mecánica y Eléctrica	DIME		0
División de la Ingeniería en Ciencias de la Terra	DICT		0

Figura 6—20 Listado mostrado al ingresar al módulo de entidades.

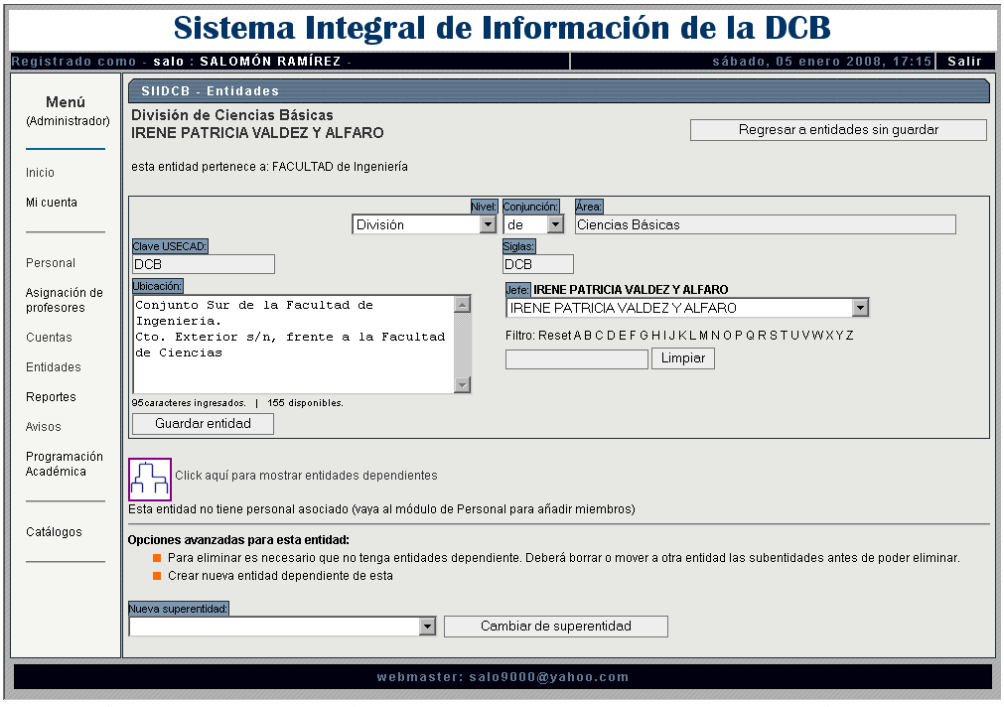


Figura 6—21 Despliegue de los datos de una de las entidades.

- **Modificación de los datos de una entidad de la DCB.**

Propósito:

Cambiar algunos de los datos de una de las entidades registradas.

Prerrequisitos:

Inicio de sesión como administrador, consulta de datos de una de las entidades.

Datos de prueba:

clave_usecad=”FIDCB”.

Pasos:

- Ver los datos desplegados.
- Realizar los cambios en los campos deseados.
- Presionar el botón “Guardar entidad”.
- Ver el resultado de la operación.

Resultados:

El sistema ha guardado en la base de datos los cambios indicados. El resultado es el esperado.

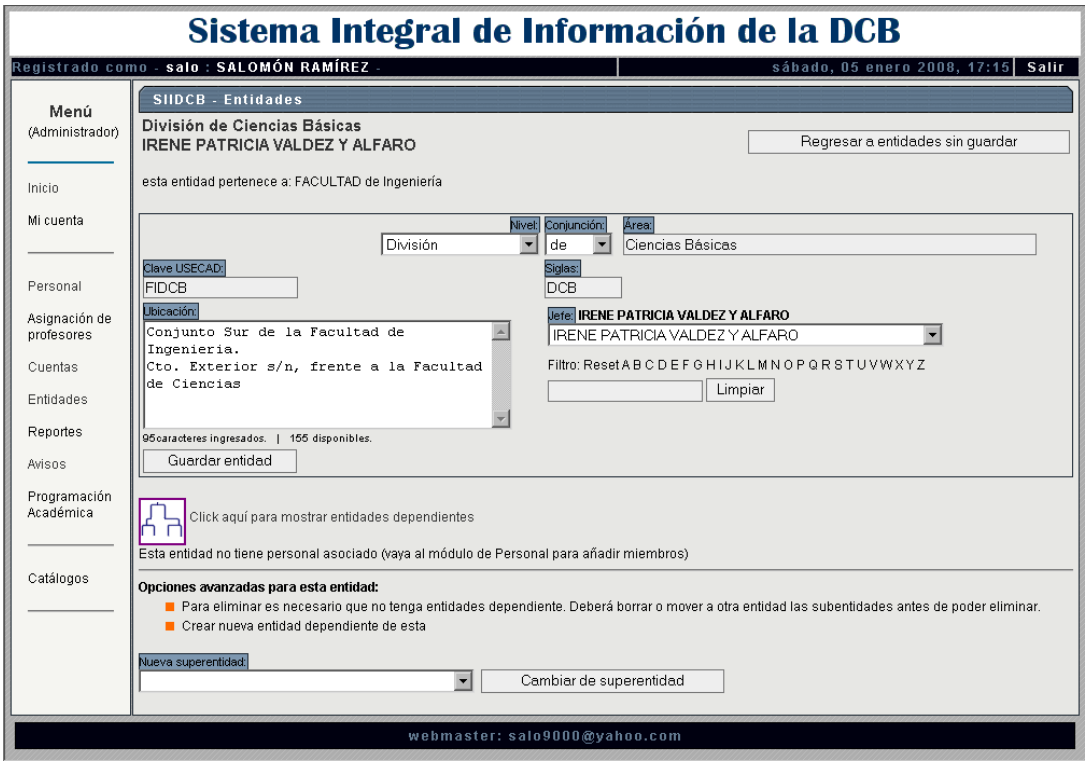


Figura 6—22 Despliegue de los datos de la entidad solicitada.

Figura 6—23 Despliegue del resultado de los cambios en la entidad.

Conclusiones.

Durante el desarrollo de un sistema computacional es de gran importancia la comunicación constante con el cliente. En el presente trabajo las entrevistas para el reconocimiento del entorno y problemática fueron indispensables y de gran ayuda a lo largo de todo el proceso de desarrollo. Una de las experiencias más importantes en el desarrollo de SIIDCB fue percatarme de esto porque siempre, en cualquier proyecto de software, será necesario el contacto con el cliente para llegar a acuerdos bien definidos y llegar al objetivo deseado.

En SIIDCB he utilizado e integrado diversas tecnologías J2EE. Tuve que recurrir a las documentaciones y bibliografía de cada una, y aunque esto requirió de mucho tiempo, me permitió trabajar con un alto entendimiento de lo que programaba y así crear una aplicación robusta. Todo el software utilizado en el desarrollo de SIIDCB es software libre, lo cual resultó muy conveniente ya que no se contaba con presupuesto para la compra de licencias de software propietario. La experiencia de haber utilizado únicamente software libre me ha servido para conocer más el ambiente propio de los proyectos libres, donde la documentación de los creadores, así como la colaboración de los mismos y de los usuarios en los foros en internet resultan una gran fuente de conocimiento acerca del desarrollo con este tipo de software.

Java es una de las plataformas de programación existentes más importantes y más completas. Es posible consultar toda la documentación de las API's de Sun en su sitio web. También ofrece muchas herramientas para aplicaciones de alto nivel tales como aplicaciones web como las API's Servlet y JSP, y los marcos de trabajo para *webapps* creados alrededor de Java se actualizan rápidamente para facilitar a los desarrolladores la elaboración de sistemas flexibles, robustos y modulares que puedan ser mantenidos por diferentes grupos de trabajo.

Con la utilización de Hibernate para la persistencia de datos del sistema tuve presente el paradigma orientado a objetos en todo momento durante la codificación. He trabajado con diferentes paradigmas cuando he utilizado diferentes lenguajes y al utilizar el paradigma orientado a objetos la abstracción de los problemas, de los dominios y de sus elementos resulta sumamente fácil y muy natural. La experiencia obtenida con mi tesis me inclina aún más por este paradigma.

Gracias al uso de la versión 5.1 del sistema manejador de base de datos MySQL he aprendido a aprovechar las grandes ventajas esta nueva versión, ya que antes de ésta MySQL no contaba con características ACID que garantizan que las transacciones son procesadas correctamente. La necesidad de utilizar *triggers* que en otros sistemas he utilizado para asegurar consistencia no se presentó gracias a las características mencionadas y de este modo pude dedicar más tiempo de trabajo a la lógica de negocio y no a la de persistencia.

Tomcat es un servidor de aplicación muy ligero y totalmente portable entre plataformas. En las pruebas de volumen realizadas este servidor se comportó de manera muy estable de modo que será posible aumentar la carga de trabajo del servidor sin alguna dificultad. El comportamiento de este servidor será a partir de ahora un buen referente para mí cuando desarrolle otras aplicaciones web.

Para crear SIIDCB utilicé herramientas CASE y aprendí que es importante conocerlas y estudiarlas ya que son precisamente software para ingeniería de software y ayudan a plasmar en documentos las ideas que uno como codificador frecuentemente maneja de manera muy abstracta. También debemos conocer un buen IDE porque ayuda a integrar todo con lo que desarrollamos el sistema; en mi caso utilicé Eclipse y me pareció muy bueno. Con todas estas herramientas se simplifica el manejo de código, las pruebas, la visualización la estructura de archivos de la aplicación, el compilado y otras tareas pueden volverse una gran molestia si se llevan a cabo manualmente, además de que ello repercute en el desperdicio de un recurso muy importante en todo proyecto: el tiempo.

Un punto importante durante el desarrollo de cualquier sistema mediano o grande y que consideré crucial para este trabajo es la organización modular del mismo. Al estudiar el patrón de desarrollo MVC observé las ventajas de su implementación en cuanto a modularidad y organización de código que proporciona. MVC es un patrón muy utilizado para este tipo de aplicaciones y sé que en el mercado de trabajo su entendimiento es un requisito.

Durante todo el proceso de elaboración de este trabajo he reafirmado y aplicado conceptos de ingeniería que estudié a lo largo de la carrera en distintas ocasiones. La ingeniería de software fue puesta en marcha totalmente y eso me proporciona mucha satisfacción porque conocer el proceso completo de elaboración de un sistema computacional es una experiencia básica para todo ingeniero en computación y me ayudará mucho en proyectos futuros en los que me involucre. También el hecho de haber desarrollado una aplicación web es para mí una satisfacción personal.

Aplicaciones y mejoras a futuro.

SIIDCB puede ser extendido para manejar otro tipo de información relacionado al personal de la DCB, tal como nombramientos del personal y otras cuestiones administrativas complejas. Las bases del sistema han sido elaboradas para añadir funcionalidades. Al mismo tiempo puede ser actualizado en cuanto a las tecnologías que utiliza. Existen varios marcos de trabajo que están ganando importancia y que utilizan el patrón MVC (Spring, Struts2, JSF, etc.). Varios de estos marcos pueden trabajar junto con Struts sin remplazarlo, así que no es necesario empezar de cero. Una recomendación es esperar a que exista documentación suficiente y las versiones de estos marcos sean estables para evitar dificultades.

Anexos y documentación del sistema.

Anexo A. Definiciones de casos de uso

Nombre:	Ingresar a SIIDCB.
Descripción: Procedimiento para iniciar una sesión en la aplicación.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Poseer un nombre de usuario y contraseña.	
Flujo Normal: 1. El actor proporciona su nombre de usuario y contraseña para su verificación. 2. El sistema verifica la validez de los datos proporcionados. 3. El sistema envía al actor la página de inicio con el menú correspondiente.	
Flujo Alternativo: Si los datos de nombre y contraseña son incorrectos, el sistema muestra la página de ingreso nuevamente indicando que se han ingresado datos inválidos.	

Nombre:	Autenticación.
Descripción: Proceso que verifica si la sesión de un usuario es válida y autoriza el acceso a un recurso solicitado.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Ninguna.	
Flujo Normal: 1. El actor solicita un recurso del sistema. 2. El sistema verifica si su sesión es válida (ha ingresado nombre de usuario y contraseña) y no ha expirado. 3. El sistema envía al actor el recurso solicitado.	
Flujo Alternativo: Si el actor no ha ingresado al sistema nombre de usuario y contraseña válidos, se muestra un aviso de que es necesario hacerlo para acceder al recurso solicitado. También se muestra una liga a la página de ingreso al sistema.	

Nombre:	Entrar a módulo de catálogos.
Descripción: Permite ver los catálogos del sistema que pueden ser editados.	
Actores: Administrador.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita el módulo de catálogos. 2. El sistema devuelve como respuesta un menú con las opciones del módulo.	
Flujo Alternativo:	

Nombre:	Ver catálogo.
Descripción: Permite ver un catálogo del sistema: nombre de los campos y valores.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de catálogos.	
Flujo Normal: 1. Desde el menú de catálogos, el actor da clic en la liga hacia un catálogo. 2. El sistema muestra el catálogo.	
Flujo Alternativo:	

Nombre:	Modificar registro de catálogo.
Descripción: Permite hacer modificaciones en uno de los registros del catálogo.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de catálogos.	
Flujo Normal: 1. En un listado de catálogos el actor modifica el valor de un campo o varios en un registro del catálogo. 2. El actor presiona el botón de guardar. 3. El sistema vuelve a mostrar el catálogo con el cambio registrado.	
Flujo Alternativo: Si existe un error al guardar el registro el sistema muestra un error.	

Nombre:	Eliminar registro de catálogo.
Descripción: Permite eliminar uno de los registros del catálogo.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de catálogos.	
Flujo Normal: 1. En un listado de catálogos el actor presiona el botón eliminar de un registro. 2. El sistema pide una confirmación de eliminación. 3. El actor confirma al dar clic al botón de confirmación. 4. El sistema vuelve a mostrar el catálogo actualizado.	
Flujo Alternativo: Si existe un error al eliminar el registro el sistema muestra un error.	

Nombre:	Añadir elemento al catálogo.
Descripción: Permite añadir un nuevo registro a un catálogo.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de catálogos.	
Flujo Normal: 1. En la página de listado de registros se muestra uno vació en el que pueden ingresarse datos para un nuevo registro. 2. El actor ingresa los datos en los campos correspondientes. 3. El actor presiona el botón guardar para ese nuevo registro. 4. El sistema vuelve a mostrar el catálogo actualizado.	
Flujo Alternativo: Si existe un error al guardar el nuevo registro el sistema muestra un error.	

Nombre:	Añadir elemento al catálogo.
Descripción: Permite añadir un nuevo registro a un catálogo.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de catálogos.	
Flujo Normal: 1. En la página de listado de registros se muestra uno vació en el que pueden ingresarse datos para un nuevo registro. 2. El actor ingresa los datos en los campos correspondientes. 3. El actor presiona el botón guardar para ese nuevo registro. 4. El sistema vuelve a mostrar el catálogo actualizado.	
Flujo Alternativo: Si existe un error al guardar el nuevo registro el sistema muestra un error.	

Nombre:	Entrar a módulo de entidades.
Descripción: Permite ver una página con opciones para el manejo de las entidades registradas en el sistema.	
Actores: Administrador.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita el módulo de entidades. 2. El sistema devuelve como respuesta un menú con las opciones del módulo.	
Flujo Alternativo:	

Nombre:	Listar entidades.
Descripción: Permite ver una página con los nombres de las entidades registradas en el sistema.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de entidades.	
Flujo Normal: 1. El actor solicita un listado de las entidades registradas. 2. El sistema muestra una lista de las entidades registradas.	
Flujo Alternativo:	

Nombre:	Consultar entidad.
Descripción: Permite ver los detalles de una de las entidades.	
Actores: Administrador.	
Precondiciones: Autenticación. Entrar a módulo de entidades.	
Flujo Normal: 1. El actor hace clic en una de las entidades de la lista de entidades. 2. El sistema muestra los datos de la entidad solicitada.	
Flujo Alternativo:	

Nombre:	Modificar entidad.
Descripción: Permite cambiar detalles de una entidad.	
Actores: Administrador.	
Precondiciones: Consultar entidad.	
Flujo Normal: 1. El actor cambia alguno(s) de los datos mostrados de una entidad. 2. El actor presiona el botón guardar. 3. El sistema guarda en la base de datos los cambios	
Flujo Alternativo: Si hay un problema al guardar en la base de datos el sistema muestra un aviso.	

Nombre:	Crear subentidad.
Descripción: Permite crear una entidad dependiente de otra.	
Actores: Administrador.	
Precondiciones: Consultar entidad.	
Flujo Normal: 1. El actor cambia alguno(s) de los datos mostrados de una entidad. 2. El actor presiona el botón guardar. 3. El sistema guarda en la base de datos los cambios	
Flujo Alternativo: Si hay un problema al guardar en la base de datos el sistema muestra un aviso.	

Nombre:	Eliminar entidad.
Descripción: Permite eliminar una entidad.	
Actores: Administrador.	
Precondiciones: Consultar entidad.	
Flujo Normal: 1. El actor selecciona la opción “eliminar entidad” de la página editar entidad. 2. El sistema pide una confirmación de eliminación. 3. El sistema guarda en la base de datos los cambios y muestra un mensaje del resultado de la operación.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos el sistema muestra un aviso.	

Nombre:	Eliminar personal de entidad.
Descripción: Permite quitar miembros del personal asociados a una entidad.	
Actores: Administrador.	
Precondiciones: Consultar entidad.	
Flujo Normal: 1. El actor selecciona los miembros del personal que desea quitar de un listado de personal de la entidad. 2. El actor presiona el botón “quitar de le entidad”. 3. El sistema guarda en la base de datos los cambios y muestra un mensaje del resultado de la operación.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos el sistema muestra un aviso.	

Nombre:	Entrar a módulo de avisos.
Descripción: Permite ver las opciones para manejo de avisos de sistema.	
Actores: Administrador.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita el módulo de avisos. 2. El sistema devuelve como respuesta un menú con las opciones del módulo.	
Flujo Alternativo: Si hay un problema al leer la base de datos el sistema muestra un aviso.	

Nombre:	Crear aviso.
Descripción: Permite crear un nuevo aviso de sistema.	
Actores: Administrador.	
Precondiciones: Entrar a módulo de avisos.	
Flujo Normal: 1. El actor presiona el botón “crear nuevo aviso”. 2. Llena el formulario con los datos del aviso y marca sus opciones. 3. Presiona el botón guardar. 4. El sistema guarda en la base de datos el nuevo aviso.	
Flujo Alternativo: Si hay un problema al guardar en la base de datos el sistema muestra un aviso.	

Nombre:	Listar avisos.
Descripción: Permite ver los avisos existentes.	
Actores: Administrador.	
Precondiciones: Entrar a módulo de avisos.	
Flujo Normal: 1. El sistema despliega una lista con los títulos de los avisos.	
Flujo Alternativo: Si hay un problema de lectura de la base de datos, el sistema muestra un aviso.	

Nombre:	Ver aviso.
Descripción: Permite ver los detalles de un aviso.	
Actores: Administrador.	
Precondiciones: Listar avisos.	
Flujo Normal: 1. El actor selecciona uno de los avisos listados. 2. El sistema devuelve una página que muestra los detalles completos del aviso.	
Flujo Alternativo: Si hay un problema de lectura de la base de datos, el sistema muestra un aviso.	

Nombre:	Eliminar aviso.
Descripción: Permite suprimir alguno de los avisos existentes.	
Actores: Administrador.	
Precondiciones: Ver aviso.	
Flujo Normal: 1. El actor da clic en el botón “eliminar este aviso”. 2. El sistema hace los cambios necesarios en la base de datos. 2. El sistema muestra un mensaje acerca del resultado de la operación.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos, el sistema muestra un aviso.	

Nombre:	Modificar aviso.
Descripción: Permite cambiar detalles de un aviso existente.	
Actores: Administrador.	
Precondiciones: Ver aviso.	
Flujo Normal: 1. El actor realiza cambios en los detalles mostrados. 2. Hace clic en el botón “guardar”. 2. El sistema guarda los cambios y muestra un mensaje acerca del resultado de la operación.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos, el sistema muestra un aviso.	

Nombre:	Entrar a módulo de personal.
Descripción: Permite ver las opciones relacionadas con registros de personal.	
Actores: Administrador, funcionario.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita el módulo de personal. 2. El sistema devuelve como respuesta un menú con las opciones del módulo.	
Flujo Alternativo:	

Nombre:	Buscar personal.
Descripción: Permite buscar registros del personal a partir de un criterio de búsqueda.	
Actores: Administrador, funcionario.	
Precondiciones: Entrar a módulo de personal.	
Flujo Normal: 1. El actor ingresa un criterio de búsqueda. 2. Presiona el botón “buscar”. 3. El sistema muestra un listado de los registros de personal coincidentes con el criterio.	
Flujo Alternativo: Si hay un problema de lectura de la base de datos, el sistema muestra un mensaje de error. Si no existen registros coincidentes con el criterio se muestra un mensaje.	

Nombre:	Crear nuevo registro de personal.
Descripción: Permite crear un nuevo registro.	
Actores: Administrador.	
Precondiciones: Entrar al módulo de personal.	
Flujo Normal: 1. El actor hace clic en el botón “Nueva alta de personal”. 2. El sistema muestra un formulario con los campos de registros de personal. 3. El actor ingresa los datos. 3. El actor presiona el botón “guardar”. 4. El sistema valida los datos y guarda en la base de datos.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Ver currículum.
Descripción: Permite desplegar el currículum asociado a un registro de personal.	
Actores: Administrador, funcionario.	
Precondiciones: Buscar personal.	
Flujo Normal: 1. El actor hace clic en el botón “ver currículum” de uno de los registros de personal listados. 2. El sistema muestra el currículum de dicho registro.	
Flujo Alternativo: Si hay un problema de lectura de la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Consultar personal.
Descripción: Permite ver los detalles de un registro de personal.	
Actores: Administrador, funcionario.	
Precondiciones: Buscar personal.	
Flujo Normal: 1. El actor hace clic en el botón “consultar” de uno de los registros de personal listados. 2. El sistema muestra los detalles de dicho registro.	
Flujo Alternativo: Si hay un problema de lectura de la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Modificar personal.
Descripción: Permite cambiar los detalles de un registro de personal.	
Actores: Administrador.	
Precondiciones: Consultar personal.	
Flujo Normal: 1. El actor hace clic en el botón “Editar este registro”. 2. Hace cambios en los detalles mostrados. 3. Presiona el botón “guardar”. 4. El sistema muestra un mensaje acerca del resultado de la operación.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Eliminar registro de personal.
Descripción: Permite borrar un registro del personal.	
Actores: Administrador.	
Precondiciones: Modificar personal.	
Flujo Normal: 1. El actor hace clic en el botón “Eliminar este registro”. 2. El sistema solicita una confirmación antes de proceder. 3. El actor confirma la operación. 4. El sistema actualiza la base de datos y muestra un mensaje con el resultado de la operación.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Entrar a módulo “Mi cuenta”.
Descripción: Permite a un usuario consultar y/o modificar algunos de sus datos personales y de cuenta.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita el módulo de “Mi cuenta”. 2. El sistema devuelve como respuesta un menú con las opciones del módulo.	
Flujo Alternativo: Si hay un problema al consultar la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Cambiar contraseña.
Descripción: Permite a un usuario cambiar la contraseña de la cuenta que esté utilizando.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Entrar a módulo de “Mi cuenta”.	
Flujo Normal: 1. El actor ingresa su contraseña actual. 2. Proporciona la nueva contraseña que desea utilizar y la confirma ingresándola nuevamente. 3. Presiona el botón “guardar”. 4. El sistema avisa que se ha guardado su cambio de contraseña.	
Flujo Alternativo: Si el actor ingresa una contraseña actual inválida el sistema no realiza cambio alguno y avisa al actor del error de contraseña. Si el actor no confirma correctamente la contraseña nueva escribiéndola dos veces de la misma manera, no se realiza ningún cambio en su contraseña y el sistema avisa al actor de su error.	

Nombre:	Editar currículum.
Descripción: Permite a un usuario capturar, actualizar o ver datos de su currículum.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Entrar a módulo de “Mi cuenta”.	
Flujo Normal: 1. El actor selecciona qué rubro de su currículum quiere modificar o ver. 2. El sistema muestra el o los formularios del rubro solicitado. 3. El actor realiza cambios en un formulario. 4. El actor presiona el botón “guardar”. 5. El sistema avisa que se han guardado los cambios en su currículum.	
Flujo Alternativo: Si el sistema no puede leer el currículum desde la base de datos muestra un aviso.	

Nombre:	Solicitar cambio de email.
Descripción: Permite a un usuario capturar solicitar un cambio en su registro de personal referente a su dirección de correo electrónico.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Entrar a módulo de “Mi cuenta”.	
Flujo Normal: 1. El sistema muestra al actor qué email está registrado en la base de datos. 2. El actor ingresa una nueva dirección de correo electrónico. 3. El sistema valida la dirección de email. 4. El actor presiona el botón “guardar”. 4. El sistema avisa que se ha creado una solicitud que revisará un administrador.	
Flujo Alternativo: Si el actor tiene una solicitud de cambio de email pendiente de revisión por un administrador, el sistema avisa de este hecho en vez de permitir enviar otra solicitud.	

Nombre:	Entrar a módulo de cuentas.
Descripción: Permite al administrador ver y modificar las cuentas para acceder al sistema.	
Actores: Administrador.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita el módulo de cuentas. 2. El sistema devuelve como respuesta un menú con las opciones del módulo.	
Flujo Alternativo:	

Nombre:	Buscar cuentas.
Descripción: Permite buscar cuentas del sistema a partir de un criterio de búsqueda.	
Actores: Administrador.	
Precondiciones: Entrar a módulo de cuentas.	
Flujo Normal: 1. El actor ingresa un criterio de búsqueda. 2. Presiona el botón “buscar”. 3. El sistema muestra un listado de las cuentas coincidentes con el criterio.	
Flujo Alternativo: Si hay un problema de lectura de la base de datos, el sistema muestra un mensaje de error. Si no existen registros coincidentes con el criterio se muestra un mensaje.	

Nombre:	Ver cuenta.
Descripción: Permite ver los detalles de una cuenta de acceso al sistema.	
Actores: Administrador.	
Precondiciones: Buscar cuentas.	
Flujo Normal: 1. El actor solicita ver una cuenta. 2. El sistema muestra los detalles de ésta.	
Flujo Alternativo: Si hay un problema de lectura de la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Eliminar cuenta.
Descripción: Permite eliminar una de las cuentas de acceso al sistema.	
Actores: Administrador.	
Precondiciones: Ver cuenta.	
Flujo Normal: 1. El actor solicita la supresión de la cuenta mostrada. 2. El sistema realiza los cambios necesarios en la base de datos.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Crear cuenta.
Descripción: Permite generar una nueva cuenta de acceso al sistema asociada a un registro de personal.	
Actores: Administrador.	
Precondiciones: Entrar a módulo de cuentas.	
Flujo Normal: 1. El actor ingresa el nombre de usuario y la contraseña para la nueva cuenta. 2. El actor presiona el botón de “guardar”. 3. El sistema crea el nuevo registro de cuenta en la base de datos.	
Flujo Alternativo: Si hay un problema al actualizar la base de datos, el sistema muestra un mensaje de error.	

Nombre:	Entrar al sistema y leer avisos.
Descripción: Permite ver los avisos de sistema y el menú principal.	
Actores: Administrador, funcionario, profesor.	
Precondiciones: Autenticación.	
Flujo Normal: 1. El actor solicita la página de inicio del sistema. 2. El sistema devuelve como respuesta un menú con los menús generales del sistema y los avisos que conciernen al tipo de actor correspondiente.	
Flujo Alternativo:	

Anexo B. Diccionario de datos.

actividad_profesional

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave de la actividad
nombre	varchar(254)	NO	NO	NO			nombre de la actividad
descripcion	varchar(254)	NO	NO	NO			descripción de la actividad

Index Name	Columns
PRIMARY	id
clave	clave
nombre	nombre

2 area_licenciatura

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	int(11)	NO	NO	NO			clave de la licenciatura
nombre	varchar(254)	NO	NO	NO			nombre de la licenciatura
descripcion	varchar(254)	NO	NO	NO			descripción de la licenciatura

Index Name	Columns
PRIMARY	id
clave	clave
nombre	nombre
nombre_2	nombre
nombre_3	nombre

3 asignatura

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave de la asignatura
tipo_clase_fk	int(11)	NO	NO	NO			id del tipo de clase (lab., teoría, etc)
horas_semana	float	NO	NO	NO			número de horas semanales
hrsl	float	NO	NO	NO			número de horas semanales (en desuso)
creditos	float	NO	NO	NO			cantidad de créditos de la asignatura
semestre	int(11)	NO	NO	NO			número del semestre al que pertenece la asignatura
nivel	int(11)	NO	NO	NO			nivel de la asignatura
mnemo_as	varchar(254)	NO	NO	NO			mnemotécnico de la asignatura
mnemo_ent	varchar(254)	NO	NO	NO			mnemotécnico (en desuso)
division_fk	int(11)	NO	NO	NO			id de la división a la que pertenece la asignatura (en desuso)
plan	varchar(254)	NO	NO	NO			plan al que pertenece la asignatura
area_fk	int(11)	NO	NO	NO			id del área a la que pertenece la asignatura
nombre	varchar(254)	NO	NO	NO			nombre de la asignatura
entidad_fk	int(11)	NO	NO	NO			id de la entidad a la que pertenece la asignatura
clasificacion_fk	int(11)	NO	NO	NO			id de la clasificación de la asignatura

Index Name	Columns

PRIMARY	id
mnemo_ent	mnemo_ent
mnemo_as	mnemo_as
asignatura_tipo_clase	tipo_clase_fk
asignatura_entidad	entidad_fk
asignatura_clasificacion	clasificacion_fk

4 aviso

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
creador_fk	int(11)	NO	YES	NO			id del miembro del personal creador del aviso
creador_nombre	varchar(254)	NO	NO	NO			nombre del creador. en caso de que el regsitro de personal sea borrado.
texto1	varchar(1024)	NO	NO	NO			texto guardado
texto2	varchar(254)	NO	NO	NO			texto alternativo 1(en desuso)
texto3	varchar(254)	NO	NO	NO			texto alternativo 2(en desuso)
fecha_creacion	datetime	NO	NO	NO			fecha en que se creó o modificó el aviso
fecha_inicio	date	NO	NO	NO			fecha de inicio de despliegue del aviso
fecha_fin	date	NO	NO	NO			fecha de expiración de despliegue del aviso.
desplegar_admin	int(11)	NO	NO	NO			indicador de despliegue a administradores.
desplegar_funcionario	int(11)	NO	NO	NO			indicador de despliegue a funcionarios.
desplegar_profesor	int(11)	NO	NO	NO			indicador de despliegue a profesores.
prioridad	int(11)	NO	NO	NO			prioridad del aviso (en desuso)
desplegar_publico	int(11)	NO	NO	NO			indicador de despliegue en la página de inicio.
titulo	varchar(254)	NO	NO	NO			titulo del aviso

Index Name	Columns
PRIMARY	id
mensaje_pers	creador_fk

5 bloque

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
semestre_fk	int(11)	NO	YES	NO			id del semestre al que pertenece el bloque de asignaturas
numero	int(11)	NO	YES	NO			número de bloque

Index Name	Columns
PRIMARY	id
bloque_semestre	semestre_fk

6 clasificacion_asignatura

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave de la clasificación de asignatura

nombre	varchar(254)	NO	NO	NO		nombre de la calsificación
descripcion	varchar(254)	NO	NO	NO		descripción de la clasificación

Index Name	Columns
PRIMARY	id
clave	clave
nombre	nombre

7 conjuncion

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
conjuncion_id	int(11)	YES	YES	NO			identificador
nombre	varchar(254)	NO	YES	NO			conjunción

Index Name	Columns
PRIMARY	conjuncion_id

8 cuenta

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
uname	varchar(24)	NO	YES	NO			nombre de usuario
pass	varchar(24)	NO	YES	NO			contraseña de la cuenta
personal_fk	int(11)	NO	NO	NO			id del registro del personal al que pertenece la cuenta
tipocuenta_fk	int(11)	NO	YES	NO		0	id del tipo de cuenta
status	int(11)	NO	NO	NO		0	idicador de status de la cuenta (0:inactivo;1,activo)
expire	date	NO	NO	NO			fecha de expiración de la cuenta
lastlogin	datetime	NO	NO	NO			fecha y hora de último login con la cuenta

Index Name	Columns
PRIMARY	id
uname	uname
cuenta_pers	personal_fk
cuenta_tipo	tipocuenta_fk

9 curr_actividad

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
curriculum_fk	int(11)	NO	YES	NO			id del currículum al que pertenece la actividad
tipo	varchar(254)	NO	NO	NO			tipo de la actividad: ACADEMICA o PROFESIONAL
nombre	varchar(254)	NO	NO	NO			nombre de la actividad
descripcion	varchar(254)	NO	NO	NO			descripción de la actividad
institucion	varchar(254)	NO	NO	NO			institición donde se llevó a cabo la actividad
fecha_inicio	date	NO	NO	NO			fecha en que se inició la actividad
fecha_fin	date	NO	NO	NO			fecha de término de la actividad

Index Name	Columns
PRIMARY	id
curractividad_curr	curriculum_f

k

10 curr_grado_acad

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
curriculum_fk	int(11)	NO	YES	NO			id del currículum al que pertenece el grado académico
nacion_fk	int(11)	NO	NO	NO			id de la nacionalidad donde se obtuvo el grado
institucion	varchar(254)	NO	NO	NO			institución donde se obtuvo el grado académico
tipo	varchar(254)	NO	NO	NO			tipo de grado académico: LICENCIATURA, MAESTRIA, DOCTORADO, ESPECIALIZACION
fecha_inicio	date	NO	NO	NO			fecha en que comenzaron los estudios para obtener el grado
fecha_fin	date	NO	NO	NO			fecha de término de los estudios para obtener el grado
area_especializacion	varchar(254)	NO	NO	NO			área de la especialización del grado
subarea_especializacion	varchar(254)	NO	NO	NO			subarea de especialización del grado
comentarios	varchar(254)	NO	NO	NO			conentarios acerca del grado
nombre	varchar(254)	NO	NO	NO			nombre completo del grado académico
descripcion	varchar(254)	NO	NO	NO			descripción del grado académico
fecha_titulacion	date	NO	NO	NO			fecha de titulación

Index Name	Columns
PRIMARY	id
currgradoacad_curr	curriculum_fk
nacion_curr	nacion_fk

11 curr_materia_impartida

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
curriculum_fk	int(11)	NO	YES	NO			id del currículum al que pertenece la materia impartida
nombre	varchar(254)	NO	NO	NO			nombre de la materia impartida
nivel_correspondiente	varchar(254)	NO	NO	NO			nivel al que corresponde la materia
periodo	varchar(254)	NO	NO	NO			periodo de impartición
descripcion	varchar(254)	NO	NO	NO			descripción de la materia impartida
institucion	varchar(254)	NO	NO	NO			institución donde fue impartida la materia
dada_en_unam	int(11)	NO	NO	NO			indicador de si fue impartida en la UNAM (en desuso)
dada_en_fi	int(11)	NO	NO	NO			indicador de si fue impartida en la FI (en desuso)
fecha_inicio	date	NO	NO	NO			fecha de inicio de impartición
fecha_fin	date	NO	NO	NO			fecha de fin de impartición
horas_teoria	int(11)	NO	NO	NO			cantidad de horas de teoría de la materia impartida

horas_laboratorio	int(11)	NO	NO	NO	cantidad de horas de laboratorio de la materia impartida
clasif_fk	int(11)	NO	NO	NO	id de la calsifaición de la asignatura
tipo_mat_imp	varchar(254)	NO	NO	NO	tipo de materia impartida
veces	int(11)	NO	NO	NO	veces que se ha impartido la materia

Index Name	Columns
PRIMARY	id
currmatimp_curr	curriculum_fk
currmatimp_clasif	clasif_fk

12 curr_membresia

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
curriculum_fk	int(11)	NO	YES	NO			id del currículum al que pertenece la membresía
asociacion	varchar(254)	NO	NO	NO			nombre o siglas de la asociación de la membresía
tipo_membresia	varchar(254)	NO	NO	NO			tipo de membresía
descripcion	varchar(254)	NO	NO	NO			descripción de la membresía
fecha_inicio	date	NO	NO	NO			fecha de inicio de la membresía
fecha_fin	date	NO	NO	NO			fecha de expiración de la membresía

Index Name	Columns
PRIMARY	id
currmembresia_curr	curriculum_fk

13 curr_nombramientos

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave de los nombramientos
nombre	varchar(254)	NO	NO	NO			nombre de los nombramientos
otros	int(11)	NO	NO	NO			indicador de otros nombramientos (en desuso)
descripcion	varchar(254)	NO	NO	NO			descripción de los nombramientos

Index Name	Columns
PRIMARY	id
clave	clave
nombre	nombre

14 curr_premio

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
curriculum_fk	int(11)	NO	YES	NO			id del currículum al que peretence el premio
nombre	varchar(254)	NO	NO	NO			nombre del premio
institucion	varchar(254)	NO	NO	NO			institución otorgante
otorgante	varchar(254)	NO	NO	NO			nombre de la persona otorgante (en desuso)

descripcion	varchar(254)	NO	NO	NO		descripción del premio
fecha	date	NO	NO	NO		año de obtención del premio

Index Name	Columns
PRIMARY	id
currpremio_curr	curriculum_fk

15 curr_producto

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
curriculum_fk	int(11)	NO	NO	NO			id del currículum al que pertenece el producto
nombre	varchar(254)	NO	NO	NO			nombre del producto
descripcion	varchar(254)	NO	NO	NO			descripción del producto
tipo_fk	int(11)	NO	NO	NO			id del tipo de producto
fecha	date	NO	NO	NO			fecha de elaboración

Index Name	Columns
PRIMARY	id
currprod_curr	curriculum_fk
currprod_tipoprod	tipo_fk

16 curriculum

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
personal_fk	int(11)	NO	YES	NO			id del registro de personal a que pertenece el currículum
fecha_modificacion	date	NO	NO	NO			fecha de la última modificación del currículum
serie	int(11)	NO	NO	NO			serie del currículum (en desuso)
capturador_fk	int(11)	NO	NO	NO			id del útimo modificador del currículum
resumen_exp_prof	varchar(254)	NO	NO	NO			texto de experiencia profesional
resumen_exp_acad	varchar(254)	NO	NO	NO			texto de experiencia académica
campo_exp_prof	text	NO	NO	NO			texto acerca del campo de experiencia profesional
logros	text	NO	NO	NO			texto acerca de los logros académicos y profesionales
licenciatura_fk	int(11)	NO	NO	NO			id de ll licenciatura principal
grado_fk	int(11)	NO	NO	NO			id de grado académico
ppal_act_prof_fk	int(11)	NO	NO	NO			id de la pricipal actividad profecional
tipo_nombramientos_fk	int(11)	NO	NO	NO			id del tipo de otros nombramientos
nombramiento_fk	int(11)	NO	NO	NO			id del tipo de nombramientos que se tienen
puesto	varchar(254)	NO	NO	NO			nombre del puesto actual
prod_cant_libros	int(11)	NO	NO	NO			cantidad de libros elaborados
prod_cant_notas_clase	int(11)	NO	NO	NO			cantidad de notas de clase elaboradas
prod_cant_mat_did	int(11)	NO	NO	NO			cantidad de materiales didácticos elaborados
prod_cant_man_prac	int(11)	NO	NO	NO			cantidad de manuales de prácticas elaborados
prod_cant_articulos	int(11)	NO	NO	NO			cantidad de artículos elaborados
prod_cant_memorias	int(11)	NO	NO	NO			cantidad de memorias elaboradas
prod_cant_patentes	int(11)	NO	NO	NO			cantidad de patentes registradas

prod_cant_trabajos_industria	int(11)	NO	NO	NO	cantidad de trabajos para la industria elaborados
prod_cant_articulos_divulga	int(11)	NO	NO	NO	cantidad de actículos de divulgación elaborados
prod_cant_foros	int(11)	NO	NO	NO	cantidad de foros en que se ha participado
prod_serv_industria	int(11)	NO	NO	NO	cantidad de servicios a la industria elaborados
prod_conv_industria	int(11)	NO	NO	NO	cantidad de conveniios con la industria elaborados
exp_asignaturas	int(11)	NO	NO	NO	tipo de experiencia en las asignaturas impartidas

Index Name	Columns
PRIMARY	id
curr_nombramiento	nombramiento_fk
licenciatura_fk	licenciatura_fk
curr_grado	grado_fk
curr_actprof	ppal_act_prof_fk
curr_tiponom	tipo_nombramientos_fk
curr_pers	personal_fk

17 entidad

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
jefe_fk	int(11)	NO	NO	NO			id del miembro del personal que es jefe de la entidad
nivel_fk	int(11)	NO	NO	NO			id del tipo o nivel de la entidad (departamento, coordinación, etc.)
super_entidad_fk	int(11)	NO	NO	NO			id de la entidad a la que pertenece la entidad
siglas	varchar(254)	NO	NO	NO			siglas de la entidad
conjuncion_fk	int(11)	NO	NO	NO			id de la conjunción que se usa en el nombre de la entidad
area	varchar(254)	NO	NO	NO			parte del nombre de la entidad que describe su área
clave_usecad	varchar(254)	NO	NO	NO			clave que usa usecad para identificar la entidad
ubicacion	varchar(254)	NO	NO	NO			ubicación de la entidad

Index Name	Columns
PRIMARY	id
entidad_conj	conjuncion_fk
entidad_nivel	nivel_fk
entidad_supent	super_entidad_fk
entidad_jefe	jefe_fk

18 ficha_biblio

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
producto_fk	int(11)	NO	YES	NO			id el tipo de producto
tipo	varchar(254)	NO	NO	NO			texto de tipo de producto
descripcion	varchar(254)	NO	NO	NO			descripción de la obra

autores	varchar(254)	NO	NO	NO	autores de la obra
traductores	varchar(254)	NO	NO	NO	traductores de la obra
titulo	varchar(254)	NO	NO	NO	título de la obra
volumen	varchar(254)	NO	NO	NO	volumen
pais	varchar(254)	NO	NO	NO	país donde fue editado
editorial	varchar(254)	NO	NO	NO	editorial ddonde fue publicado
fecha_edicion	date	NO	NO	NO	fecha de edición de la obra
coleccion	varchar(254)	NO	NO	NO	nombre de la colección a la que pertenece la obra
paginas	int(11)	NO	NO	NO	cantidad de páginas de la obra
pub_period_periodicidad	varchar(254)	NO	NO	NO	perioricidad de la obra
pub_period_fecha	date	NO	NO	NO	fecha del ejemplar si se trata de una obra periódica
rango_paginas	varchar(254)	NO	NO	NO	rango de páginas de publicación (si se trata de un artículo)
temas	varchar(254)	NO	NO	NO	palabras clave acerca de los temas de la obra
notas	varchar(254)	NO	NO	NO	notas de la obra

Index Name	Columns
PRIMARY	id
fichab_prod	producto_fk

19 grado

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
niv_esc	int(11)	NO	YES	NO			nivel de escolaridad
jerarquia	int(11)	NO	YES	NO			jerarquía del grado
nombre	varchar(254)	NO	NO	NO			nombre del grado
descripcion	varchar(254)	NO	NO	NO			descripción del grado

Index Name	Columns
PRIMARY	id
nombre	nombre

20 grupo

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
grupo	int(11)	NO	NO	NO			número de grupo
profesor_fk	int(11)	NO	NO	NO			id del profesor (miembro del personal) asignado al grupo
hora_inicio	int(11)	NO	NO	NO			hora de inicio de la clase
hora_fin	int(11)	NO	NO	NO			hora de fin de la clase
d_lun	tinyint(1)	NO	NO	NO			indicador de impartición en lunes
d_mar	tinyint(1)	NO	NO	NO			indicador de impartición en martes

d_mie	tinyint(1)	NO	NO	NO	indicador de impartición en miércoles
d_jue	tinyint(1)	NO	NO	NO	indicador de impartición en jueves
d_vie	tinyint(1)	NO	NO	NO	indicador de impartición en viernes
d_sab	tinyint(1)	NO	NO	NO	indicador de impartición en sábados
salon_fk	int(11)	NO	NO	NO	id del salon donde se imparte la clase
entidad_fk	int(11)	NO	NO	NO	id de la entidad a la que pertenece la materia impartida (en desuso)
semestre_fk	int(11)	NO	NO	NO	id del semestre al que pertenece el grupo
asignatura_fk	int(11)	NO	NO	NO	id de la asignatura del grupo
inscritos	int(11)	NO	NO	NO	cantidad de alumnos iscritos en el grupo
oyentes	int(11)	NO	NO	NO	cantidad de oyentes permitidos
cupo	int(11)	NO	NO	NO	cantidad máxima de alumnos inscritos
seccion	enum('A','B','C','D','E')	NO	NO	NO	sección del grupo
bloque_fk	int(11)	NO	NO	NO	id del bloque al que pertenece el grupo
calif_10	int(11)	NO	NO	NO	cantidad de alumnos que obtuvieron calificación de 10
calif_9	int(11)	NO	NO	NO	cantidad de alumnos que obtuvieron calificación de 9
calif_8	int(11)	NO	NO	NO	cantidad de alumnos que obtuvieron calificación de 8
calif_7	int(11)	NO	NO	NO	cantidad de alumnos que obtuvieron calificación de 7
calif_6	int(11)	NO	NO	NO	cantidad de alumnos que obtuvieron calificación de 6
calif_5	int(11)	NO	NO	NO	cantidad de alumnos que obtuvieron calificación de 5
calif_np	int(11)	NO	NO	NO	cantidad de alumnos que obtuvieron NP
primer_ingreso	tinyint(1)	NO	NO	NO	idicador de que el grupo es de alumnos de primer ingreso

Index Name	Columns
PRIMARY	id
grupo_profesor	profesor_fk
grupo_salon	salon_fk
grupo_semestre	semestre_fk
grupo_asignatura	asignatura_fk
grupo_entidad	entidad_fk
grupo_bloque	bloque_fk

21 licenciatura

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
area_fk	int(11)	NO	NO	NO			id del área de la licenciatura
subarea_fk	int(11)	NO	NO	NO			id de la subárea de la licenciatura
area	varchar(254)	NO	NO	NO			texto del área de la licenciatura (en desuso)
subarea	varchar(254)	NO	NO	NO			texto de la subárea de la licenciatura (en desuso)
clave	int(11)	NO	NO	NO			clave de la licenciatura
nombre	varchar(254)	NO	NO	NO			nombre completo de la licenciatura
descripcion	varchar(254)	NO	NO	NO			descripción de la licenciatura

Index Name	Columns
PRIMARY	id
clave	clave

nombre	nombre
licenciatura_area	area_fk
licenciatura_subarea	subarea_fk

22 nacion

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave de la nacionalidad
nombre	varchar(254)	NO	NO	NO			nombre del páis
gentilicio	varchar(254)	NO	NO	NO			gentilicio
nacionalidad	varchar(254)	NO	NO	NO			nombre de la nacionalidad
notas	varchar(254)	NO	NO	NO			notas (en desuso)

Index Name	Columns
PRIMARY	id
clave	clave
nombre	nombre

23 nivel

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
nivel_id	int(11)	YES	YES	YES			identificador
nombre	varchar(254)	NO	YES	NO			nombre del tipo
jerarquia	int(11)	NO	NO	NO			jerarquía del tipo (en desuso)

Index Name	Columns
PRIMARY	nivel_id

24 pers_ent

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
personal	int(11)	YES	YES	NO			id del personal
entidad	int(11)	YES	YES	NO			id de la entidad

Index Name	Columns
PRIMARY	personal,entidad
persent_ent	entidad

25 personal

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
paterno	varchar(254)	NO	NO	NO			apellido paterno
materno	varchar(254)	NO	NO	NO			apellido materno
nombres	varchar(254)	NO	NO	NO			nombres
titulo_fk	int(11)	NO	NO	NO			id del título personal
grado_new_fk	int(11)	NO	NO	NO			id del grado del personal
rfc	varchar(254)	NO	NO	NO			RFC
licenciatura_fk	int(11)	NO	NO	NO			id de la licenciatura que tiene el mimbro del personal
sexo_fk	int(11)	NO	NO	NO			id del sexo (en desuso)
edocivil_fk	int(11)	NO	NO	NO			id del estado civil (en desuso)
nacion_fk	int(11)	NO	NO	NO			id de la nacionalidad (en desuso)
fechaIngreso	date	NO	NO	NO			fecha de ingreso a la DCB
numero_trabaj	int(11)	NO	NO	NO			número de trabajador

ador					
folio	varchar(254)	NO	NO	NO	folio que usa la Facultad de Ingeniería
curp	varchar(254)	NO	NO	NO	CURP
direccion_calle	varchar(254)	NO	NO	NO	calle de domicilio
direccion_colonia	varchar(254)	NO	NO	NO	colonia de domicilio
direccion_delegacion	varchar(254)	NO	NO	NO	delegación de domicilio
direccion_codpost	varchar(254)	NO	NO	NO	código postal de domicilio
direccion_ciudad	varchar(254)	NO	NO	NO	ciudad de domicilio
direccion_pais	varchar(254)	NO	NO	NO	país de domicilio
telefono_casa	varchar(254)	NO	NO	NO	teléfono de casa
telefono_oficina_1	varchar(254)	NO	NO	NO	teléfono 1 de oficina
telefono_oficina_2	varchar(254)	NO	NO	NO	teléfono 2 de oficina
telefono_celular	varchar(254)	NO	NO	NO	teléfono celular
oficina_direccion	varchar(254)	NO	NO	NO	dirección de oficina
profesion	varchar(254)	NO	NO	NO	profesión
email	varchar(254)	NO	NO	NO	dirección de correo electrónico
otro_medio	varchar(254)	NO	NO	NO	otro medio de comuncación
tipopersonal_fk	int(11)	NO	NO	NO	id del tipo de personal
activo	tinyint(1)	NO	NO	NO	indicador de status del miembro del personal (0:inactivo;1:activo))
genero	enum('M','F')	NO	NO	NO	sexo: masculino o femenino
edo_civil	enum('SOLTERO','CASADO')	NO	NO	NO	estado civil: casado o soltero
puesto	varchar(254)	NO	NO	NO	puesto (en desuso)
cedula_profesional	varchar(254)	NO	NO	NO	cedula profesional
folio_provisional	tinyint(1)	NO	NO	NO	indicador de folio provisional

Index Name	**Columns**
PRIMARY	id
numeroTrabajador	numero_trabajador
folio	folio
pers_sexo	sexo_fk
pers_titulo	titulo_fk
pers_edociv	edocivil_fk
pers_licenciatura	licenciatura_fk
pers_nacion_new	nacion_fk
pers_grado_new	grado_new_fk
pers_tipopers	tipopersonal_fk

26 salon

Column Name	**Data Type**	**Primary Key**	**Not Null**	**AutoInc**	**Flags**	**Default Value**	**Comment**

id	int(11)	YES	YES	YES		identificador
numero	varchar(254)	NO	NO	NO		número de salón
zona	varchar(254)	NO	NO	NO		zona de ubicación del salón
edificio	varchar(254)	NO	NO	NO		edificio donde se encuentra el salón
piso	varchar(254)	NO	NO	NO		piso en que se encuentra el salón
cupo	int(11)	NO	NO	NO		cupo del salón
comentarios	varchar(254)	NO	NO	NO		comentarios
superficie	int(11)	NO	NO	NO		cantidad de metros cuadrados del salón
sma_seg	varchar(254)	NO	NO	NO		sistema de seguridad para el salón
clave_acceso	varchar(254)	NO	NO	NO		

Index Name	Columns
PRIMARY	id
numero	numero

27 semestre

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
fecha_inicio	date	NO	NO	NO			fecha de inicio del semestre
fecha_fin	date	NO	NO	NO			fecha de fin del semestre
nombre	varchar(20)	NO	NO	NO			nombre del semestre
activo	tinyint(1)	NO	NO	NO			indicador de status del semestre (0:inactivo;1:activo))
editable_funcionarios	tinyint(1)	NO	NO	NO			indicador de si es editable por los funcionarios

Index Name	Columns
PRIMARY	id
nombre	nombre

28 solicitud_cambio_email

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
personal_fk	int(11)	NO	YES	NO			id del personal al que pertenece la solicitud
status	int(11)	NO	YES	NO			status de la solicitud (en desuso)
email_nuevo	varchar(254)	NO	NO	NO			nueva dirección de email solicitada
mensaje	varchar(254)	NO	NO	NO			mesaje de respuesta de la solicitud (en desuso)
fecha	datetime	NO	NO	NO			fecha de creación de la solicitud

Index Name	Columns
PRIMARY	id
solicitud_cambio_email_pers	personal_fk

29 sub_area_licenciatura

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
super_area_fk	int(11)	NO	NO	NO			id del área a la que pertenece el área (en desuso)
clave	int(11)	NO	NO	NO			clave de la subárea
nombre	varchar(254	NO	NO	NO			nombre de la subárea

	)				
descripcion	varchar(254)	NO	NO	NO	descripción de la subárea

Index Name	Columns
PRIMARY	id
clave	clave
nombre	nombre

30 tipo_act_prof

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
princ_act	int(11)	NO	YES	NO			prncipal actividad
clave	varchar(254)	NO	NO	NO			clave del tipo de actividad
nombre	varchar(254)	NO	NO	NO			nombre del tipo de actividad
descripcion	varchar(254)	NO	NO	NO			descripción del tipo de actividad

Index Name	Columns
PRIMARY	id
clave	clave

31 tipo_clase

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave del tipo de clase
nombre	varchar(254)	NO	NO	NO			nombre del tipo de clase
descripcion	varchar(254)	NO	NO	NO			descripción del tipo de clase

Index Name	Columns
PRIMARY	id
clave	clave

32 tipo_cuenta

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
tipocuenta_id	int(11)	YES	YES	NO			identificador
nombre	varchar(254)	NO	YES	NO			nombre del tipo de cuenta
permisos_fk	int(11)	NO	YES	NO			id de los permisos para el tipo de cuenta (en desuso)

Index Name	Columns
PRIMARY	tipocuenta_id

33 tipo_nombramientos

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave de tipo de nombramientos
nombre	varchar(254)	NO	NO	NO			nombre de tipo de nombramientos
descripcion	varchar(254)	NO	NO	NO			descripción del tipo de nombramientos

Index Name	Columns
PRIMARY	id
clave	clave

nombre	nombre

34 tipo_personal

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
tipopersonal_id	int(11)	YES	YES	YES			identificador
nombre	varchar(254)	NO	NO	NO			nombre del tipo de personal

Index Name	Columns
PRIMARY	tipopersonal_id
nombre	nombre

35 tipo_producto

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
id	int(11)	YES	YES	YES			identificador
clave	varchar(254)	NO	NO	NO			clave del tipo de producto
nombre	varchar(254)	NO	NO	NO			nombre del tipo de producto
descripcion	varchar(254)	NO	NO	NO			descripción del tipo de producto

Index Name	Columns
PRIMARY	id

36 titulo

Column Name	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
titulo_id	int(11)	YES	YES	YES			identificador
nombre	varchar(254)	NO	NO	NO			abreviación del título personal
completo	varchar(254)	NO	NO	NO			nombre del título personal sin abreviar

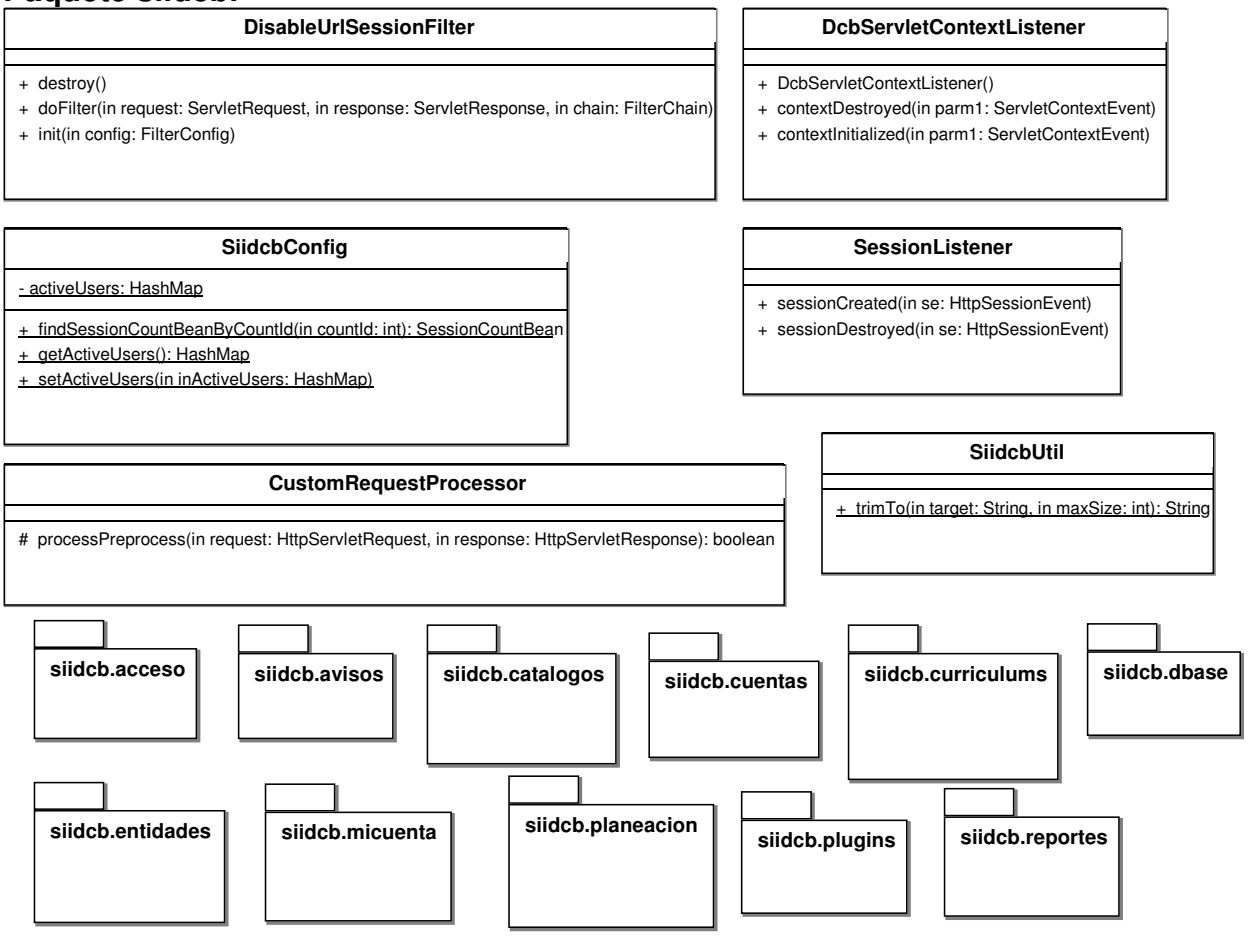
Index Name	Columns
PRIMARY	titulo_id
nombre	nombre

Anexo C. Diagramas de clase.

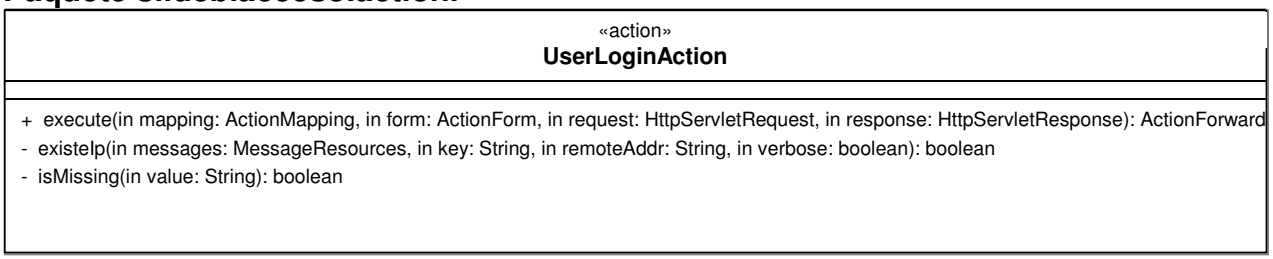
A continuación se muestran los diagramas de clase para la aplicación SIIDCB. En su mayor parte los paquetes mostrados contienen clases que extienden las clases org.apache.struts.action.Action y org.apache.struts.action.ActionForm. Éstas son dos clases del marco de trabajo Struts; la primera sirve para implementar en ella la lógina de negocio y la segunda para hacer mapeos de los formularios HTML mostrados al usuario y realizar validaciones de datos fácilmente.

Las clases de modelo no se muestran aquí pero son mapeos directos de las tablas existentes en la base de datos del sistema. Se encuentran en el paquete “siidcb.dbase”. Puede consultarse el diagrama entidad relación de la base datos y el diccionario de datos si es necesario.

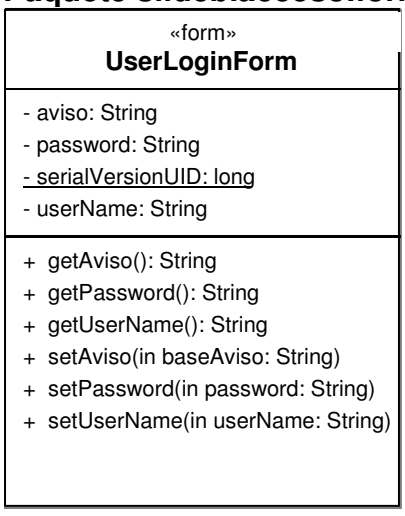
Paquete siidcb.



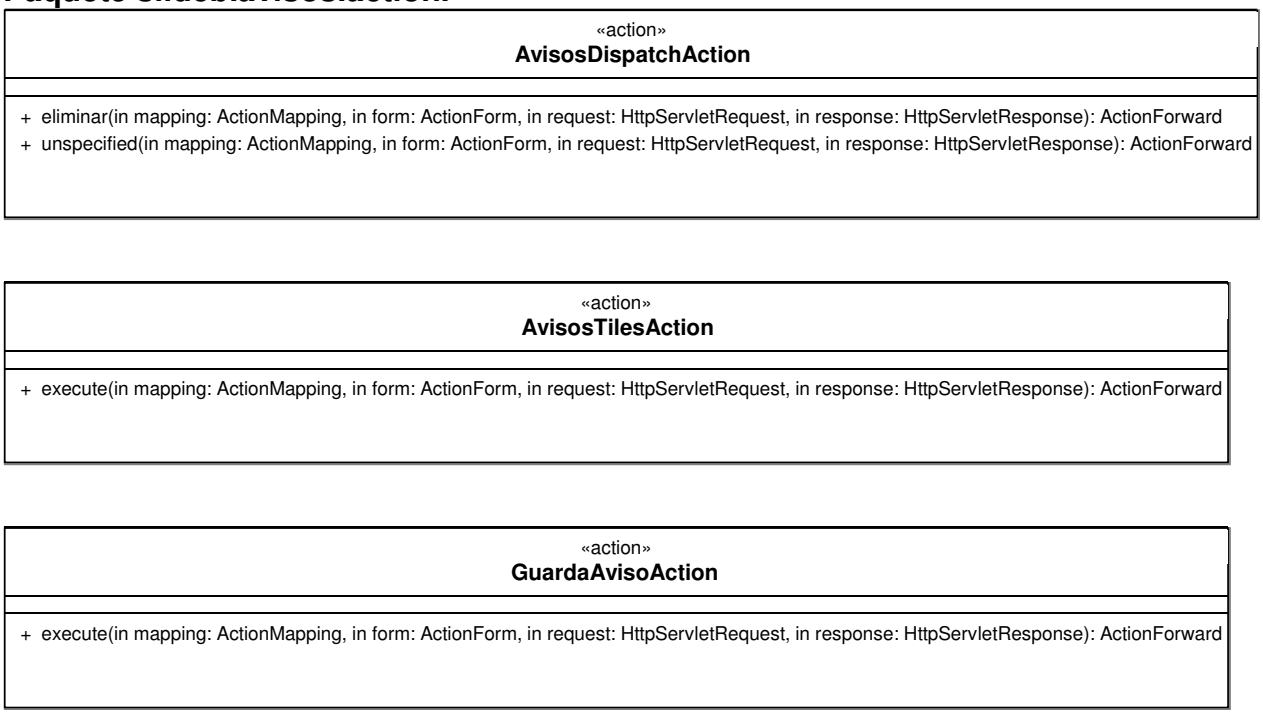
Paquete siidcb.acceso.action.



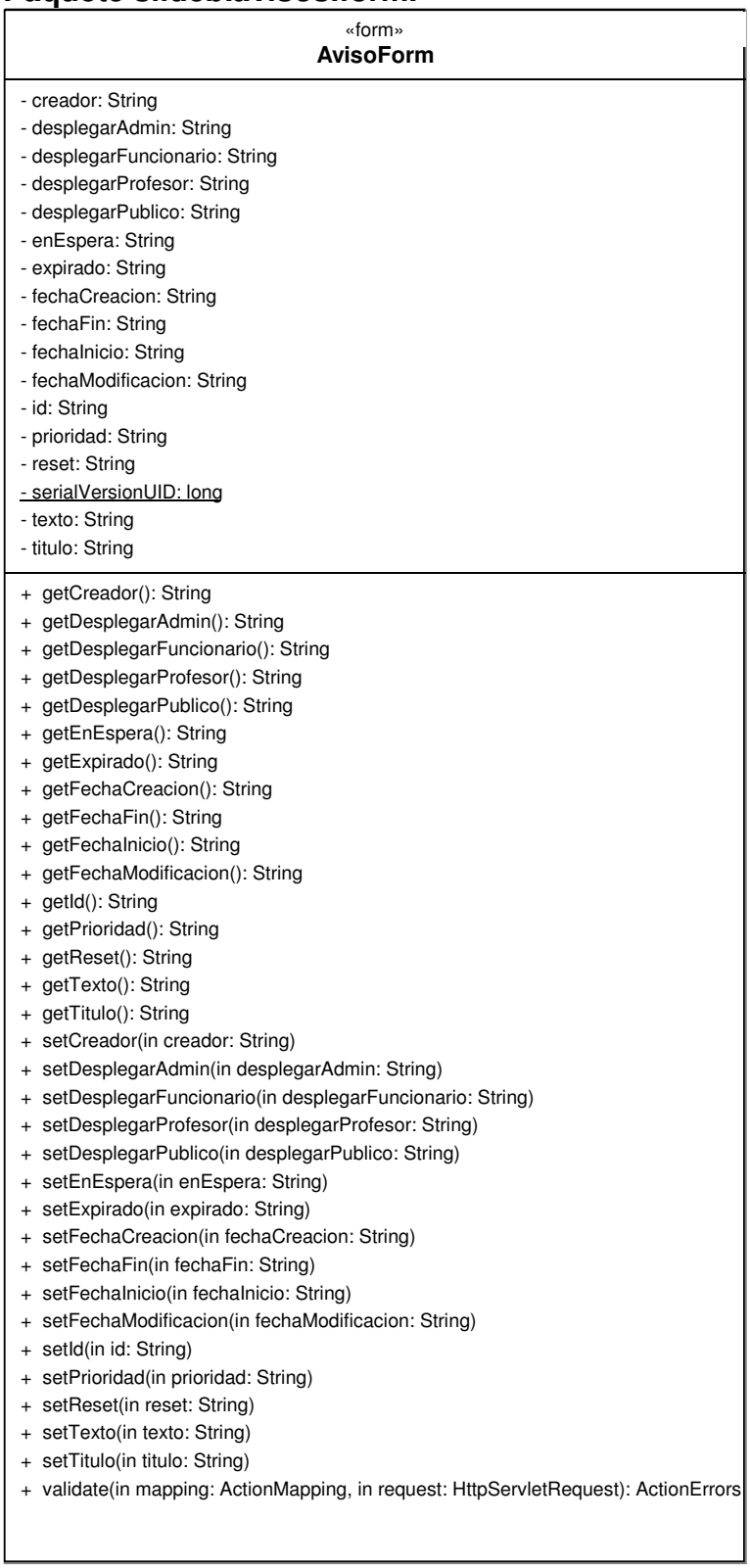
Paquete siidcb.acceso.form.



Paquete siidcb.avisos.action.

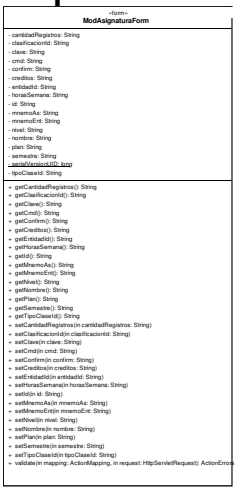


Paquete siidcb.avisos.form.

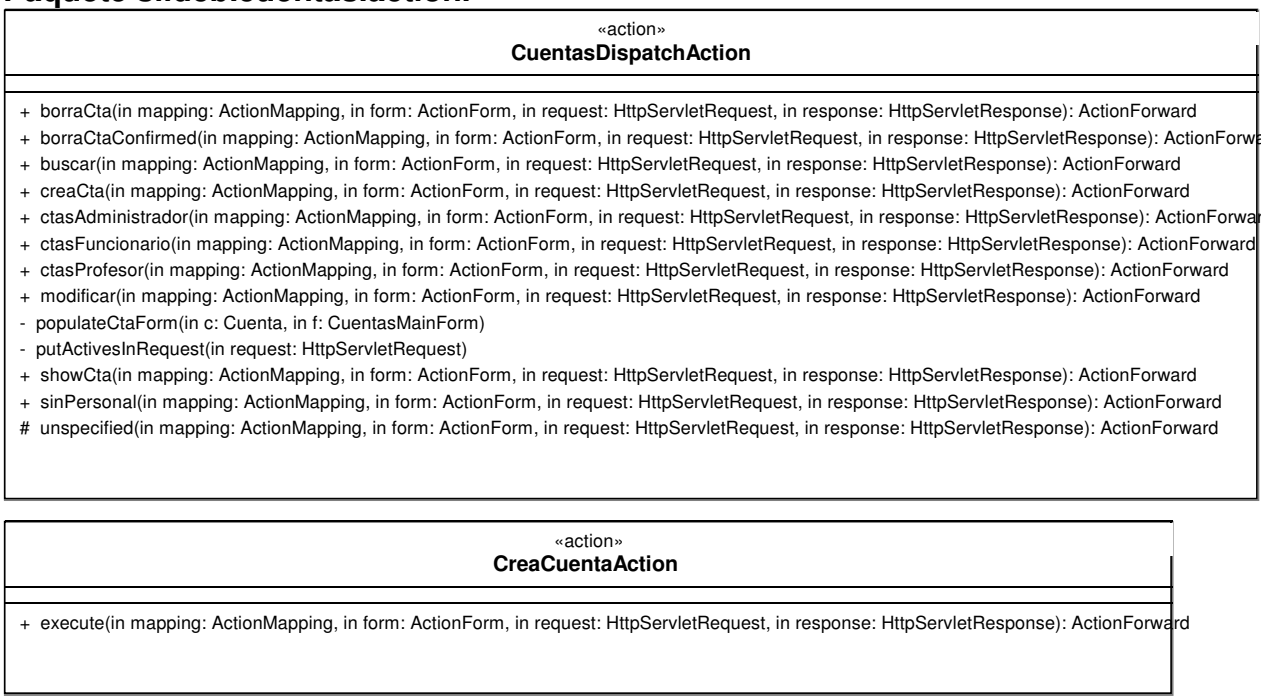


Paquete siidcb.catalogos.action.

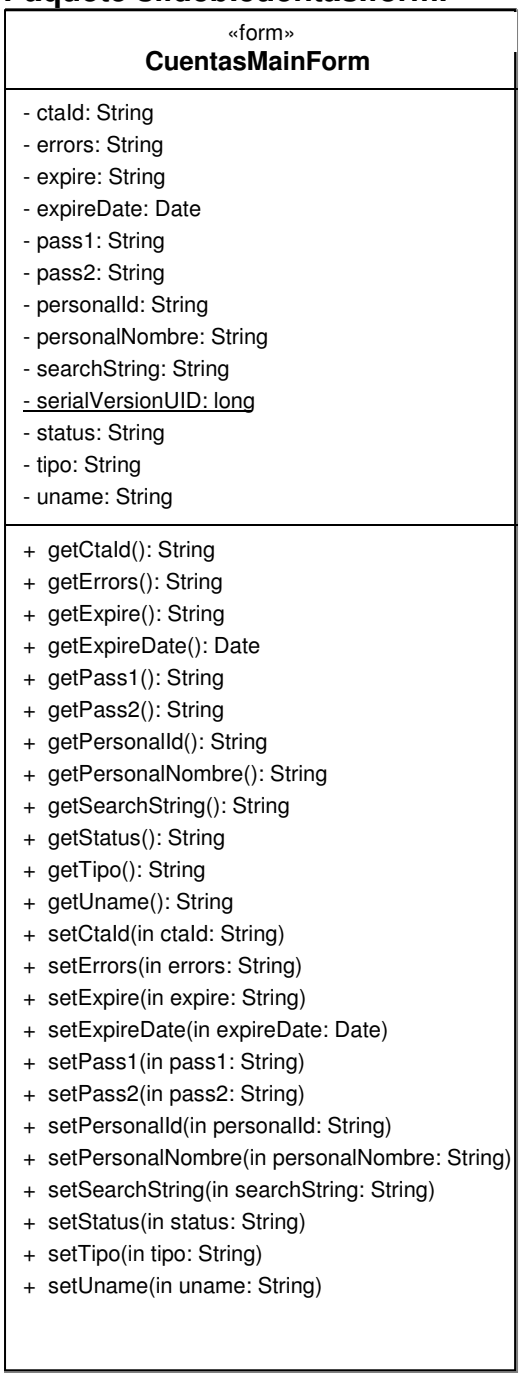
Paquete siidcb.catalogos.form.



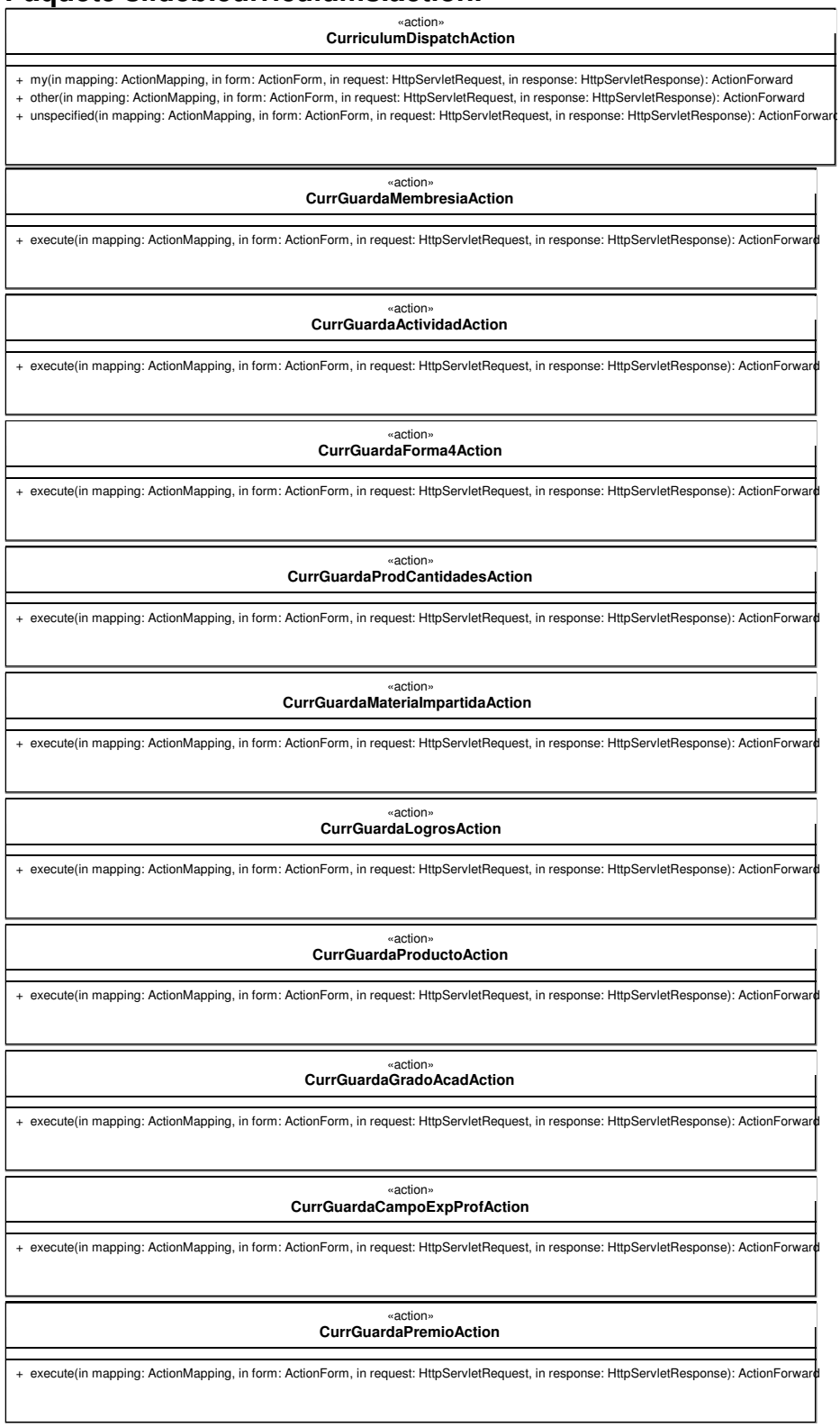
Paquete siidcb.cuentas.action.



Paquete siidcb.cuentas.form.

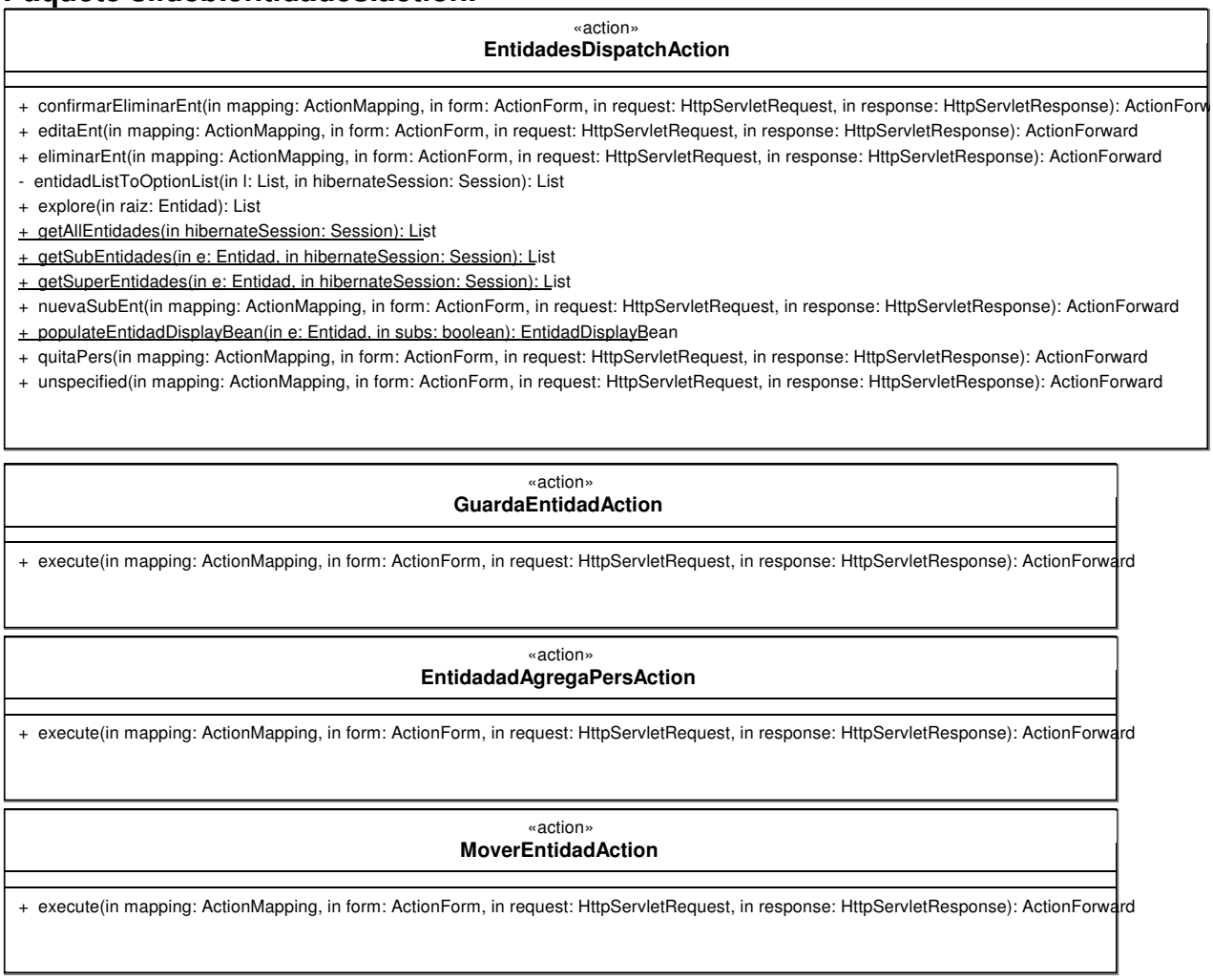


Paquete siidcb.curriculums.action.

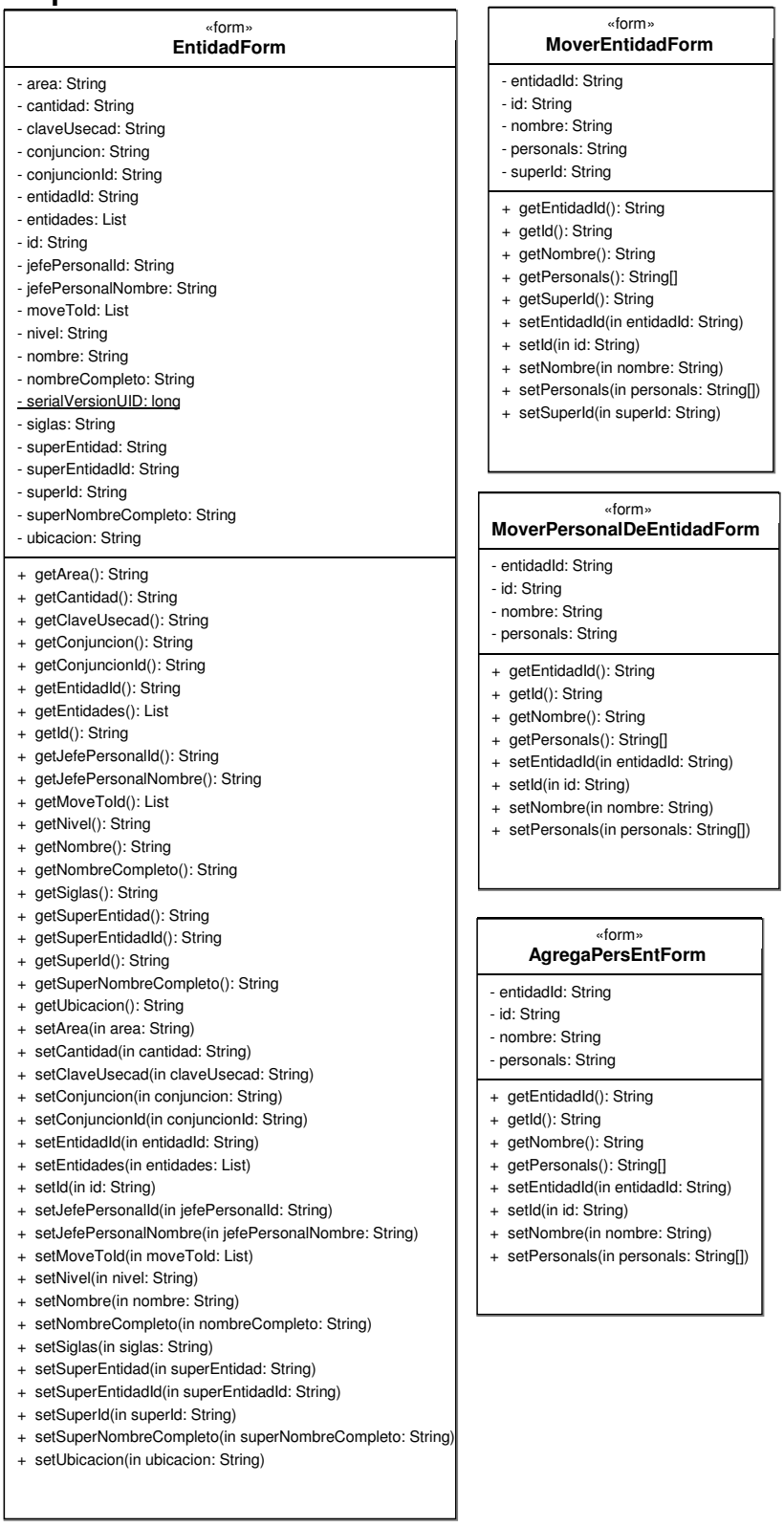


Paquete siidcb.curriculums.form.

Paquete siidcb.entidades.action.



Paquete siidcb.entidades.form.



Paquete siidcb.micuenta.action.

«action»
UsuarioActualizaInfoDispatchAction

+ showEmailForm(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ showPasswordForm(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ unspecified(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

«action»
ProcesarSolicitudCambioEmail

+ execute(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

«action»
UsuarioActualizaPasswordAction

+ execute(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

«action»
UsuarioActualizaEmailAction

+ execute(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

«action»
SolicitudesCambioEmailTilesController

+ execute(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

Paquete siidcb.micuenta.form.

«form»
CambioPasswordForm

- contrasenaActual: String
- mensaje: String
- nuevaContrasena1: String
- nuevaContrasena2: String
- personalId: String
- serialVersionUID: long

+ getContrasenaActual(): String
+ getMensaje(): String
+ getNuevaContrasena1(): String
+ getNuevaContrasena2(): String
+ getPersonalId(): String
+ setContrasenaActual(in contrasenaActual: String)
+ setMensaje(in mensaje: String)
+ setNuevaContrasena1(in nuevaContrasena1: String)
+ setNuevaContrasena2(in nuevaContrasena2: String)
+ setPersonalId(in personalId: String)
+ validate(in mapping: ActionMapping, in request: HttpServletRequest): ActionErrors

«form»
CambioEmailForm

- email: String
- emailActual: String
- personalId: String
- serialVersionUID: long

+ getEmail(): String
+ getEmailActual(): String
+ getPersonalId(): String
+ setEmail(in email: String)
+ setEmailActual(in emailActual: String)
+ setPersonalId(in personalId: String)
+ validate(in mapping: ActionMapping, in request: HttpServletRequest): ActionErrors

Paquete siidcb.personal.action

«action»
PersonalDispatchAction

+ buscar(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ listAll(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ listFoliosProvisionales(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
# unspecified(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ verResumen(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

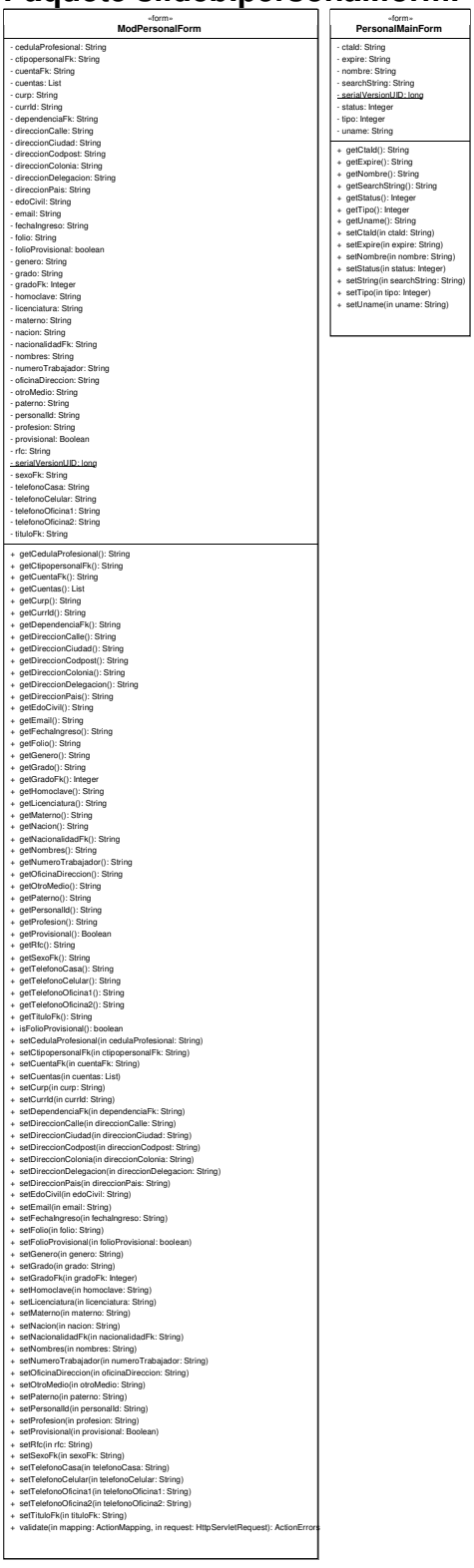
«action»
ModificaDispatchAction

+ borraPersonal(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ borraPersonalConfirmed(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ showModForm(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
# unspecified(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

«action»
GuardaPersonalAction

+ execute(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
- populatePersonal(in f: ModPersonalForm, in p: Personal, in hibernateSession: Session)

Paquete siidcb.personal.form.



Paquete siidcb.planeacion.action.

«action»
PlaneacionFuncDispatchAction

- no: Byte
- si: Byte

+ activarSemestres(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ anadirAsignatura(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ confirmarEliminarBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ crearPrimerBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ eliminarBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ eliminarGrupo(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ getParameterThenAttribute(in request: HttpServletRequest, in name: String, in debug: boolean): Integer
+ guardarGrupo(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ insertarBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ insertarGrupo(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ menuSemestre(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ moverBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ moverGrupo(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ unspecified(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ verAsignaturas(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ verBloques(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ verGrupos(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

«action»
PlaneacionDispatchAction

- no: Byte
- si: Byte

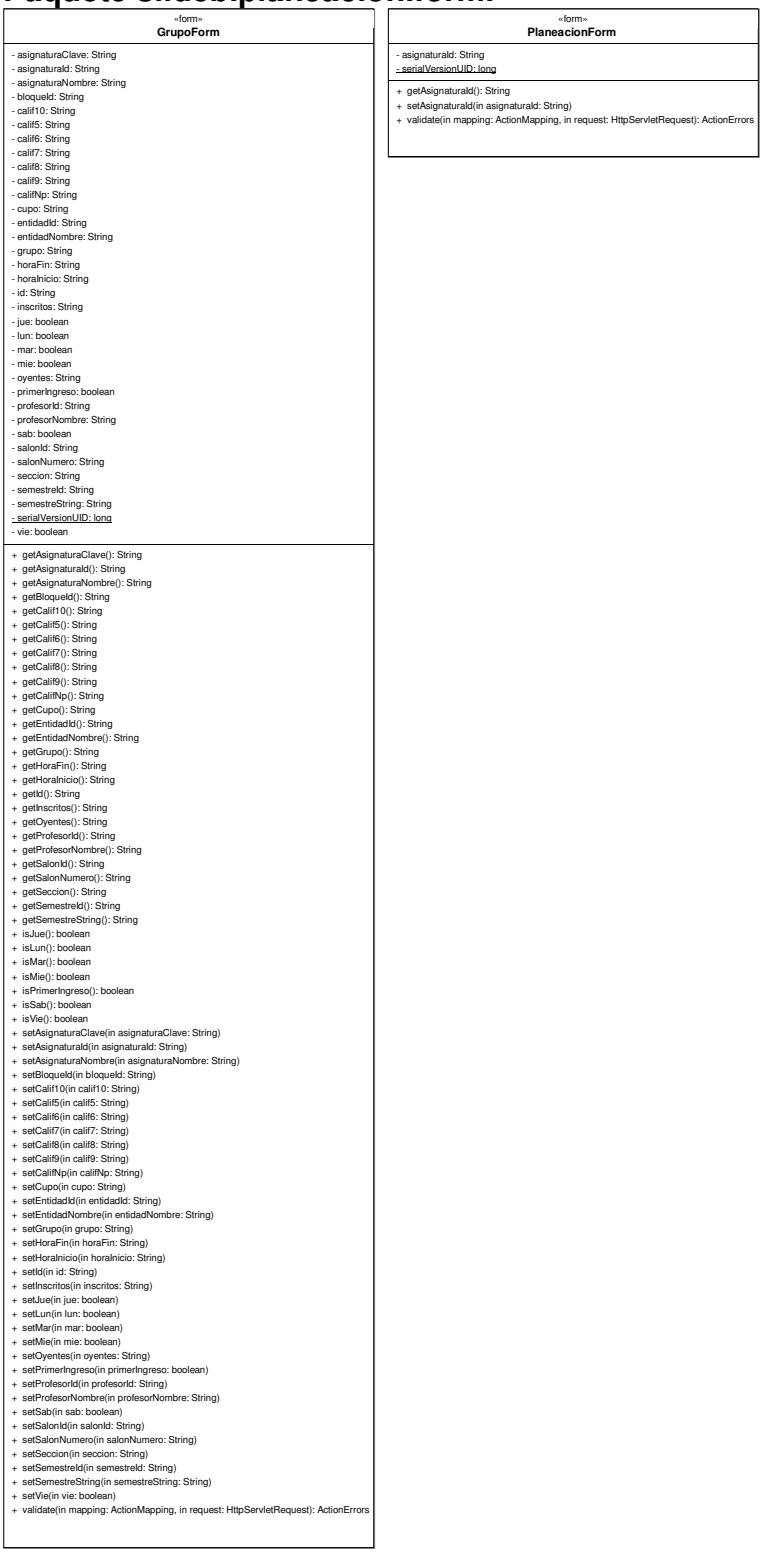
+ activarSemestres(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ anadirAsignatura(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ confirmarEliminarBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ crearPrimerBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ eliminarBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ eliminarGrupo(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ getParameterThenAttribute(in request: HttpServletRequest, in name: String, in debug: boolean): Integer
+ insertarBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ insertarGrupo(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ menuSemestre(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ moverBloque(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ moverGrupo(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ unspecified(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ verAsignaturas(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ verBloques(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward
+ verGrupos(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

«action»
GuardarGrupoAction

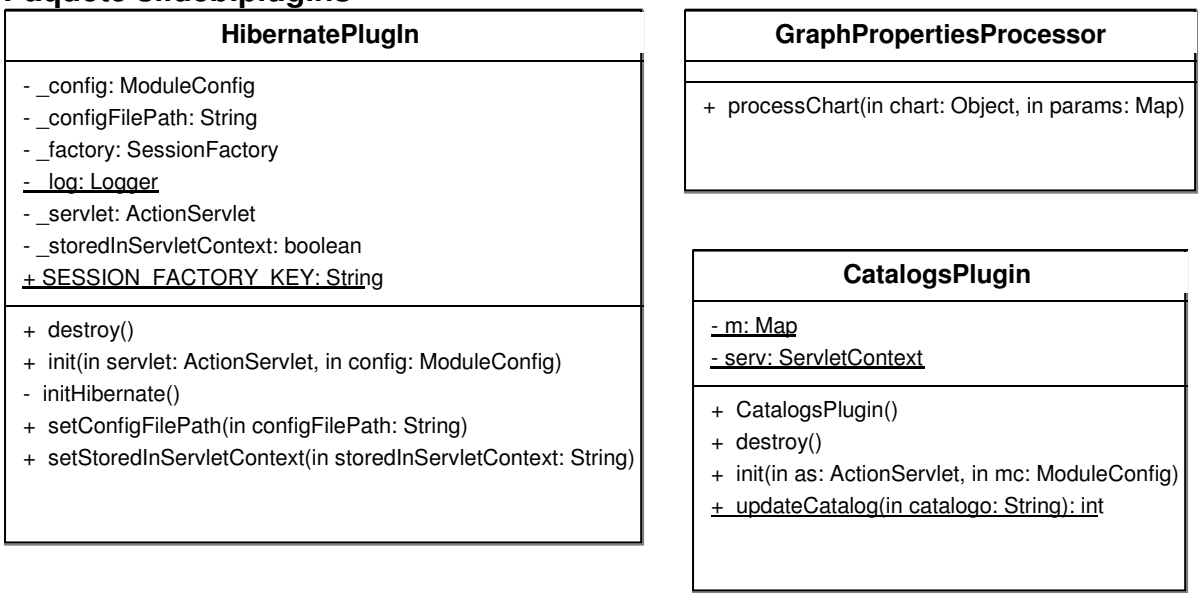
- no: Byte
- si: Byte

+ execute(in mapping: ActionMapping, in form: ActionForm, in request: HttpServletRequest, in response: HttpServletResponse): ActionForward

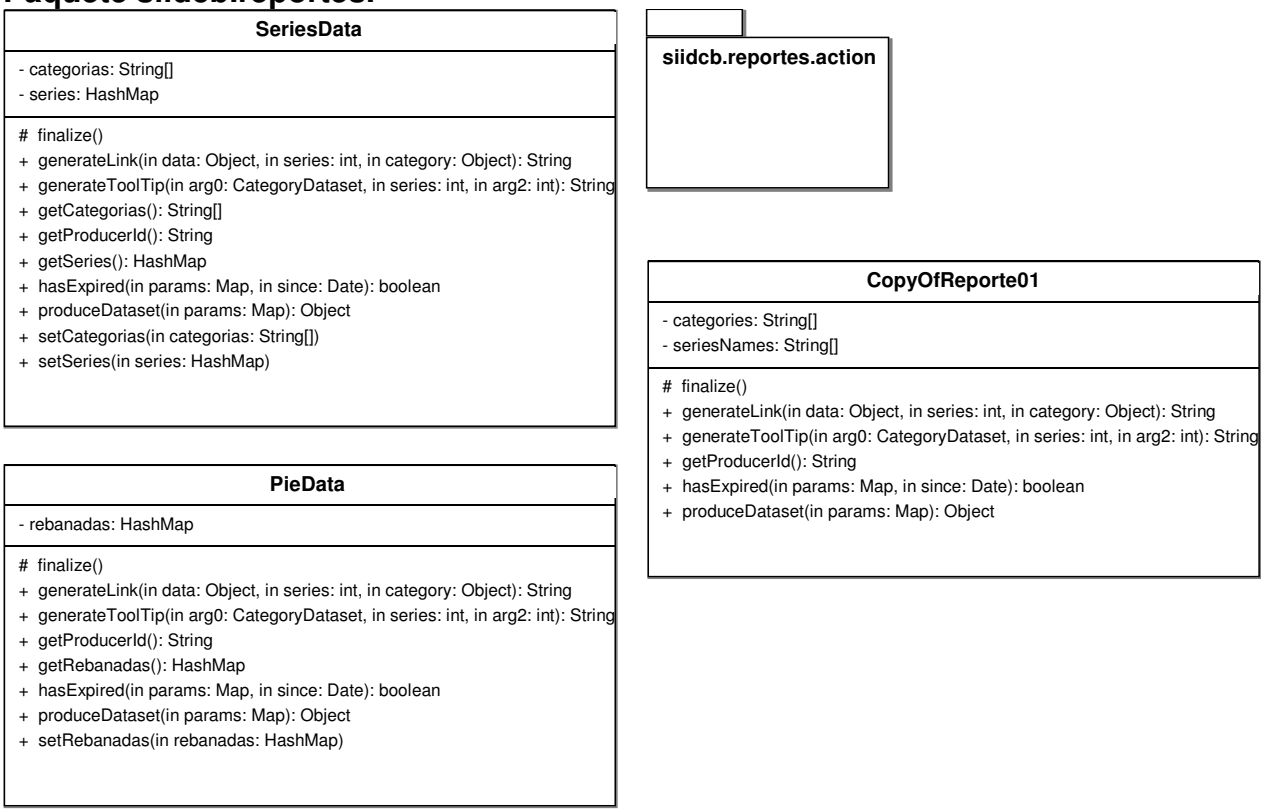
Paquete siidcb.planeacion.form.



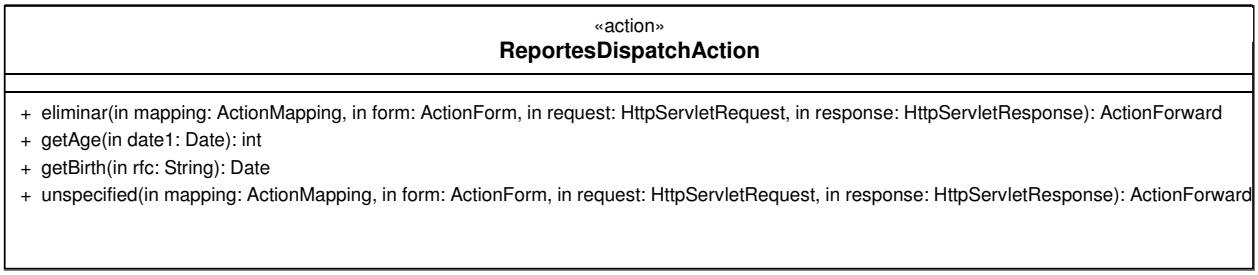
Paquete siidcb.plugins



Paquete siidcb.reportes.



Paquete siidcb.reportes.action.



Anexo D. Instrucciones de configuración de MyEclipse Database Explorer.

1. Abrir la perspectiva "MyEclipse Database Explorer": dar clic en el botón de perspectivas  y seleccionar "MyEclipse Database Explorer". Del lado izquierdo de la ventana de Eclipse aparecerá un menú que dice "DB Browser". Dentro de ésta se listarán las bases de datos que el *plugin* MyEclipse Database Explorer tiene configuradas.

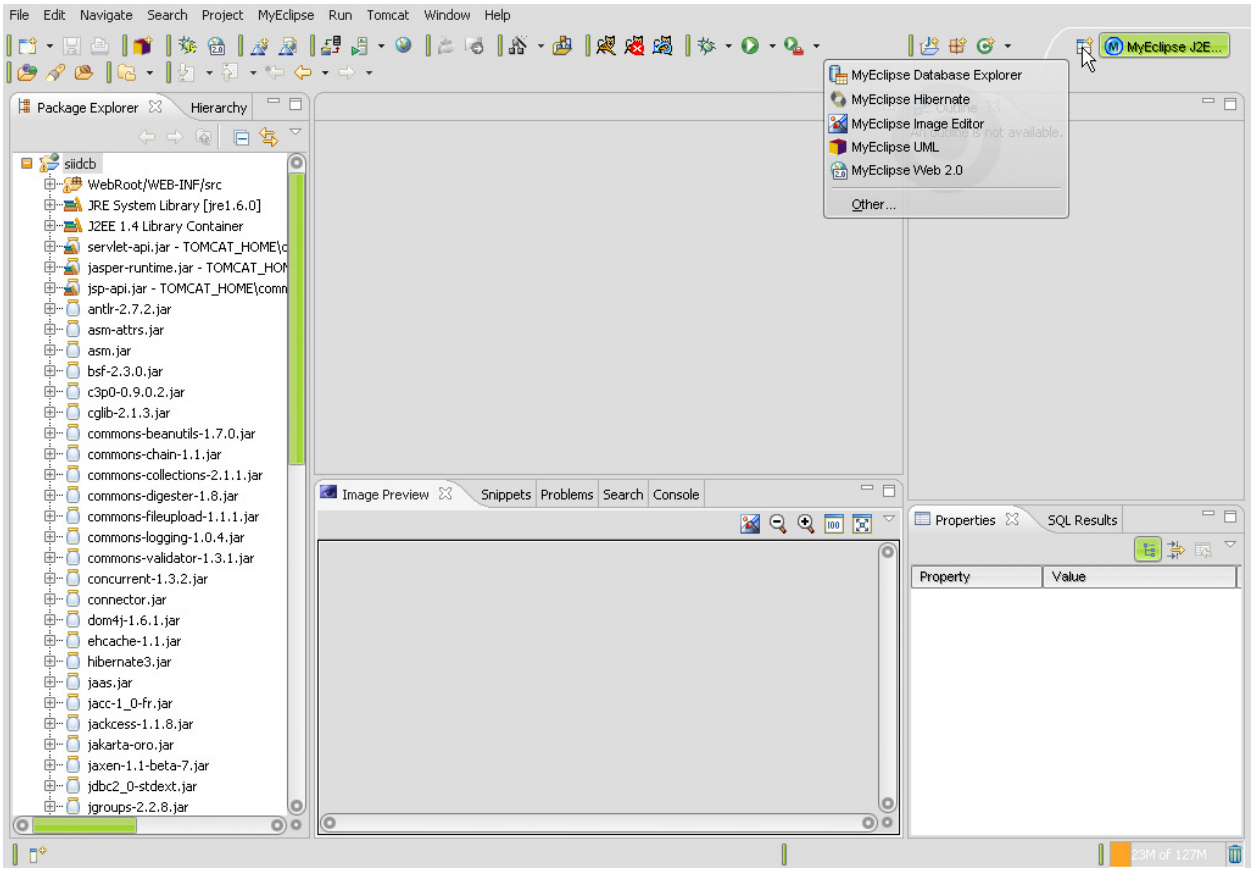


Figura 6—24 Eclipse con la perspectiva "MyEclipse J2EE Development" abierta.

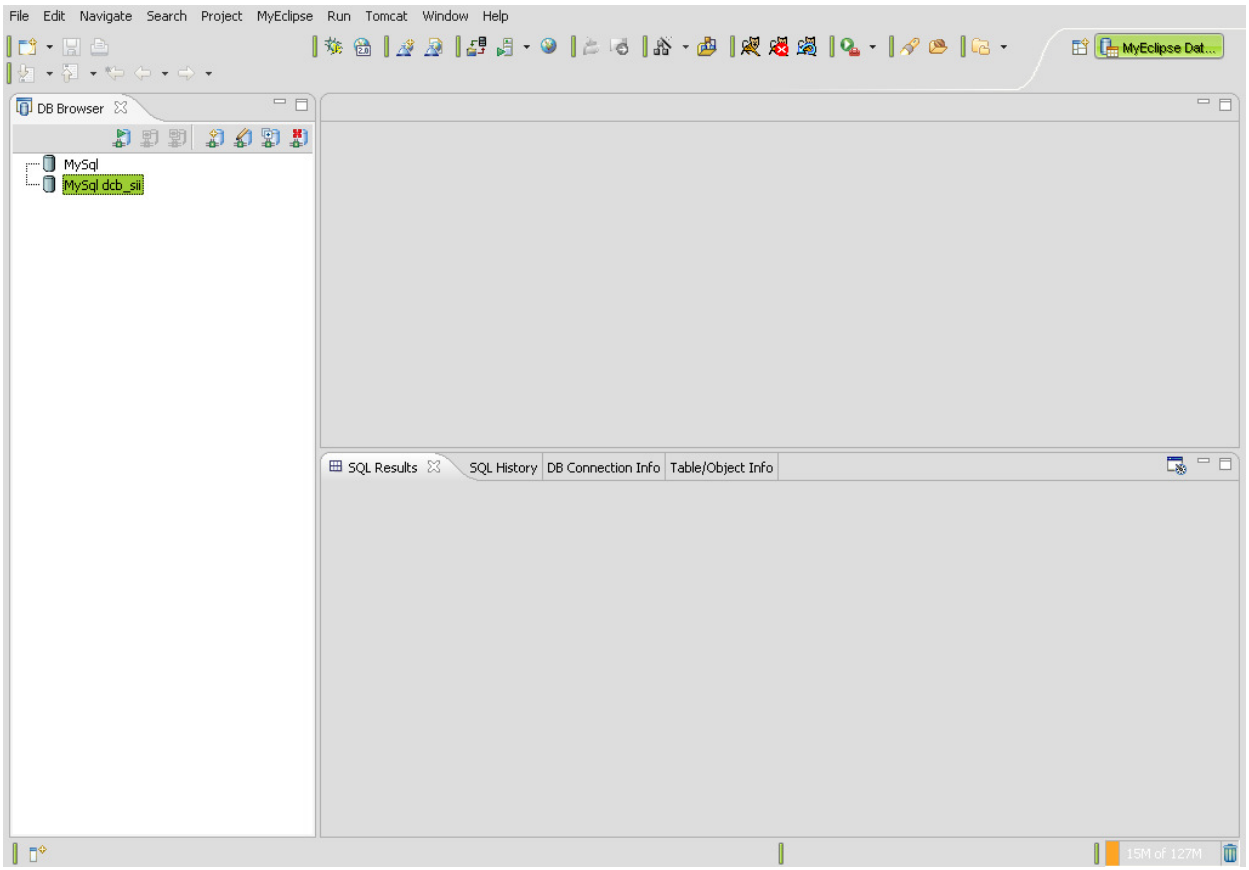


Figura 6—25 Eclipse con la perspectiva "MyEclipse Database Explorer" abierta.

En la figura, del lado izquierdo se ve el listado de bases de datos configuradas para el *plugin*.

2. Creación de un nuevo perfil de conexión a una base de datos: en la parte superior del listado de bases de datos (que puede estar vació) se observan varios botones. Éstos son para conectarse a una base configurada, cerrar una conexión, cerrar todas las conexiones, crear, editar, copiar o borrar un perfil de conexión. Para crear un nuevo perfil dar clic en el botón "New" que tiene el siguiente ícono:  . Aparecerá una ventana que pide los datos de conexión y el controlador a utilizar.

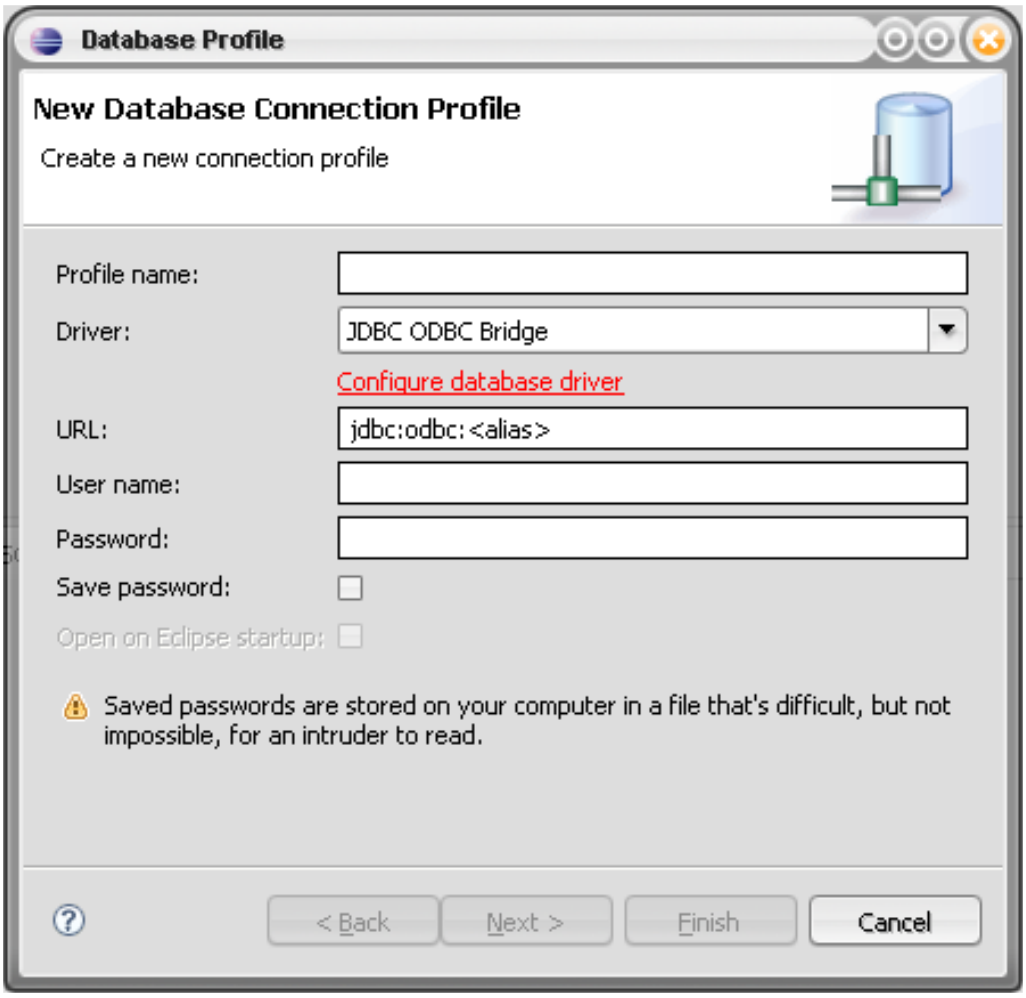


Figura 6—26 Ventana que solicita los datos de la conexión.

Ingresar en dicha ventana el nombre que se dará al perfil, el controlador que utilizará para conectarse a la base de datos, la URL de la base, el nombre de usuario y la contraseña.

Si no se tiene un driver configurado es necesario dar clic en “Configure database driver” y dar uno de alta.

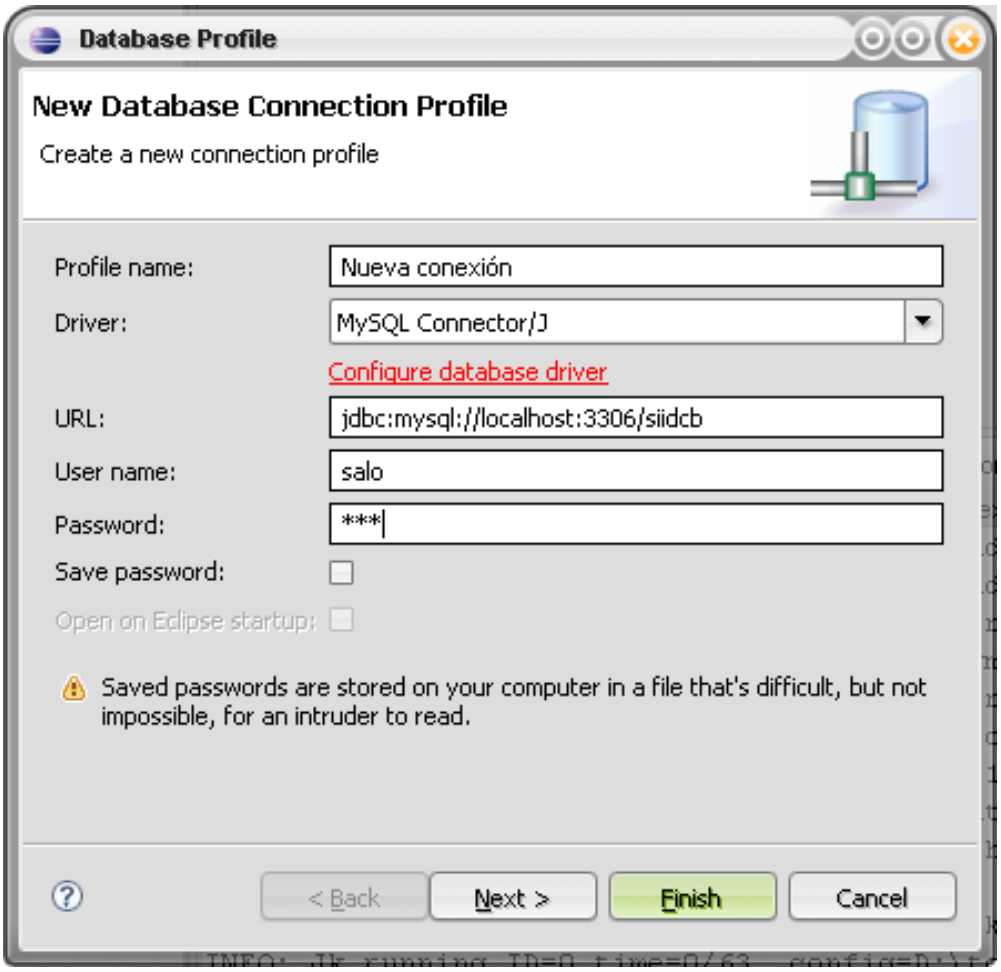


Figura 6—27 La ventana de datos llena.

3. Dar clic en el botón “Finish” para terminar la creación de la conexión. Esta nueva conexión deberá aparecer en el listado.

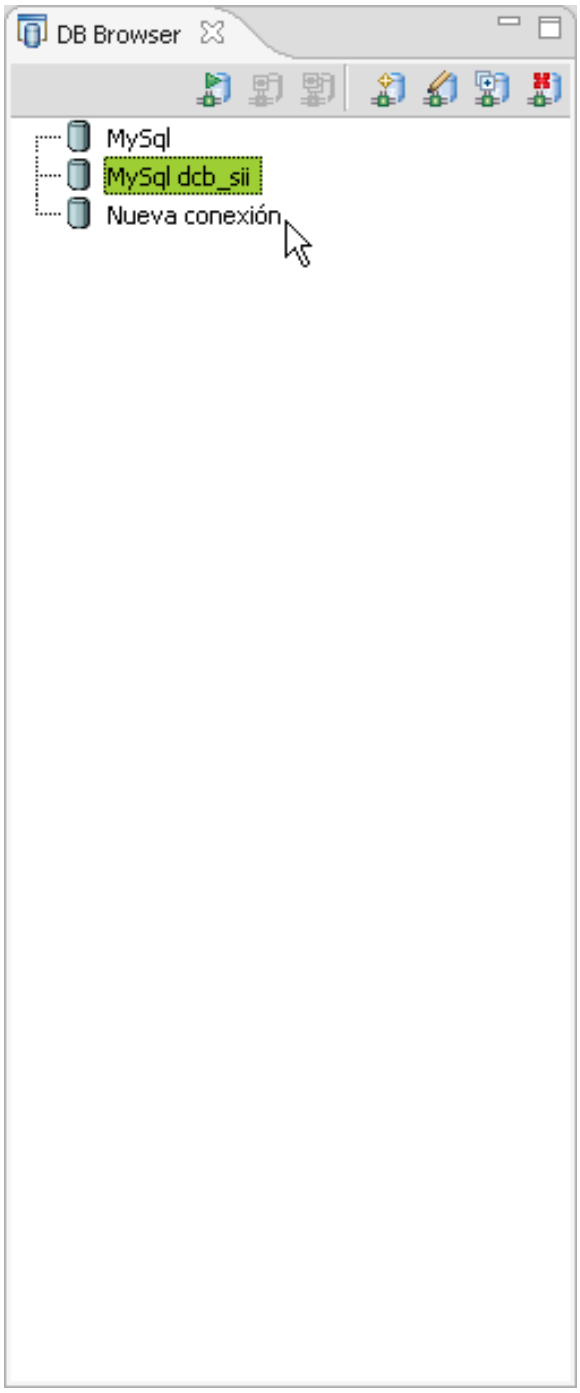


Figura 6—28 Listado que muestra la nueva conexión.

Anexo E. Instrucciones para creación y configuración las clases de modelo.

Para un optimo funcionamiento del sistema es necesario que las clases persistentes en el paquete siidcb.dbase (que constituyen el modelo en el patrón MVC) estén mapeadas correctamente a las tablas existentes en la base de datos. De modo que si se hacen cambios a una tabla como añadirle columnas, es necesario que la clase que modela esa tabla tenga nuevos atributos mapeados a estas nuevas columnas.

Existen dos formas para mapear estos elementos: manualmente o con alguna herramienta que lo haga automáticamente. Para instrucciones de cómo editar los archivos de configuración y crear las clases manualmente, consultar la documentación de Hibernate. A continuación se explica cómo llevar a cabo estas tareas con el plugin para Hibernate incluido en el *workbench* MyEclipse:

1. Abrir la perspectiva “MyEclipse Hibernate”: dar clic en el botón de perspectivas  y seleccionar “MyEclipse Hibernate”. Esto abre varios menús donde se ve la o las conexiones a bases de datos configuradas del lado izquierdo (debe tenerse algún perfil configurado en el MyEclipse Database Explorer. Ver Anexo A).

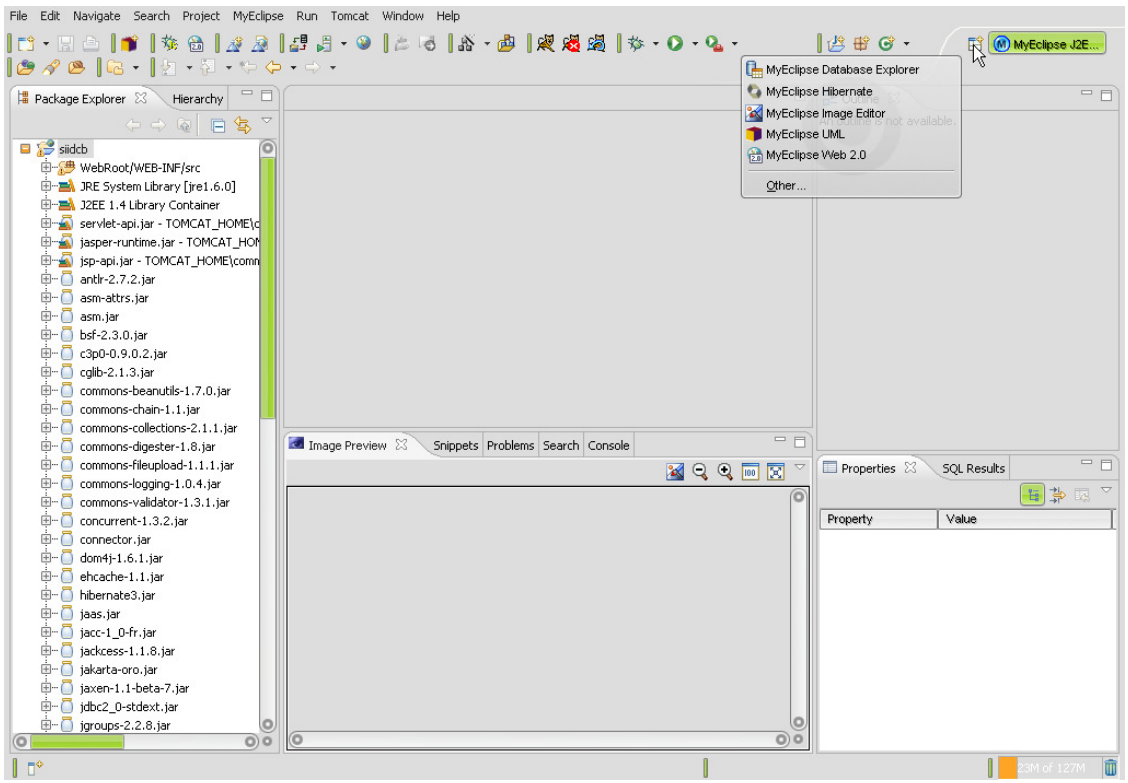


Figura 6—29 Eclipse con la perspectiva “MyEclipse J2EE Development” abierta.

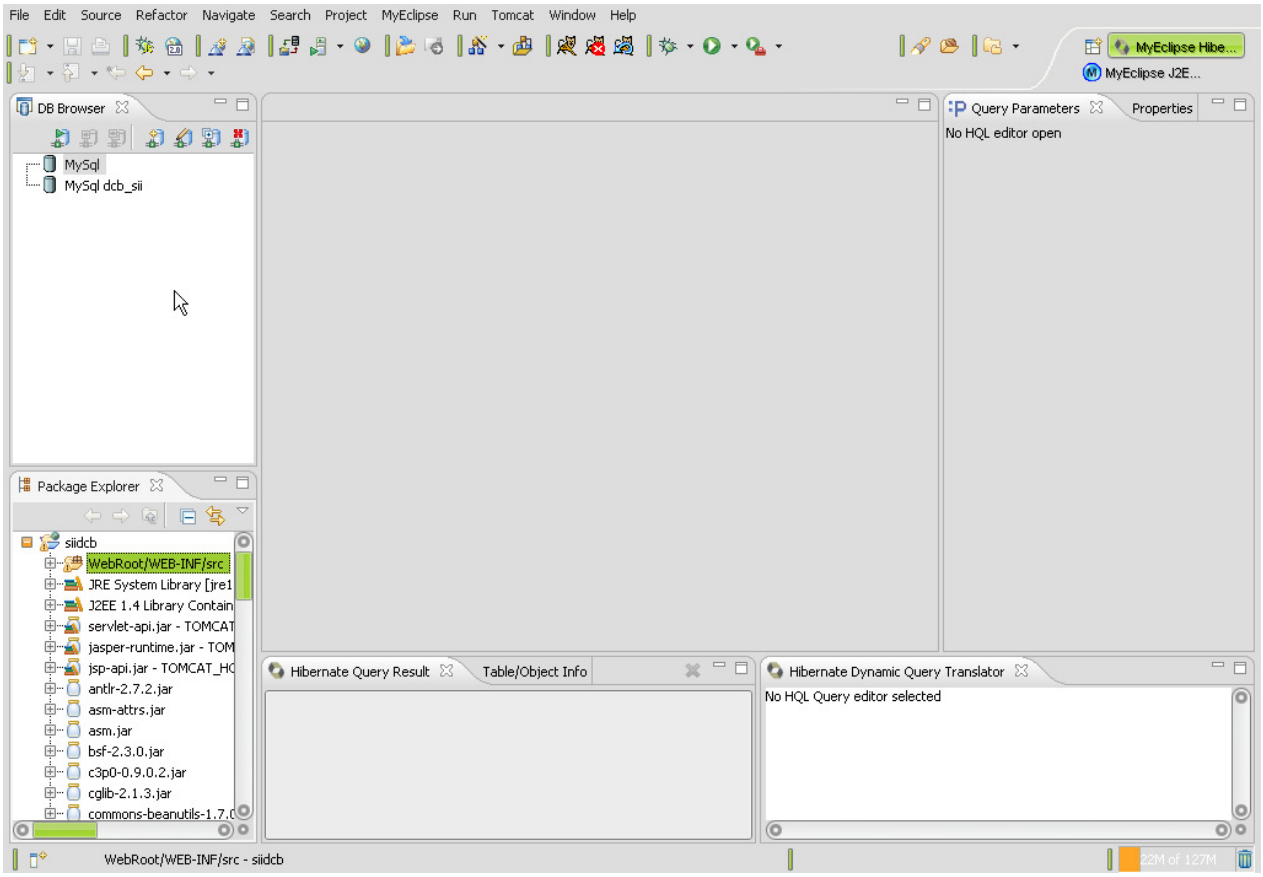


Figura 6—30 Eclipse con la perspectiva "MyEclipse Hibernate" abierta.

2. Abrir la conexión a la base de datos: dar clic derecho sobre la base de datos que se va a trabajar y seleccionar la opción "Open conection...".

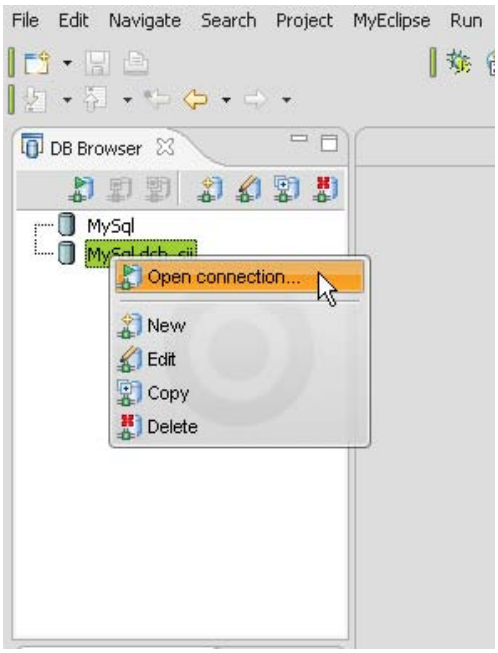


Figura 6—31 Abrir conexión.

Si en la configuración de la base de datos se ha decidido guardar la contraseña, esta operación no pedirá contraseña. De lo contrario debe ingresarse la contraseña de la base abrir la conexión.

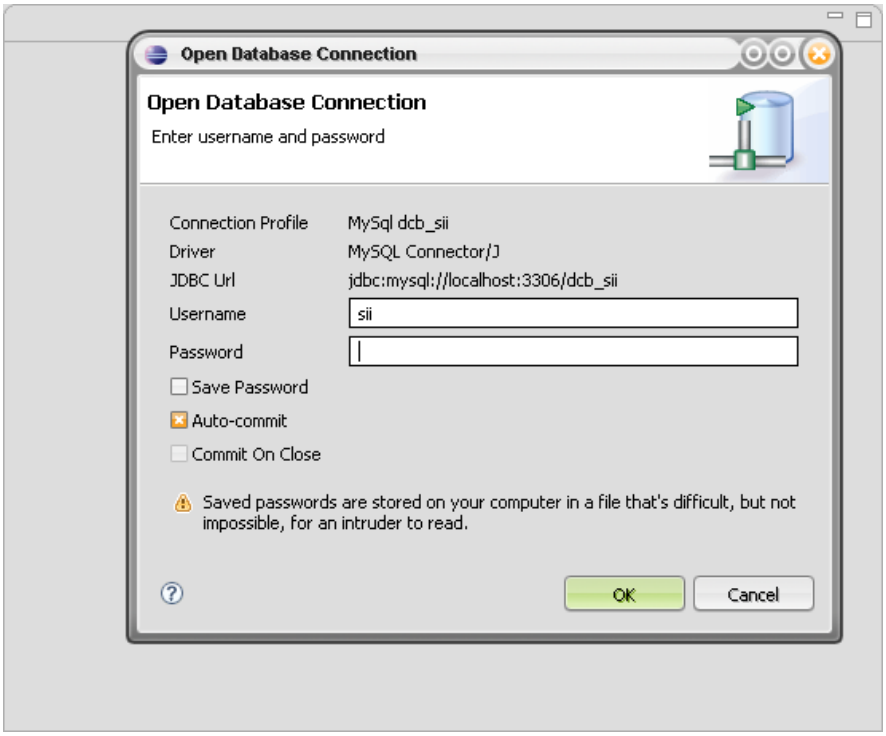


Figura 6—32 Ventana que solicita la contraseña a la base de datos.

3. Seleccionar las tablas que serán mapeadas: una vez abierta la conexión es posible ver las tablas de la base de datos dando clic en los signos '+' como se muestra en la figura.

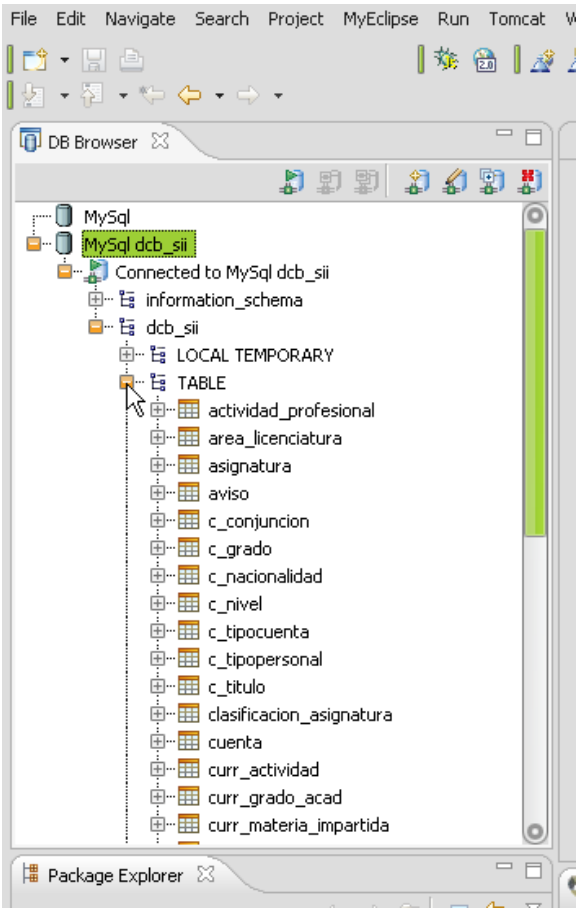


Figura 6—33 Navegación dentro de la base de datos.

4. Hacer ingeniería inversa sobre las tablas para generar las clases del modelo: seleccionar las tablas que serán mapeadas y dar clic derecho sobre alguna de ellas. Esto abre un menú donde aparece la opción "Hibernate Reverse Ingineering...". Esto abrirá una ventana con las opciones de mapeo.

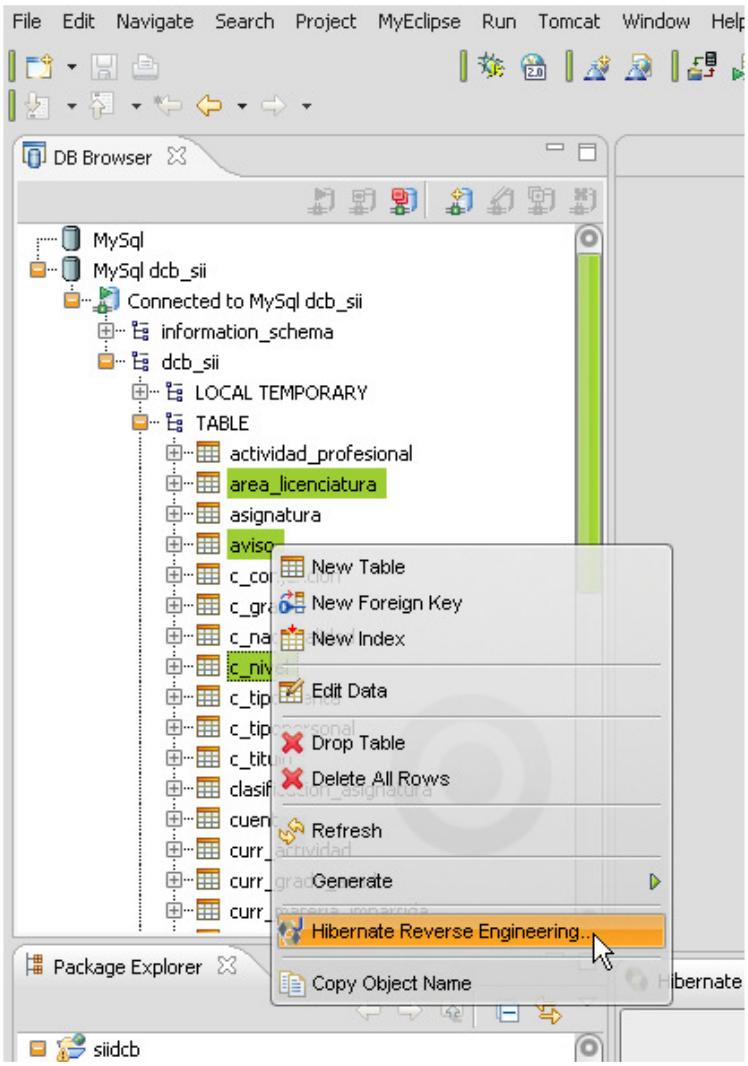


Figura 6—34 Hacer ingeniería inversa sobre las tablas seleccionadas.

5. Especificar opciones: para especificar el directorio donde se encuentra el código fuente de la aplicación se debe dar clic en “Browse” y seleccionar el directorio correspondiente (en este caso /siidcb/WebRoot/WEB-INF/src). Los archivos de configuración se guardan en el mismo directorio del paquete especificado (en este caso es el paquete siidcb.dbase) y son archivos XML editables. Marcar la opción “Hibernate mapping file (*.hbm.xml) for each database table” para generar un archivo por tabla en lugar de una archivo con todos los mapeos. También marcar la opción “Update hibernate configuration with mapping file location” para que el archivo de configuración de Hibernate sea actualizado acerca de los nuevos mapeos. Marcar la opción “Java Data Access Object (DAO) (Hibernate 3 only)” para crear los objetos de acceso.

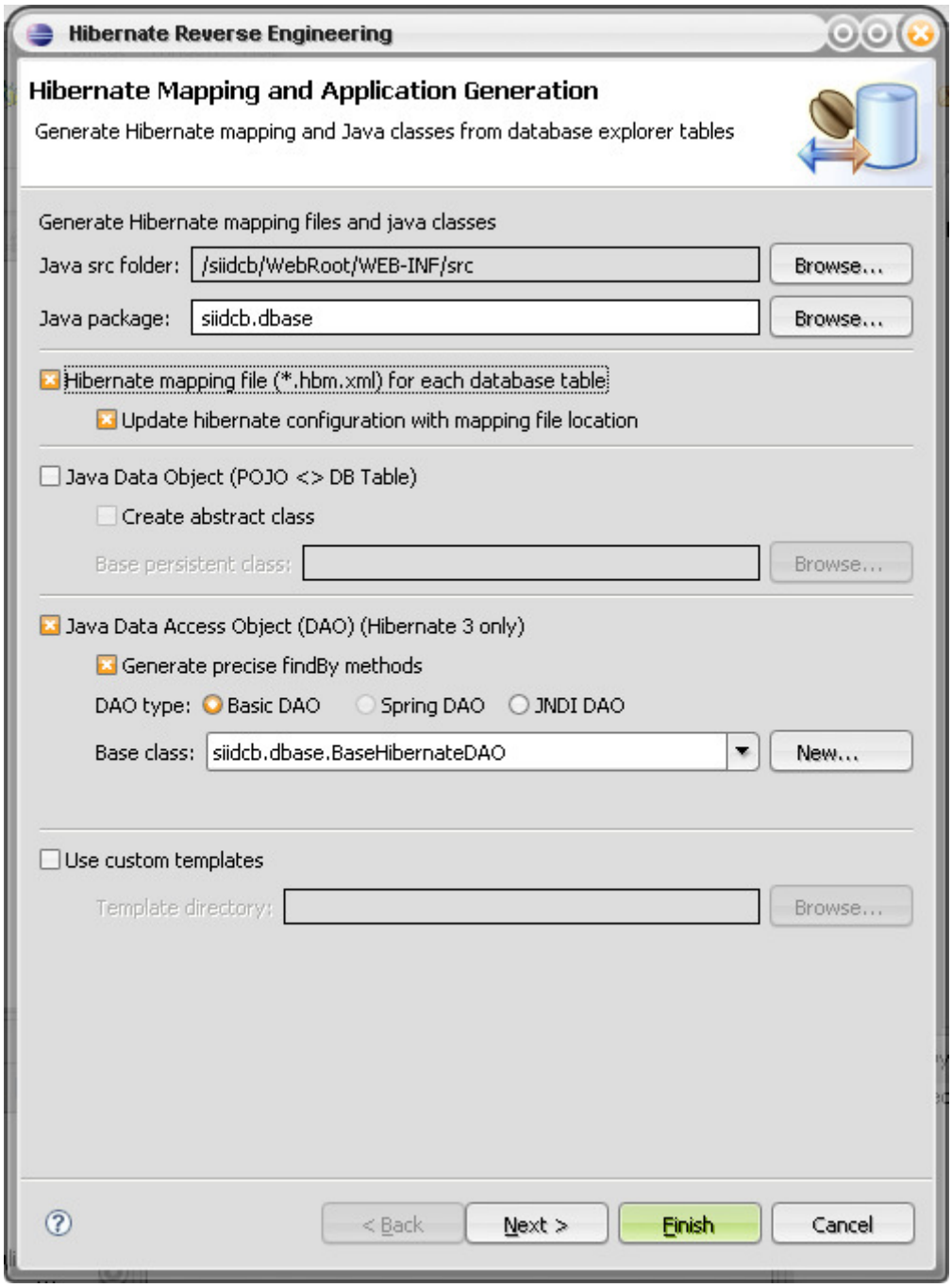


Figura 6—35 Opciones básicas de mapeo de las tablas.

6. Dar clic en el botón “Finish” para crear los objetos en el paquete especificado. Con estos objetos podemos acceder a las tablas correspondientes desde cualquier clase de la aplicación.

Nota: Es importante borrar las definiciones de tablas inexistentes en la base datos del archivo de configuración de Hibernate (/siidcb/WebRoot/WEB-INF/src/hibernate.dfg.xml)

por ejemplo, cuando se elimina una tabla. También sus referencias a esta en las definiciones de otras tablas (estas referencias existen cuando hay llaves foráneas).

Anexo F. Manual del administrador.

SIIDCB.
Manual del Administrador.
Enero de 2008

SIIDCB es un sistema computacional diseñado para proporcionar una interfaz amigable para acceder a una base de datos a usuarios que acceden por medio de un navegador web. El presente documento detalla los requisitos del sistema y contiene la información necesaria para ponerlo en marcha.

Requerimientos.

Para poder operar SIIDCB es necesario contar con los siguientes servicios:

- Apache Tomcat Web Server versión 5 en adelante como contenedor de Servlets para albergar la aplicación.
- MySQL versión 5 en adelante para la base de datos para almacenar los contenidos de la aplicación.

Tanto el servidor de base de datos MySQL como el servidor web Tomcat están disponibles para sistema operativo Linux y Windows.

Estos dos servicios pueden estar albergados en distintas computadoras servidores pero es necesario asegurarse que la computadora con el servidor web Tomcat tenga autorización para conectarse al servidor de base de datos.

Si ambos servicios se encuentran en la misma computadora servidor por lo general no hay problemas de acceso a MySQL por parte de Tomcat ya que las conexiones desde la dirección del *localhost* (127.0.0.1) por default se autorizan. Si usted no administra el servicio de base de datos y tiene dudas en este respecto acuda al administrador para cualquier aclaración.

Instalación, configuración y puesta en marcha.

Creación de la base de datos en el servidor y asignación de permisos de acceso.
Para crear bases de datos en el servidor MySQL, el usuario *root* o algún otro usuario con los permisos necesarios debe crear la base de datos ‘dcb_sii’ y otorgar permisos a un usuario sobre ésta. Los datos de ese usuario (nombre y contraseña) se usarán en la aplicación para configurar el acceso a la base de datos creada.

Con el comando "*create database*" seguido por el nombre de la base a crear se crea la base. Una vez creada, MySQL muestra el mensaje "Query OK".

```
mysql> create database dcb_sii;
Query OK, 1 row affected (0.03 sec)

mysql>
```

A continuación es necesario otorgar acceso total a esta base de datos recién creada a una cuenta que ocupará SIIDCB. Esto se hace con el comando 'grant'.

```
mysql> grant all on dcb_sii.* to 'sii' identified by 'password';
```

En el código mostrado se otorgan todos los privilegios (incluyendo creación y destrucción) sobre la base de datos dcb_sii al usuario 'sii' si accede al servidor con la contraseña 'password'. Ahora el usuario 'sii' puede crear y destruir tablas, modificarlas, etc. También puede utilizar el comando 'drop database' para eliminar la base datos y volver a crearla vacía con el comando 'create database'.

En este punto es posible especificar desde qué dirección IP se conectará SIIDCB a la base de datos para aumentar el grado de seguridad del sistema. Ya que SIIDCB sólo accederá desde una dirección IP podemos asignar permisos a la base de datos especificando dirección IP:

```
mysql> grant all on dcb_sii.* to 'sii'@'132.248.139.67' identified by 'password';
```

También podemos usar el comodín '%' para la dirección y así permitir el acceso desde todas las direcciones IP de la red local con '132.248.139.%'.

Despliegue de la aplicación en el contenedor de Servlets/servidor web Tomcat.

Para instalar SIIDCB en el servidor Tomcat, es necesario copiar el archivo WAR de la aplicación en el subdirectorio '*webapps*' del directorio donde se encuentra instalado Tomcat. Una vez copiado se reinicia el Tomcat y éste extrae del archivo 'siidcb.war' los archivos de la aplicación y los coloca en un directorio llamado siidcb; también intentará iniciar la aplicación, pero muy probablemente marcará algún error si aún no está debidamente configurada. A continuación se explica qué es necesario configurar para el funcionamiento correcto de SIIDCB.

Configuración de SIIDCB para acceso a la base de datos.

Ya que SIIDCB utiliza Hibernate para acceder a la base de datos, es en la configuración de este servicio donde deben especificarse algunos datos para la conexión. Para configurar Hibernate es necesario editar el archivo /WEB-INF/src/hibernate.cfg.xml.
En este archivo, en el elemento *session-factory* dentro del elemento *hibernate-configuration*, deben definirse los siguientes parámetros:

```
<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="connection.url">jdbc:mysql://localhost:3306/dcb_sii
    </property>
<property name="connection.username">sii</property>
<property name="connection.password">password</property>
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
```

En la propiedad 'dialect' se especifica al servicio Hibernate (que forma parte de la aplicación) que debe usar el dialecto propio de una base de datos MySQL.

En la propiedad 'connection.url' se indica el servidor, el puerto y la base de datos al que se conectará Hibernate.

En la propiedad 'connection.username' se especifica el nombre de usuario de la cuenta con los permisos necesarios para utilizar la base. Y en la propiedad 'connection.password' la contraseña de dicha cuenta.

En la propiedad 'connection.driver_class' se indica la clase del driver de JDBC para bases de datos MySQL.

Nota importante.

Existe una situación con un plugin de Tiles (parte del marco de trabajo Apache Struts) que utiliza SIIDCB en la que el sistema no inicia si no se encuentra cierto archivo: es esencial especificar en la cabecera del archivo /WEB-INF/tiles-defs.xml la ubicación de su archivo descriptor (Document Type Definition (DTD)).

Éste archivo descriptor se especifica en la etiqueta: <!DOCTYPE tiles-definitions>
Las posibles opciones a esta ubicación son:

a) Una ubicación en internet (generalmente en un sitio de Apache):

```
<!DOCTYPE tiles-definitions PUBLIC
   "-//Apache Software Foundation//DTD Tiles Configuration//EN"
   "http://jakarta.apache.org/struts/dtds/tiles-config.dtd">
```

ATENCIÓN: Se corre el riesgo de que si Tomcat no tiene acceso a la red (y por lo tanto al archivo), la SIIDCB no podrá iniciar.

b) Alguna ubicación en un sistema de archivos local (**recomendado**). Los siguientes son tres ejemplos.

```
<!DOCTYPE tiles-definitions SYSTEM "file:///dtds/tiles-config.dtd">
<!DOCTYPE tiles-definitions SYSTEM "file://c:/dtds/tiles-config.dtd">
<!DOCTYPE tiles-definitions SYSTEM "tiles-config_1_1.dtd">
```

En este último caso el archivo puede variar de ubicación dependiendo del sistema operativo. En Linux, Tomcat busca el archivo dentro del directorio donde se encuentran los ejecutables de Tomcat.

Bloqueo y autorización de direcciones IP's.

SIIDCB puede configurarse para limitar el acceso sólo a ciertas direcciones IP. Para esto se debe editar el archivo /WEB-INF/src/ApplicationResources.properties.

Para especificar las direcciones IP que tendrán acceso a SIIDCB, en el archivo de configuración mencionado hay que ubicar la propiedad **validIps** y a continuación del signo igual escribir, separadas por comas, las direcciones autorizadas.

Para bloquear el acceso desde ciertas direcciones IP debe editarse la propiedad **invalidIps**, y al igual que en el caso anterior, escribir, separadas por comas, las direcciones bloquedas.

Tanto en la especificación de IP's autorizadas como para las bloqueadas puede utilizarse el comodín asterisco en uno de los 4 números decimales de la dirección. Por ejemplo: **validIps=132.248.139.*** permite que cualquier computadora cuya dirección IP comience con 132.248.139 y termine con cualquier valor pueda iniciar sesión en SIIDCB.

Así, **validIps= *.*.*.*** permite el acceso desde cualquier dirección.

Cuando un cliente desea iniciar sesión, SIIDCB verifica que su dirección no esté incluida en las direcciones bloqueadas. Si está, aunque se encuentre en las direcciones autorizadas no se le permitirá en acceso. Por ejemplo, si un cliente desde la dirección 132.248.139.67 intenta acceder y en el archivo de configuración está "**validIps=132.248.139.*, 192.168.139.***" y "**invalidIps=132.248.139.92**", aunque el cliente posee una dirección que está incluida en las IP's autorizadas, al estar también en las bloqueadas no se le permite el acceso. Esto es una manera de filtrar las direcciones de una manera sencilla y permitir usar el comodín asterisco para indicarlas.

Puesta en marcha de la aplicación.

Una vez que se ha configurado SIIDCB correctamente, la siguiente vez que Tomcat sea iniciado la aplicación iniciará en el contexto "/siidcb" es decir que la URL para acceder a SIIDCB será la dirección del servidor que lo contiene seguido de "/siidcb". Por ejemplo "http://basicas.fi-c.unam.mx:9090/siidcb".

Archivos de bitácoras (logs) de SIIDCB.

Para ver los mensajes de información y de error del servidor web y de SIIDCB debe consultarse el archivo "catalina.out" dentro del directorio "logs" de Tomcat. En un sistema Linux se recomienda usar el comando "tail" para poder leer el archivo y ver los cambios que se hacen a este al momento mismo en que ocurren. Esto se logra con el parámetro "–F" (*follow*). Por ejemplo:

```
tomcat@basicas:~$ cd logs
tomcat@basicas:~/logs$ tail -F catalina.out
```

Este archivo debe ser consultado en caso de mal funcionamiento del sistema ya que aquí se despliega la información de excepciones de la aplicación e información localización de errores (*debugging)*. Este es un archivo esencial para las tareas de mantenimiento.

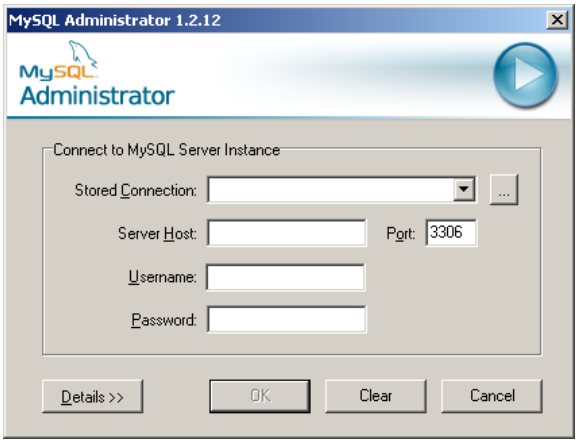
Mantenimiento de la base de datos de SIIDCB por medio de MYSQL Administrator.

A continuación se muestra como llevar a acabo algunas de las tareas de mantenimiento de la base de datos con la herramienta MySQL Administrator disponible gratuitamente en la página de MySQL (http://www.mysql.com/).

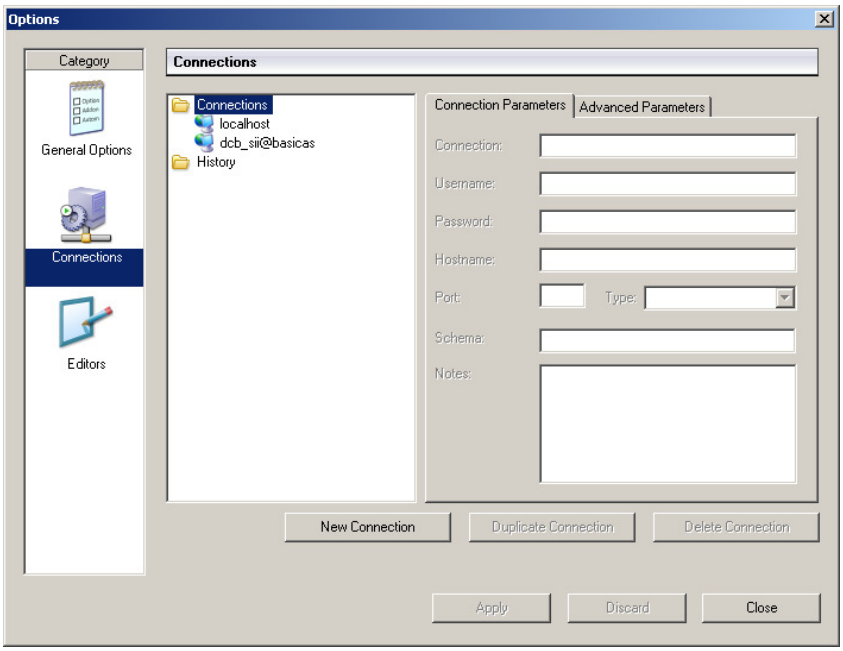
Una vez instalado el programa, lo primero que hay que hacer es crear un perfil de conexión a la base de datos de SIIDCB.

Creación de un perfil de conexión a una base de datos.

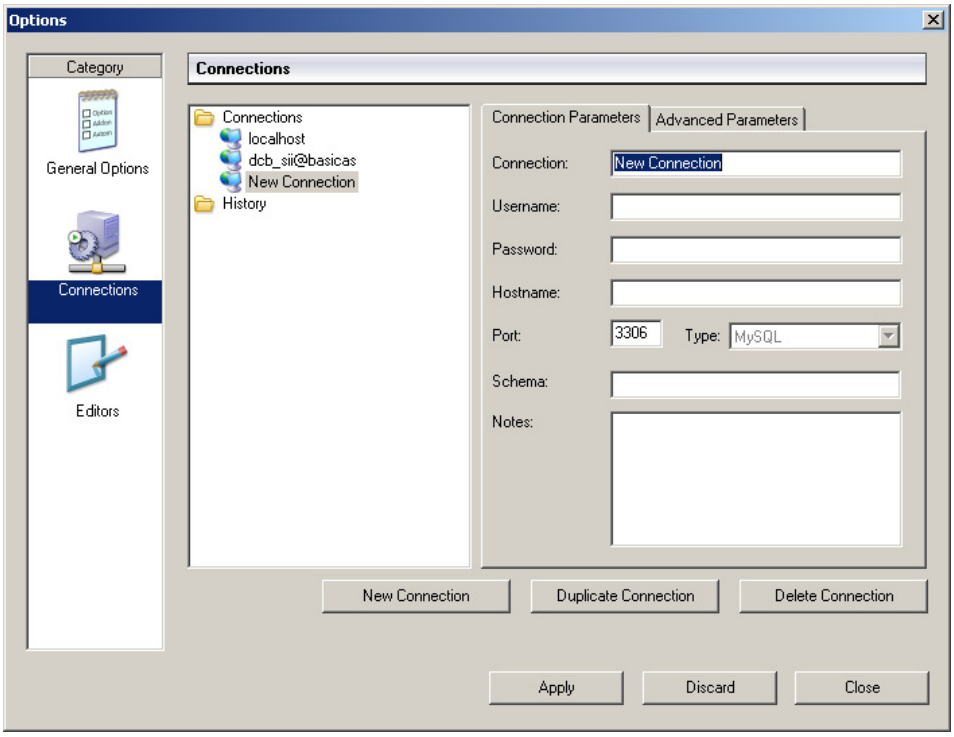
Al iniciar MySQL Administrator se muestra una pequeña ventana para abrir una conexión de base da datos. En ésta se ven los perfiles de conexión guardados previamente y se puede seleccionar uno en el combo indicado como "Stored Connection". A la derecha de dicho combo existe un botón con tres puntos.



Primero debe hacerse clic en el botón que tiene tres puntos. Con esto se muestra una ventana con las opciones del programa.



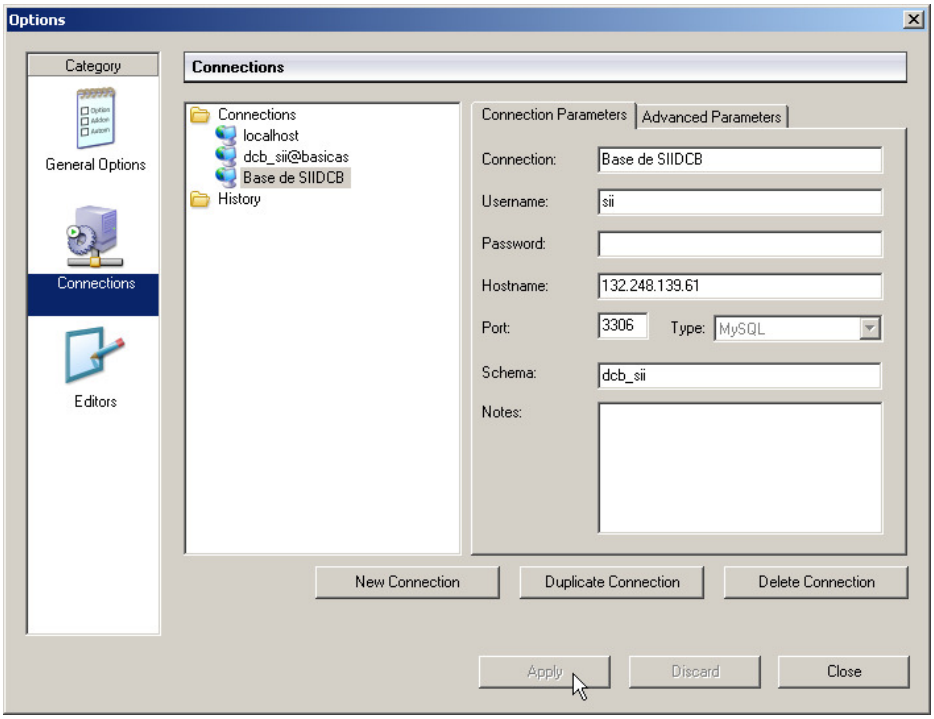
En la sección de conexiones (Connections) se muestran los perfiles de conexión existentes y un botón y un botón que dice “New Connection”. Hacer clic en ese botón.

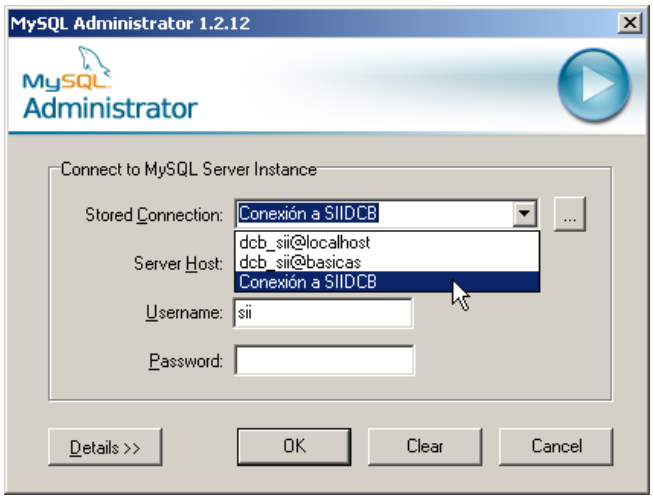


El cursor se colocará en el campo “Connection”, ahí debe especificarse un nombre para identificar el perfil de conexión. Por ejemplo “Base de SIIDCB”.

En el campo “Username” se escribe el nombre de usuario de la cuenta de la base de datos y en el campo “Password” la contraseña si quiere que sea guardada en el perfil y no se tenga que escribir cada vez que se utilice el perfil de conexión, pero no se recomienda esto así que puede dejarse vació este campo.

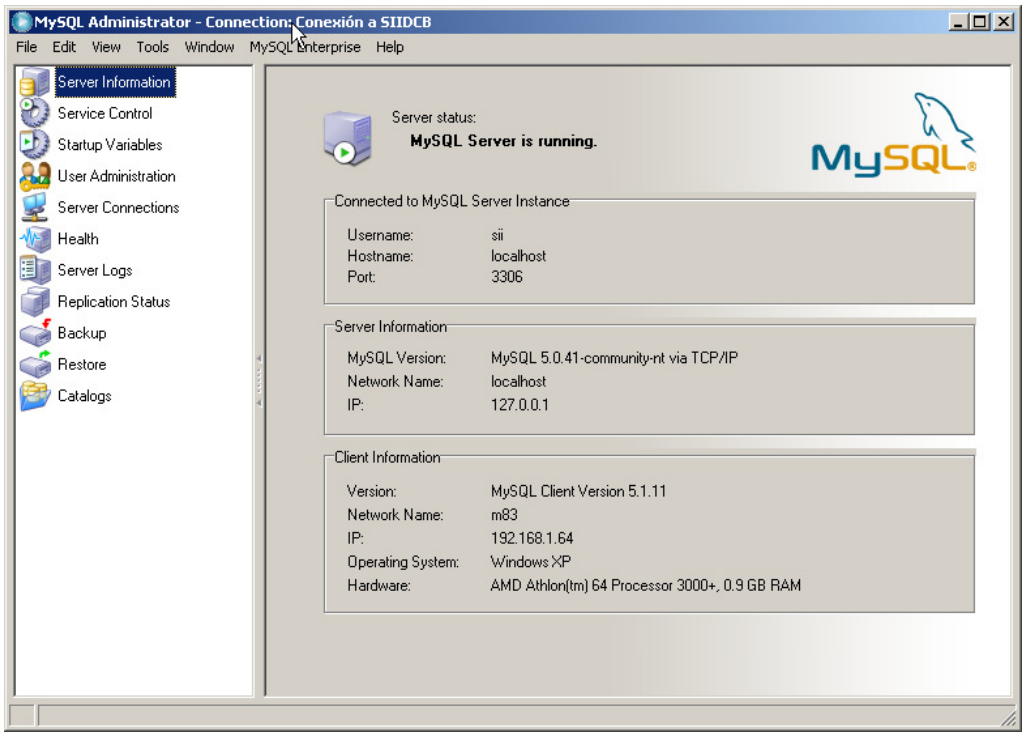
En el campo “Hostname” se ingresa la dirección de la computadora que alberga el servidor de base de datos y en el campo “Port” el puerto que generalmente es el 3306.
En el campo “Schema” se debe poner el nombre de la base de datos, que es “dcb_sii”.
Una vez llenados estos campos debe hacerse clic en el botón “Apply” para guardar los parámetros recién ingresados. Después de haber creado el perfil debemos dar clic al botón “Close” y así regresaremos a la ventana inicial de conexión, donde en la lista de perfiles encontraremos y podremos seleccionar el perfil que acabamos de crear para conectarnos a la base de datos de SIIDCB después de ingresar la contraseña y de dar clic en el botón “OK”.





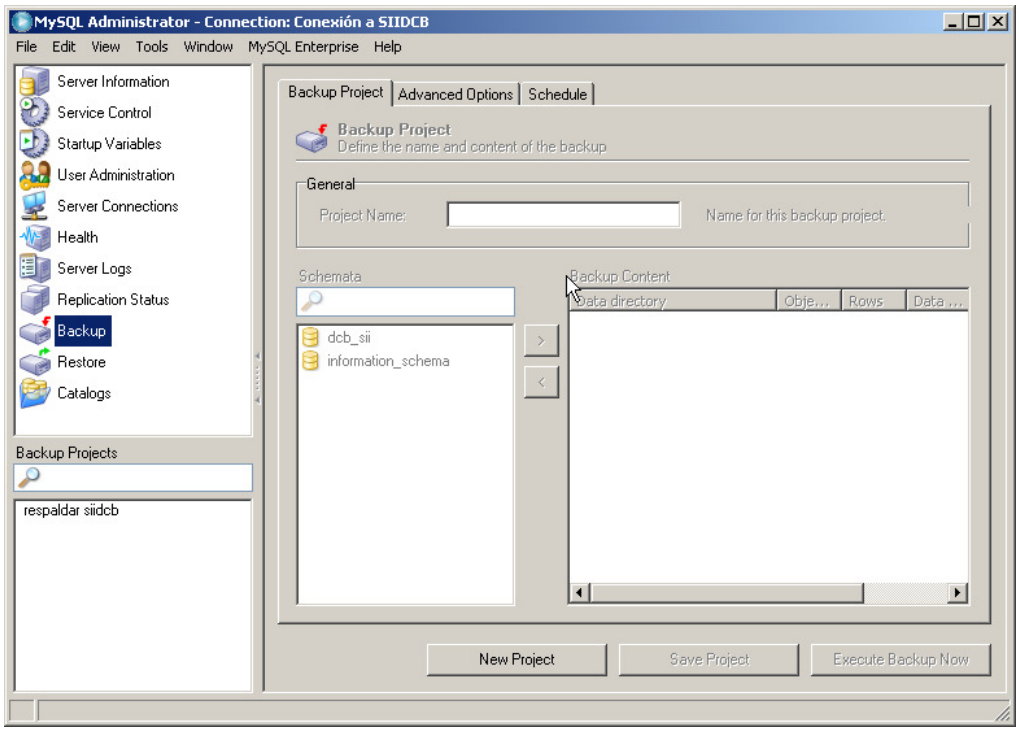
Respaldo y restauración de la base de datos.

A continuación se explica cómo crear archivos de respaldo y hacer restauraciones de la base de datos a partir de éstos por medio de MySQL Administrator.



Una vez conectado a la base de datos, en el menú izquierdo del programa se ven varias opciones, algunas de las cuales sólo están disponibles para el usuario *root*. Para nuestros propósitos sólo necesitamos las tres últimas: “Backup”, “Restore” y “Catalogs”.

Si deseamos crear un archivo de respaldo de la base de datos de SIIDCB es necesario seleccionar la opción “Backup”. Entonces se mostrará la ventana con las opciones de respaldo.

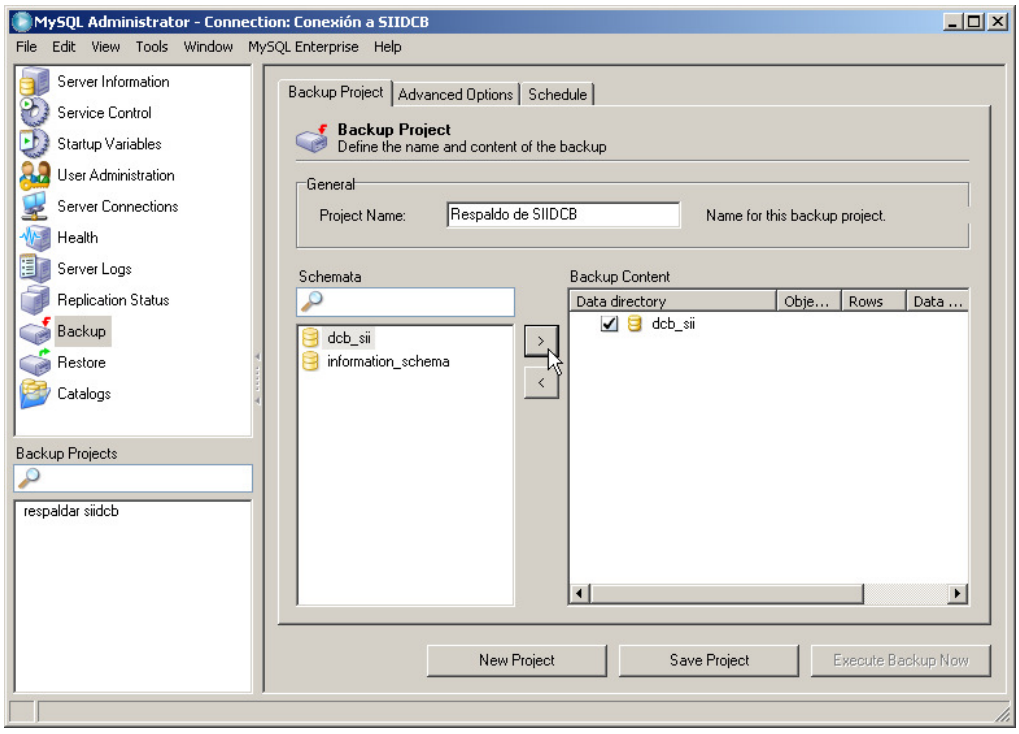


Del lado inferior izquierdo de esta ventana se muestran los proyectos de respaldo que han sido creados y guardados para realizar respaldos rápidamente. Ahora vamos a crear un proyecto de respaldo de la base de datos de SIIDCB para poder hacer lo mismo.

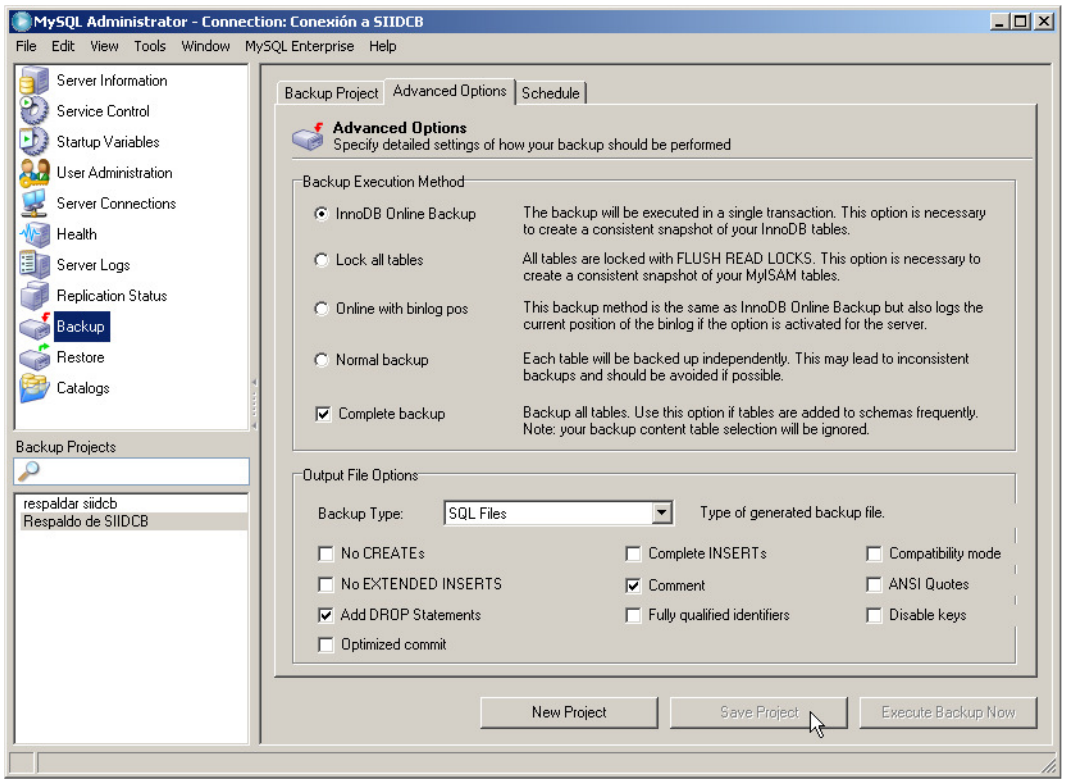
Creación del proyecto de respaldo.

En la ventana de opciones de respaldo debe darse clic al botón “New Project” y se activará entonces el campo “Project name” para asignar un nombre al perfil de respaldo. En este ejemplo le pondremos el nombre “Respaldo de SIIDCB”.

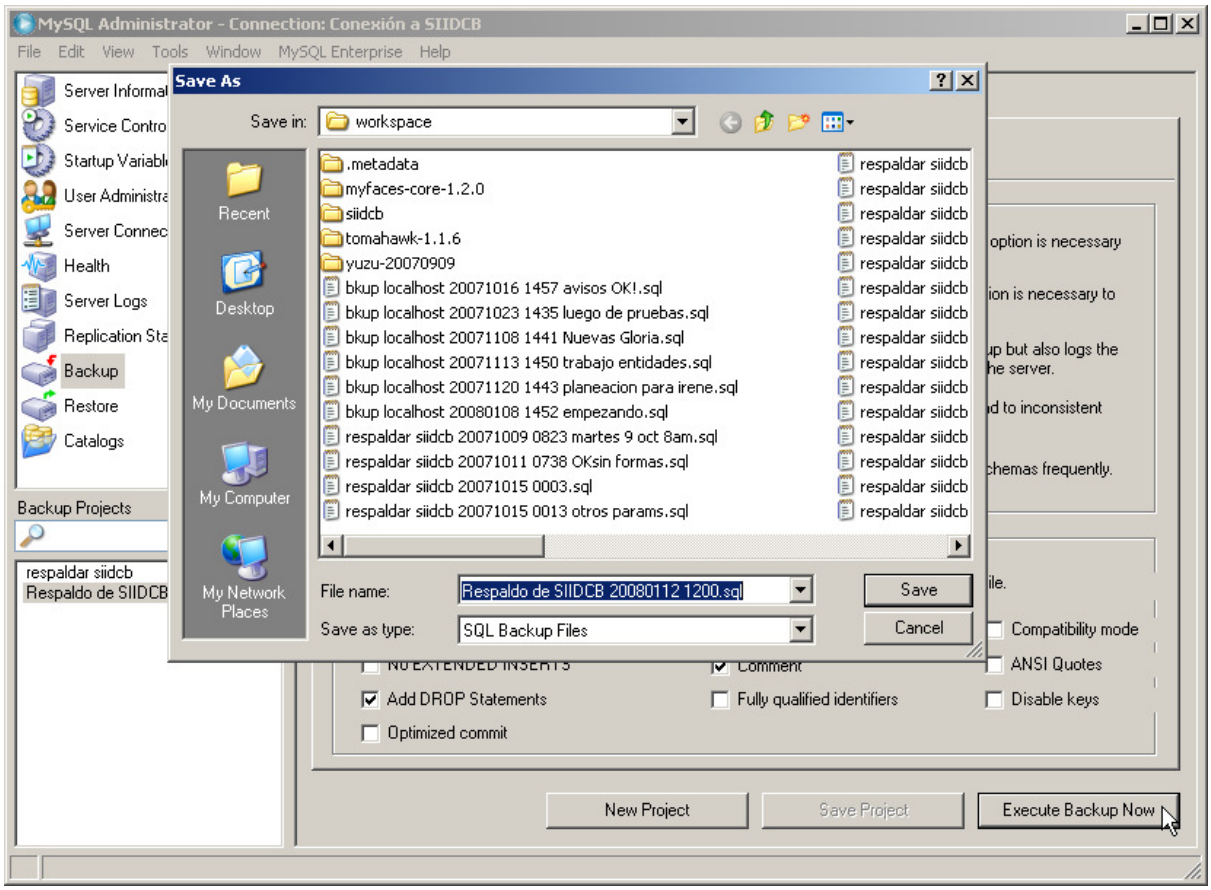
A continuación debemos seleccionar la o las bases de datos que queremos respaldar; en nuestro caso sólo es la de SIIDCB la que respaldaremos. En la parte inferior izquierda de la venta se observa una lista titulada como “Schemata” con las bases de datos existentes en el servidor de base de datos, ahí encontraremos la base de datos de SIIDCB (dcb_sii); debemos seleccionarla dando clic sobre ella y luego presionar el botón “>” para añadirla a las bases de datos que serán respaldadas en este proyecto.



Ahora debemos especificar las opciones avanzadas para el respaldo. Para esto debemos dar clic a la pestaña que dice “Advanced Options”y aparecerán las opciones donde debemos seleccionar “InnoDB Online Backup” de las opciones radio y marcar la caja de selección que dice “Complete backup”. Y en las opciones de la parte inferior, marcar solamente “Add DROP statements” y “Comment”. Una vez indicadas todas estas opciones presionamos el botón “Save Project” para guardar nuestro nuevo proyecto de respaldo.



Una vez que el nuevo proyecto de respaldo ha sido guardado, éste aparece listado junto con los proyecto previamente existentes. Para realizar el respaldo de la base de datos con el proyecto que acabamos de crear sólo hace falta verificar que en el listado esté seleccionado y dar clic al botón “Execute backup now”. El programa pedirá una ubicación para guardar un archivo al que pondrá el nombre del proyecto seguido por la fecha y hora en que se realizó el respaldo (este nombre puedce ser cambiado a cualquier otro).

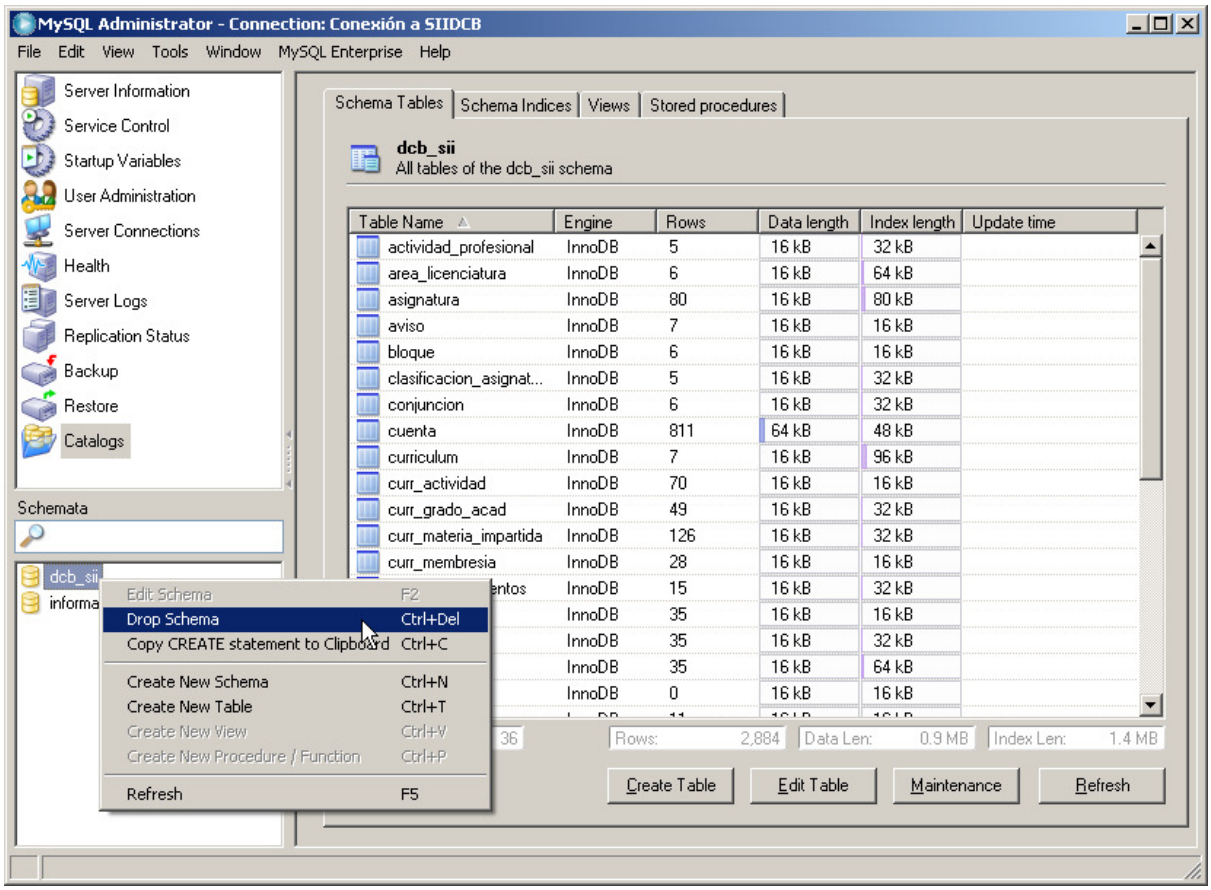


Restauración de la base de datos a partir de un archivo de respaldo.

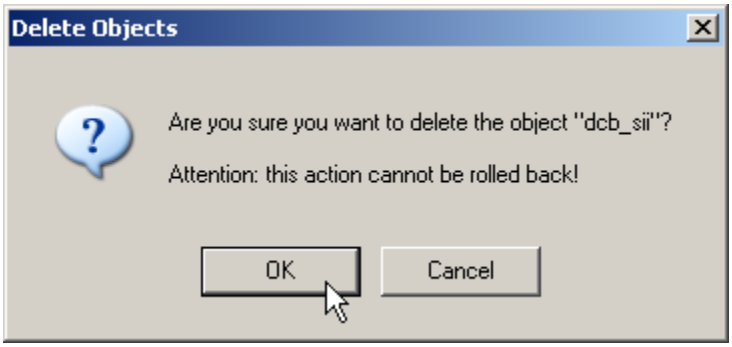
En la sección anterior se mostró como generar una archivo de respaldo (que es un script de SQL). Ahora veremos cómo utilizar dicho archivo para restaurar la base de datos al estado en que se encontraba cuando se hizo el respaldo.

El primer paso en abrir MySQL Administrator y conectarse a la base de datos como se ha visto anteriormente. A continuación es necesario seleccionar la opción “Catalogs” en el menú del lado izquierdo.

Es necesario borrar por completo la base de datos antes de restaurarla para asegurar que la base de datos quede exactamente como al momento de haber hecho el respaldo. Para esto debemos dar clic secundario sobre el nombre de la base de datos de SIIDCB en la lista indicada con “Schemata” que se encuentra en la parte inferior izquierda de la ventana, y seleccionar la opción “Drop Schema”.

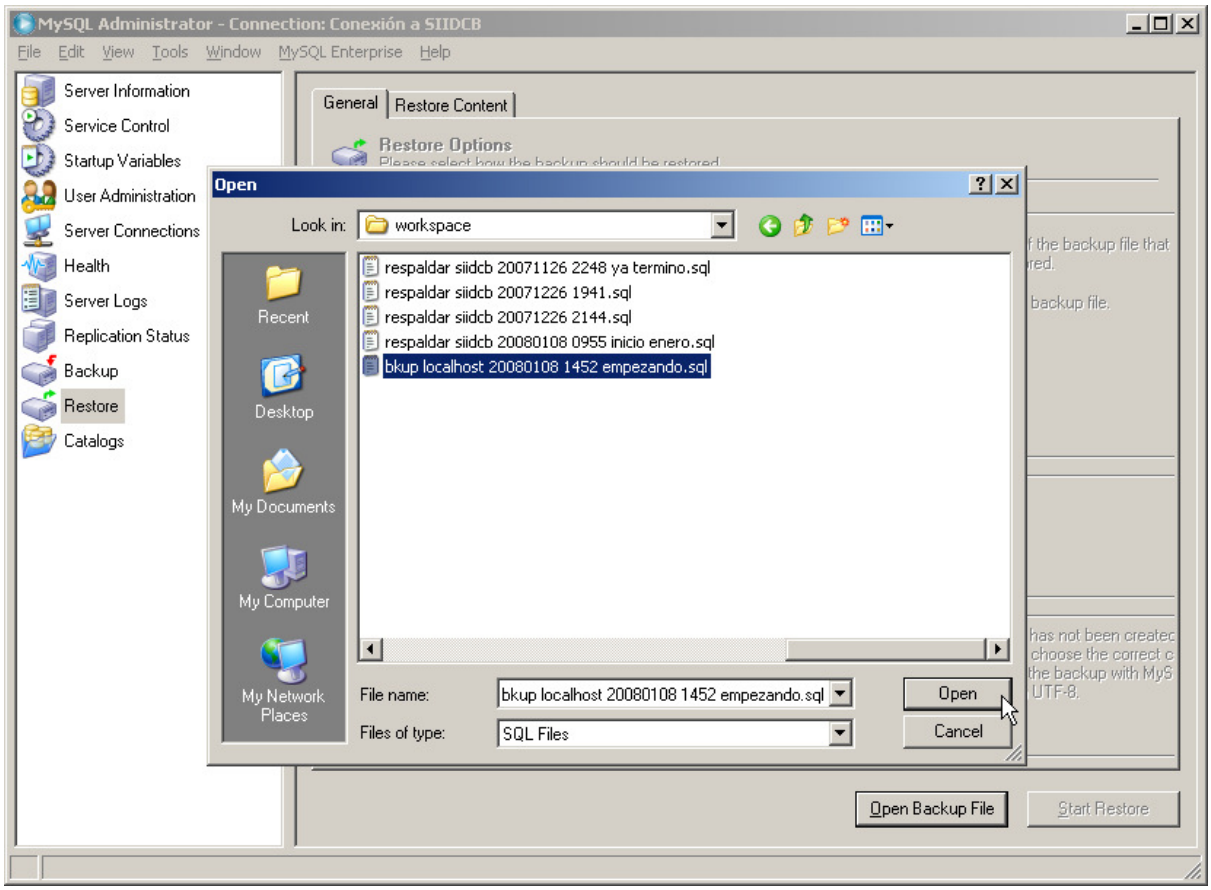


El programa pedirá un confirmación de eliminación de la base de datos donde debemos presionar el botón “OK”.

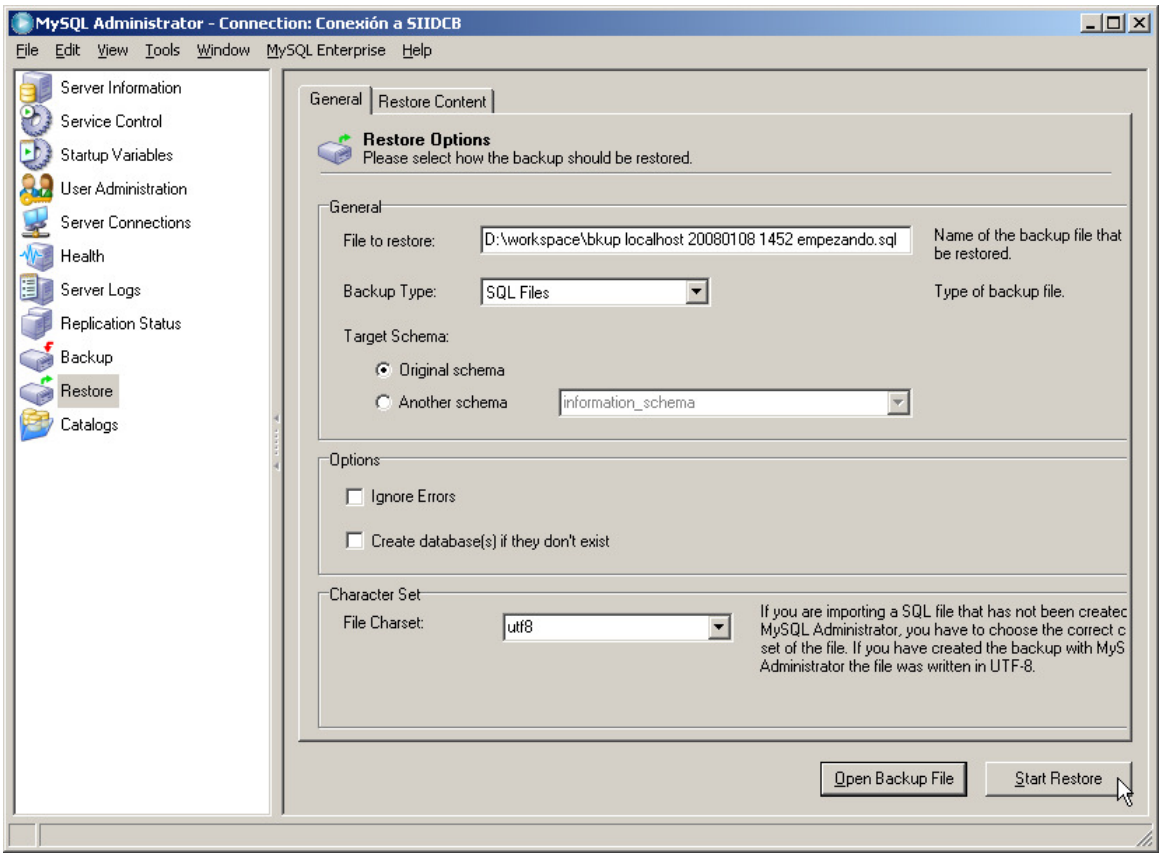


En este punto el servidor de base de datos no contiene ninguna que se llama dcb_sii ya que la hemos eliminado totalmente, sin embargo la cuenta con la que nos hemos conectado aún tiene todos los permisos sobre cualquier base que tenga el nombre dcb_sii, de modo que podemos volver a crearla, lo cual haremos pero con el archivo de respaldo creado previamente.

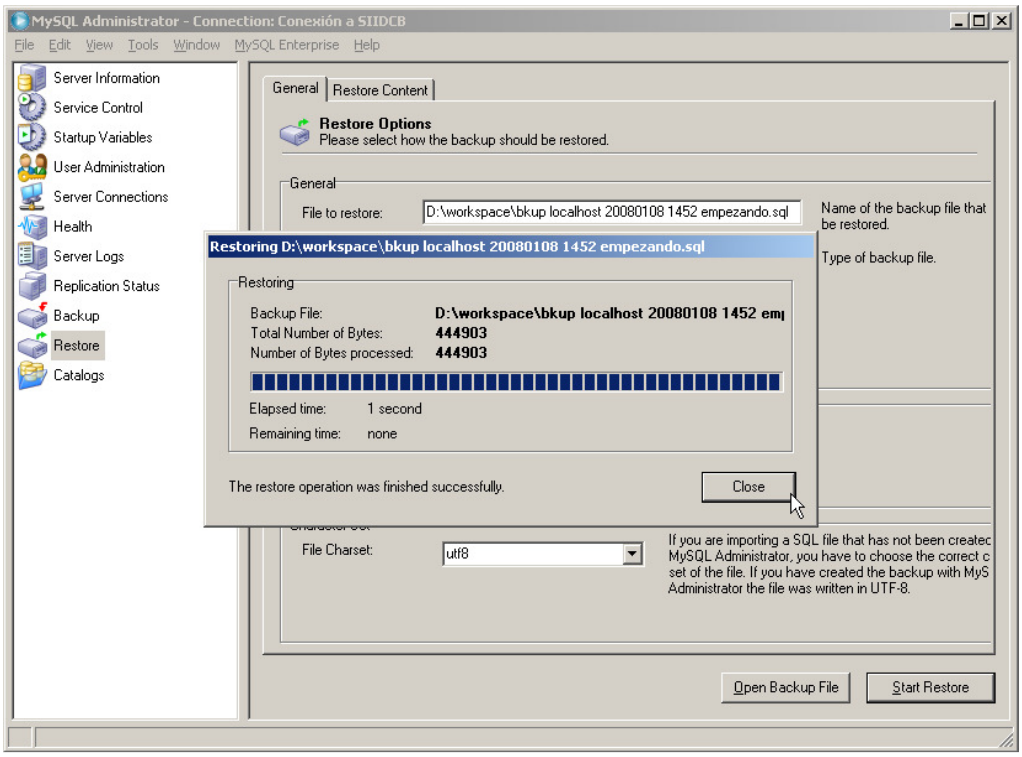
Para restaurar la base de datos ahora hay que seleccionar la opción “Restore” del menú del lado izquierdo del programa y abrir el archivo de respaldo. Para esto debe darse clic al botón “Open Backup File”, ubicar el archivo dentro del explorador de archivos que aparece y presionar el botón “Abrir”.



Una vez cargado el archivo de respaldo sólo debemos asegurarnos que esté seleccionado la opción “Original Schema” en la parte de “Target Schema” y presionar el botón “Start Restore” a continuación.



El finalizar la restauración correctamente, MySQL Administrator muestra el aviso “The restore operation was finished succesfully.”



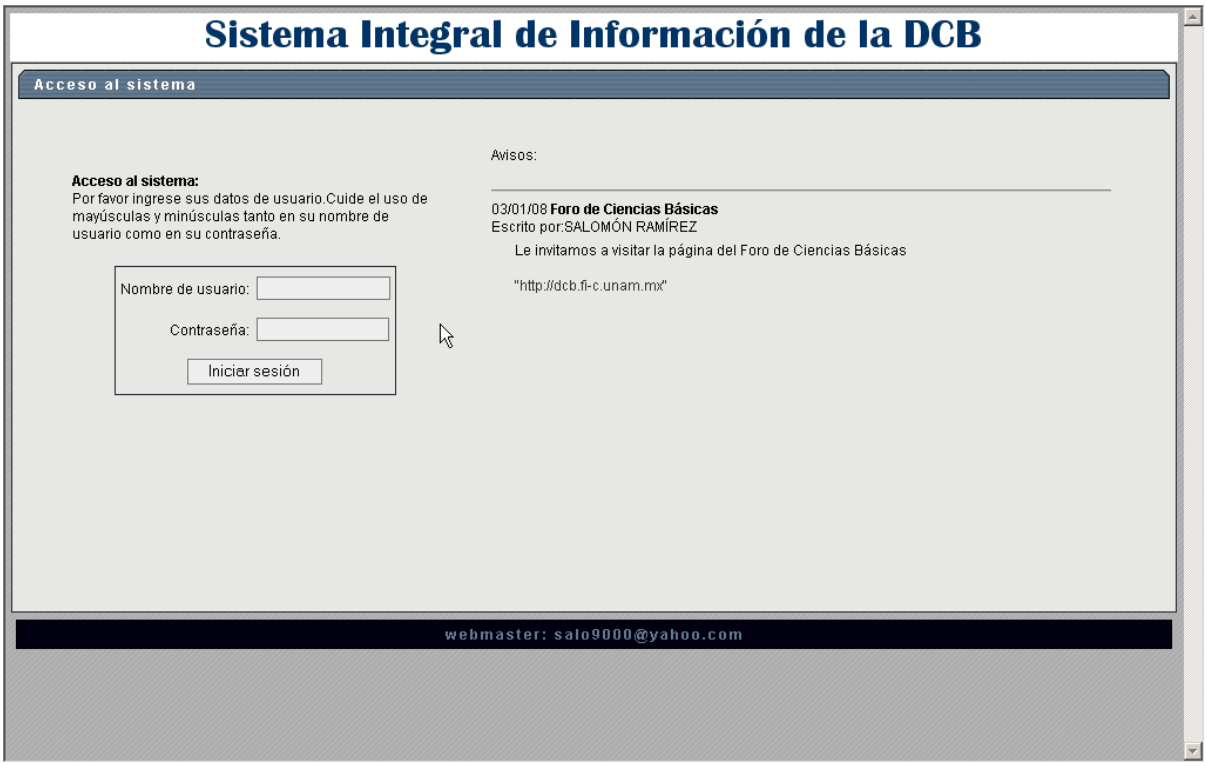
Acceso a la aplicación.

Ya que SIIDCB es una aplicación web, el acceso debe realizarse por medio de un navegador web como Microsoft Internet Explorer, Netscape, Firefox, etc. Para ingresar debe solicitarse la URL donde está localizada la aplicación. Originalmente la aplicación se ha ubicado en “http://www.basicas.fi-c.unam.mx:9090/siidcb”.

Para acceder debe contarse con una cuenta en el sistema, mismas que son creadas sólo por los administradores del sistema. Los datos requeridos en la página de entrada del sistema son el nombre de usuario de la cuenta y la contraseña.
Las cuentas de acceso a SIIDCB son de tres tipos:

- Administrador.
- Funcionario.
- Profesor.

Cada tipo tiene acceso sólo a ciertos contenidos y funcionalidades dentro de la aplicación. Por ejemplo, las cuentas tipo “Profesor” no pueden consultar ni hacer cambios a la información relacionada con el personal de la DCB, sólo pueden ingresar para actualizar su dirección de correo electrónico y para ingresar su currículum.



Una vez que se ha ingresado al sistema proporcionando nombre de usuario y contraseña válidos, SIIDCB muestra la página de inicio donde se observa el menú principal del lado izquierdo y del lado derecho los avisos para el tipo de cuenta con el que se ha ingresado.

Glosario.

AJAX: (Asynchronous JavaScript and XML – JavaScript Asíncrono y XML) técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantienen comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre una misma página web sin necesidad de recargarla manualmente.

API: (Application Programming Interface - Interfaz de Programación de Aplicaciones) conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

Aplicación web o sistema web: aplicación que puede ser accedida a través de internet o una intranet. También es llamada *webapp,*

Autenticación: proceso por medio del cual en una aplicación se verifica que un usuario es quien realmente dice que es. Por lo general esto se hace cuando el usuario proporciona su nombre de usuario y su contraseña y estos se comparan con datos a los que sólo la aplicación puede acceder.

Clase: declaración o abstracción de objetos; definición general de un conjunto de objetos con características en común.

Cliente: aplicación o sistema que accede a un servicio remoto en el sistema de otra computadora llamada servidor por medio de una red.

GUI: (Graphical User Interfase – Intefaz Gráfica de Usuario) es un tipo de interfaz que permite a un usuario interactuar con una computadora o con dispositivos controlados por una computadora. En lugar desplegar menús por medio de texto o de requerir la escritura de comandos por parte del usuario, una GUI muestra íconos, señaladores visuales u otros elementos gráficos para representar la información y las acciones que puede llevar a cabo el usuario. Las acciones se realizan por medio de la manipulación de estos elementos gráficos (clics, *drag-and-drop*, etc.)

HTTP: (Hypertext Transfer Protocol – Protocolo de Trasnferencia de Hipertextos) es un protocolo de comunicación empleado para solicitar y enviar páginas de internet.

JavaBean: JavaBeans, o simplemente beans, son clases escritas en el lenguaje Java de acuerdo a una convención en particular establecida por Sun Microsystems. Son empleados para encapsular muchos objetos en uno solo (el bean) de modo que éste sea fácilmente manipulado en lugar de enviar y recibir objetos individuales. Poseen métodos públicos de lectura y escritura para cada uno de sus atributos que son privados.

JDBC: (Java Database Connectivity – Conectividad Java a Bases de Datos) API para trabajar con bases de datos desde la plataforma Java. Esta API es independientemente de la base de datos a la que se accede. Utiliza un driver, que es una clase ofrecida por el vendedor de la BD, que implementa la interfaz Driver adecuada para acceder a la base de datos.

Lógica de negocio: término para designar los algoritmos del funcionamiento de una aplicación que manipulan el intercambio información entre una base de datos y el usuario.

Objeto: representación detallada, concreta y particular de una clase. Tal representación determina su identidad, su estado y su comportamiento particular en un momento dado.

ODBC: (Open Database Connectivity – Conectividad Abierta a Bases de Datos) estándar desarrollado por Microsoft Corporation que permite el acceso a cualquier tipo de base de datos desde otro tipo de aplicaciones.

ORM: (Object-Relational Mapping - Mapeo Objeto-Relacional) es una técnica de programación para convertir datos entre sistemas de tipos incompatibles. Se utiliza en programas escritos con lenguajes orientados a objetos que utilizan bases de datos.

Persistencia de objetos: en programación orientada a objetos, es la capacidad que tienen los objetos de conservar su estado e identidad entre distintas ejecuciones del programa que los creó o de otros programas que accedan a ellos. La persistencia permite al programador almacenar, transferir y recuperar el estado de los objetos ya sea por serialización, motores o servicios de persistencia, o bases de datos orientadas a objetos.

Servlet: (Java Servlet API) API que permite a los desarrolladores añadir contenido dinámico a un servidor web empleando la plataforma Java. Generalmente el contenido generado se compone de páginas HTML, pero pueden emplearse otras tecnologías tales como XML.

UML: (Unified Modeling Language - Lenguaje Unificado de Modelado) lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

URL: (Uniform Resource Locator – Localizador Uniforme de Recurso) secuencia de caracteres, de acuerdo a un formato estándar, que se usa para especificar recursos, como documentos e imágenes en Internet, por su localización. Por ejemplo

http://basicas.fi-c.unam.mx:9090/siidcb/img/siidcb.gif es la URL para una imagen de tipo GIF ubicada en el servidor de HTTP basicas.fi-c.unam.mx.

Usuario: en el contexto computacional el término usuario hace referencia a la persona que hace uso de una aplicación o sistema de cómputo. Por lo general los usuarios deben identificarse por cuestiones de seguridad, bitácoras, etcétera con un nombre de usuario (*username*) y una contraseña (*password*). Al proceso de identificación se le conoce también como autenticación.

Bibliografía.

Bauer, Christian y King, Gavin.
"Java Persistence with Hibernate"
Manning Publications, Nueva York, Nueva York, E.U.A., 2006.

Barclay, Kenneth.
"Object-Oriented Design with UML and Java"
Butterworth-Heinemann, 2003.

Doray, Arnold.
"Beginning Apache Struts: From Novice to Professional"
Apress, Nueva York, Nueva York, E.U.A., 2006.

Forta, Ben et al.
"Java Server Pages Application Development"
Sams, E.U.A., 2000.

Melton, Jim.
"SQL y Java: guia para SQLJ, JDBC y tecnologias relacionadas"
Alfaomega, Mexico, D. F., 2002.

Shenoy, Srikanth y Mallya, Nithin.
"Struts Survival Guide: Basics to Best Practices"
Serie: J2ee Survival.
ObjectSource Publications, Austin, Texas, E.U.A, 2004.

Referencias.

Software.

Jackcess
Biblioteca Java para MS Access.
http://jackcess.sourceforge.net/

c3p0
Biblioteca para crear conexiones a una base de datos con JDBC.
http://sourceforge.net/projects/c3p0

Cewolf
Biblioteca basada en JFreeChart para crear gráficos.
http://cewolf.sourceforge.net/

Hibernate
Software para persistencia relacional para Java y .NET.
http://www.hibernate.org/

Struts Framework
Marco de trabajo *open-source* para webapps basado en el patrón MVC.
http://struts.apache.org/

MySQL
Manejador de base de datos.
http://www.MySQL.com/

Java EE
Plataforma Java Enterprise Edition (Java EE).
http://java.sun.com/javaee/

Eclipse
Entorno de desarrollo para Java.
http://www.eclipse.org/

Omondo
Software para UML.
http://www.eclipsedownload.com/

MicroNova YUZU
Taglib JSP *open-source* que soporta recursión y muchas otras características avanzadas.
http://www.micronova.com/yuzu.jsp

MyEclipse
Extensión para eclipse.
http://www.myeclipseide.com/

W3C
Consorcio intenacional para la creación de estándares en la WWW.
http://w3c.org/

WAPT (Web Applications Testing)
Herramienta para pruebas de aplicaciones Web.
http://www.loadtestingtool.com/index.shtml

Documentos en línea.

Convenciones de Código Java.
“Code Conventions for the JavaTM Programming Language”
http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

Plugin de Hibernate para Struts del sitio oficial de Hibernate.
“HibernatePlugIn for Struts”
http://www.hibernate.org/105.html

Configuración de C3PO.
“HowTo configure the C3P0 connection pool”
http://www.hibernate.org/214.html

Sobre el caché de Hibernate.
“Cachés, concurrencia e Hibernate”
http://www.javahispano.org/contenidos/es/caches__concurrencia_e_hibernate/

Tutorial de Hibernate 3 de Roseindia.
“Complete Hibernate 3.0 Tutorial”
http://www.roseindia.net/hibernate/index.shtml

Plugin de Hibernate para Struts de Roseindia.
“Developing Struts Hibernate Plugin”
http://www.roseindia.net/struts/struts-hibernate/struts-hibernate-plugin.shtml