UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE ADMINISTRACIÓN Y CONTROL PARA EL PROGRAMA DE TECNOLOGÍA EN CÓMPUTO DE LA DIVISIÓN DE INGENIERÍA ELÉCTRICA (PORTAL PROTECO)

T E S I S
QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN
P R E S E N T A N
ALFREDO BENAVIDES MARTÍNEZ
JULIO CÉSAR SAYNEZ FABIÁN

DIRECTOR DE TESIS:
ING. ALEJANDRO VELÁZQUEZ MENA

CIUDAD UNIVERSITARIA 2007

Agradecimientos

A mis padres Juan Carlos y Leticia, por su amor y apoyo incondicional.

A Shumí y Tito, por su amor y dedicación de toda la vida.

A mis hermanos Nancy, Erika, Daniel, Lalo y Simba, por su amistad y comprensión.

A Eva y David, por todo el apoyo y cariño.

A mis abuelos Cristina y Carlos por todo el amor que siempre me han dado.

A mis tíos Valentín y Cristina por la confianza y cariño que siempre me han demostrado.

A mis tíos Martha, Ciro, Marcela, Gerardo por los buenos ejemplos que me han dado.

A mis primos Andrea, Leonardo y Jade para que continúen mejorando siempre.

A mis tíos y primos.

A mis amigos de toda la vida José Luis, Arturo y Víctor por siempre estar ahí.

A mis amigos Lulú, Karina, Jannet, Diana, Nayeli, Luis Miguel, JR, Serge, Cinthya, Carlos, Vlad, Mena, Dan, Saynez, Niño, Stimpy, César, Jorge, Master, por el apoyo incondicional y sobre todo su amistad.

Al PROTECO por toda su enseñanza.

A todos los PROTECOS y EXPROTECOS, porque continuemos con esta tradición de compañerismo y aprendizaje.

A mis Profesores, por toda su sabiduría que compartieron.

A mi Facultad, por darme la oportunidad de formar parte de ella.

A mi Universidad, por enseñarme que ser universitario es una forma de vida y no solo una etapa en la vida.

A todos muchas !GRACIAS¡

Alfredo Benavides Martínez.

En este trabajo existen varias personas involucradas, quisiera comenzar, con mi increíble familia que hizo posible que aprendiera bastante acerca del mundo de la tecnología. Les agradezco a mis padres Rocío y Francisco Javier, así como a mi hermano Miguel Ángel.

A mis tíos y primos por alentarme siempre para seguir adelante.

A todos mis amigos por compartir tantos momentos y enseñarme que el éxito no tiene sentido si no tienes a nadie con quien compartirlo, gracias a Cinthia Lévaro Arroyo, Sergio David Jardón Avalos, Alejandra Chavira Flores, Dan Schmidt Valle, Edgar Eduardo García Cano Castillo, Alfredo Benavides Martínez, José Alberto Conrrado Flores, Jazmin Sánchez Reyes, Martha Angélica Nakayama Cervantes, César Arian Ortega Arias, Carolina Martínez, Tanya Itzel Arteaga Ricci, José Ricardo Manríquez Betanzos, Hector Armando Pérez Jasso, Karla Romero González.

Al Ing. Alendro Velásquez Mena por confiar en mí y poner a prueba mis habilidades.

A todos aquellos que participan en PROTECO.

A mis sinodales Ing. Heriberto Olguín Romo, M. en I. Jorge Valeriano Assem, Ing. Carlos Alberto Román Zamitiz, M.I. Aurelio Adolfo Millán Nájera.

Por último quiero agradecer a los estupendos profesores que he tenido. Es su dedicación la que hizo que tuviera éxito en mi empeño.

Julio César Saynez Fabián.

TEMARIO

FIGURAS

1. Conceptos Básicos

1.1. Historia del PROTECO

El Programa de Tecnología en Cómputo (PROTECO) surge con la idea de involucrar a futuros ingenieros en las áreas de investigación y docencia utilizando recursos en cómputo, estos programas fueron pensados para implementarse en todas las divisiones de estudios de la Facultad de Ingeniería. Con este propósito surge el primer programa auspiciado por la División de Ingeniería Eléctrica, que tiene a su cargo las carreras de Ingeniero en Computación, Ingeniero Eléctrico/Electrónico e Ingeniero en Telecomunicaciones. El fundador del programa es el Ing. Heriberto Olguín Romo.

El Programa de Tecnología en Cómputo desde su creación ha contado con gran respaldo y aceptación, convirtiéndose este en el programa de becarios líder de la facultad.

Para el año de 1999 en la Universidad y por lo tanto en la Facultad de Ingeniería(FI) se vivió el conflicto académico por todos conocido, que posteriormente derivó en la huelga que por más de un año no permitió el desarrollo pleno de actividades en la FI y por lo tanto del PROTECO. Para el año 2000 con el reinicio de actividades de nuestra Universidad, la FI retomo sus actividades paulatinamente, en este contexto el programa de tecnología en cómputo intenta resurgir y en ese afán de renovación que invadía cada rincón de nuestra Universidad, el programa cambia de nombre a PROTECO[1].

1.1.1. Objetivo del PROTECO

El programa tiene como objetivo fundamental, coadyuvar en la generación de futuros investigadores y académicos, así como fortalecer la formación de ingenieros.

1.1.2. Objetivos generales del PROTECO

Establecer un proceso permanente para la capacitación de personal docente y/o de investigación en diferentes áreas, tales como: Telecomunicaciones, Ingeniería de Software, Ingeniería Asistida por Computadora, Ingeniería Mecánica, Ingeniería Industrial e Ingeniería Civil, mediante la participación de los integrantes en cursos especializados y en el desarrollo de proyectos.

Apoyar la formación de personal técnico, docente y de investigación en las áreas ya mencionadas; así como, ampliar la formación profesional de los alumnos participantes.

[1] PTC fue el primer nombre que se le dio al programa.

1.1.3. Objetivos específicos del PROTECO

Preparar personal de apoyo en las áreas indicadas para:

- Los requerimientos de docencia y de investigación de la Facultad de Ingeniería.
- La elaboración de material didáctico e impartición de cursos.
- Los requerimientos de innovación tecnológica, tanto de la facultad como de los sectores productivo y gubernamental.
- Crear, mantener y ampliar centros y laboratorios de: telecomunicaciones, diseño de aplicaciones para computadora y de diseño y manufactura.
- Propiciar la participación de los sectores productivo y gubernamental e instituciones externas a la UNAM en el programa.

1.1.4. Ventajas

Mediante el Programa de Tecnología en Cómputo se obtiene las siguientes ventajas:

- Se captará a un número importante de posibles profesionales, docentes y/o investigadores en áreas emergentes para el País, la UNAM y la FI.
- Se logra el máximo potencial de cada uno de los alumnos y profesores que formen parte del programa, mediante el compromiso total, de todos sus integrantes.
- Los egresados del Programa cuentan con visiones, aptitudes, habilidades y herramientas distintas a las antes acostumbradas.
- Se establece un vínculo de amplia cooperación entre los sectores productivos, de gobierno y la Facultad de Ingeniería.
- Se crean laboratorios con equipos de tecnología de vanguardia. A los cuales podrán acudir los sectores productivos y gubernamentales.
- Se establecen convenios para el desarrollo de proyectos tecnológicos y de investigación con los sectores productivos y gubernamentales del país.
- Con los alumnos que participan en el programa, se incrementa su participación en las diferentes asignaturas de la carrera que cursan, por lo que se eleva el nivel académico de la facultad.
- Se amplía la formación profesional de los alumnos que participan en el programa.

- Se establece una liga fundamental entre la División Profesional y la de Estudios de Posgrado, y se orienta a un mayor número de alumnos a realizar estudios de Posgrado.

- Se participa con los investigadores de algunos institutos y centros de la UNAM y externos a ella en el desarrollo de proyectos.

- El personal docente y de investigación que egresa del programa puede actualizar sus conocimientos, asistiendo a los nuevos cursos que surjan en el desarrollo del programa.

1.1.5. Estructura

Para ser parte del programa se requiere presentar un examen de conocimientos generales en matemáticas, programación e inglés. Habiendo aprobado el examen y cubierto los requerimientos de entrada al programa, se inicia la fase de preparación para convertirse en becario del programa, la cual es conocida como fase de Prebecario. Esta etapa es la fase más importante del programa ya que en esta se sientan las bases de la preparación del becario, la etapa consiste en una serie de cursos intensivos de diversos lenguajes de programación, sistemas operativos, redes y bases de datos.

Cada semestre ingresará una generación de alumnos a los que se aplicará un examen y se seleccionarán un máximo de treinta, a los cuales se les impartirán cursos de veinte horas a la semana durante el semestre.

En base a las calificaciones obtenidas en los cursos, promedio general en la carrera y una entrevista personal, se seleccionan los alumnos, los cuales pasarán a la fase de desarrollo y formarán una generación de becarios del programa.

Al finalizar esta etapa el prebecario se integra de lleno a la vida del programa convirtiéndose en becario. El programa se caracteriza por formar a su propio personal mediante la impartición de cursos de selección a los futuros miembros del programa.

1.1.6. Organización

Internamente el programa se divide en secciones: Sistemas Operativos/Redes, Lenguajes y Bases de Datos. Existen coordinadores en cada área que orientan a los becarios a su cargo en los diversos proyectos que se encuentren realizando.

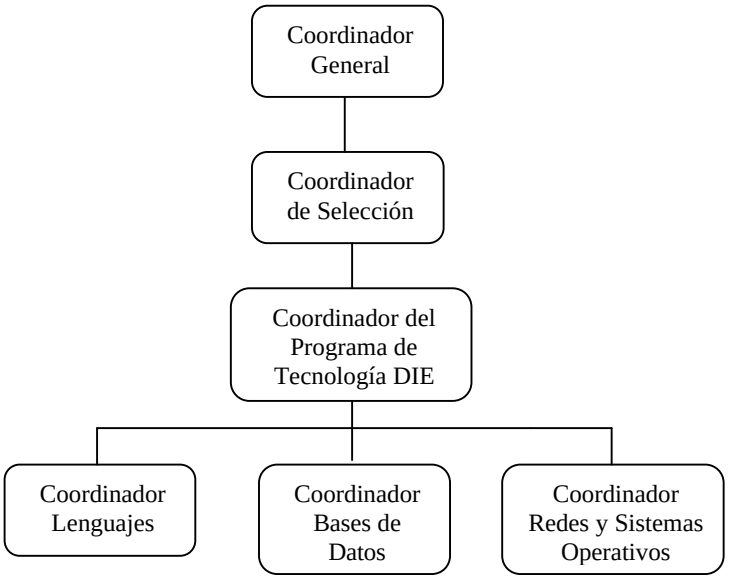


Figura 1-1 Organigrama PROTECO.

1.1.7. Actividades

El programa internamente trabaja en diversos proyectos que son asignados por diversas entidades de la facultad, de la universidad o de instituciones privadas o gubernamentales. Internamente los proyectos de mayor importancia para el programa son los cursos que se imparten. Estos cursos se dividen en dos proyectos, los cursos para prebecarios, cursos intersemestrales.

Los cursos para prebecarios son los cursos de formación de becarios, que se imparten de forma intensiva a las nuevas generaciones de becarios PROTECO.

Los cursos intersemestrales es el proyecto que con mayor constancia permite adquirir recursos al programa, estos cursos son impartidos al público general interesado y gozan de un gran prestigio dentro y fuera de la Universidad.

1.2. Objetivo del portal PROTECO

El principal objetivo del portal, es el de llevar un control del proceso administrativo que se utiliza para la impartición de cursos del PROTECO. Este proceso involucra la creación de los cursos, inscripciones e impresión de resultados.

1.2.1. Objetivos particulares

- Crear una herramienta de apoyo para la impartición de los cursos PROTECO, tanto para los instructores como para los alumnos.

- Crear un medio de intercambio de información simple y objetivo entre los instructores y los alumnos de los cursos PROTECO.

- Crear una herramienta que pueda ser utilizada por alumnos e instructores desde la comodidad de su casa o centro de trabajo.

1.3. Ingeniería de Software

Según la definición del IEEE, "software es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo". Según el mismo autor, "un producto de software es un producto diseñado para un usuario". En este contexto, la Ingeniería de Software (SE del inglés *Software Engineering*) es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software", que en palabras más llanas, se considera que "la Ingeniería de Software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software", es decir, "permite elaborar consistentemente productos correctos, utilizables y costo-efectivos" [2].

El proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad"[3].El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo.

El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición. La concepción define el alcance del proyecto y desarrolla un caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

La Ingeniería de Software es relativamente nueva ya que aparece a finales de los años sesenta y principios de los setenta, comenzando con las Técnicas de Programación Estructurada, incorporándolas a las fases del ciclo vital de software. La Programación Estructurada fue seguida por otros métodos estructurados de análisis y también métodos

[2] Cota A. 1994 "Ingeniería de Software". Soluciones Avanzadas. Julio de 1994. pp. 5-13.
[3] Jacobson, I. 1998. "Applying UML in The Unified Process" Presentación. Rational Software.

estructurados de diseño. Además, comenzaron a usarse tecnologías orientadas a objetos. En un principio la programación era la tarea de oro de la Ingeniería de Software pero ahora la ingeniería y el diseño de requisitos son más importantes.

En los años noventa la gerencia de proyecto ganó interés y llegó a ser un componente importante en la Ingeniería de Software. En la década pasada, los estándares de la Ingeniería de Software y la madurez de proceso han caracterizado la industria del software como una disciplina madura.

En un nivel más técnico, la Ingeniería de Software comienza con una serie de tareas que hacen modelos y que resultan en una especificación completa de requisitos y una representación comprensiva de diseño del software que será construido.

1.3.1. Ciclo de desarrollo

El software es un elemento lógico, por lo que tiene unas características muy diferentes a las del hardware:
El software se desarrolla, no se fabrica en el sentido clásico de la palabra. Ambas actividades se dirigen a la construcción de un "producto", pero los métodos son diferentes. Los costos del software se encuentran en la Ingeniería, esto implica que los proyectos no se pueden gestionar como si lo fueran de fabricación. A mediados de la década de 1980, se introdujo el concepto de "fábrica de software", que recomienda el uso de herramientas para el desarrollo automático del software.

Si se representa gráficamente la proporción de fallos en función del tiempo, para el hardware se tiene la figura conocida como "curva de bañera". Al principio de su vida hay bastantes fallos (normalmente por defectos de diseño y/o fabricación), una vez corregidos se llega a un nivel estacionario (bastante bajo). Sin embargo conforme pasa el tiempo, aparecen de nuevo, por efecto de: mala calidad, suciedad, malos tratos, temperaturas extremas y otras causas. El hardware empieza a estropearse.

El software no se estropea. La gráfica de fallos en función del tiempo, tendría forma de caída desde el principio, hasta mantenerse estable por tiempo casi indefinido. El software no es susceptible a los males del entorno que provocan el deterioro del hardware. Los efectos no detectados harán que falle el programa durante las primeras etapas de su vida, sin embargo una vez corregidas, no se producen nuevos errores. Aunque no se estropea, si puede deteriorarse. Esto sucede debido a los cambios que se efectúan durante su vida.

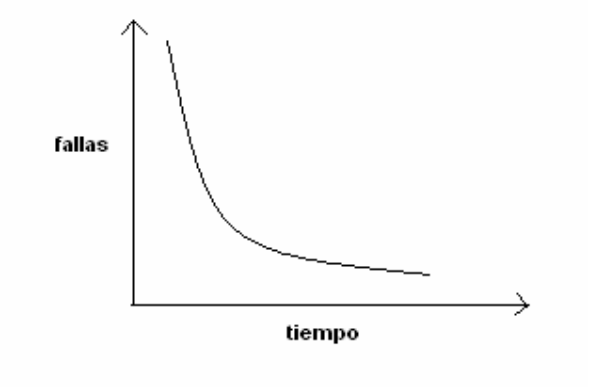


Figura 1-2 Fallas en función del tiempo.

Cuando un componente hardware se estropea, se cambia por otro que actúa como una "pieza de repuesto", mientras que para el software, no es habitual este proceso, lo cual significa que el mantenimiento de los programas es muy complejo.

La mayoría del software se construye a medida, en vez de ensamblar componentes previamente creados. Por contra en el hardware se dispone de todo tipo de circuitos integrados, para fabricar de manera rápida un equipo completo. Los ingenieros de software no disponen de esta comodidad, aunque ya se están dando los primeros pasos en esta dirección, que facilitaría tanto el desarrollo de aplicaciones informáticas.

La formalización del proceso de desarrollo se define como un marco de referencia denominado ciclo de desarrollo del software o ciclo de vida del desarrollo del software o ciclo de vida del desarrollo.

Se puede describir como, "el período de tiempo que comienza con la decisión de desarrollar un producto software y finaliza cuando se ha entregado éste". Este ciclo, por lo general incluye las fases:

- requisitos
- diseño
- implantación
- prueba
- instalación
- aceptación

El ciclo de desarrollo software se utiliza para estructurar las actividades que se llevan a cabo en el desarrollo de un producto software. A pesar de que no hay acuerdo acerca del uso y la forma del modelo, este sigue siendo útil para la comprensión y el control del proceso.

Seguidamente se exponen las distintas aproximaciones de desarrollo de software, en función del tipo de ciclo de vida:

a) Aproximación convencional

Se introdujo por Winston Royce en la década de 1970, como una técnica rígida para mejorar la calidad y reducir los costos del desarrollo de software. Tradicionalmente es conocido como "modelo en cascada", porque su filosofía es completar cada paso con un alto grado de exactitud, antes de iniciar el siguiente.

Esquemáticamente se puede representar de la siguiente forma:

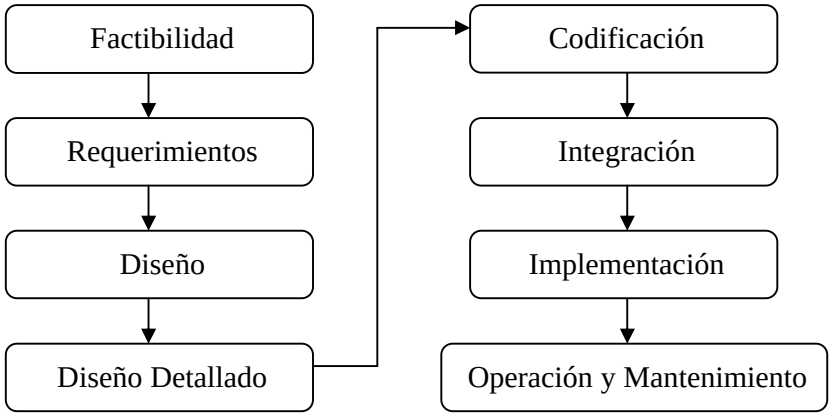


Figura 1-3 Modelo en cascada.

Donde:

FACTIBILIDAD: Definir un concepto preferente para el producto de software y determinar su factibilidad de ciclo de vida y superioridad frente a otros conceptos.

REQUERIMIENTOS: Elaborar una especificación completa y validada de las funciones requeridas, sus interfaces y el rendimiento del producto de software.

DISEÑO: Elaborar una especificación completa y validada de la arquitectura global hardware-software, de la estructura de control y de la estructura de datos del producto, así como un esquema de los manuales de usuarios y planes de test.

DISEÑO DETALLADO: Elaborar una especificación completa y verificada de la estructura de control, de la estructura de datos, de las interfaces de relación, dimensionamiento y algoritmos claves de cada componente de programa (rutina con un máximo de 100 instrucciones fuentes).

CODIFICACIÓN: Construir un conjunto completo y verificado de componentes de programas.

INTEGRACIÓN: Hacer funcionar el producto de software compuesto de componentes de programa.

IMPLEMENTACIÓN: Hacer funcionar el sistema global hardware-software incluyendo conversión de programas y datos, instalación y capacitación.

OPERACIÓN Y MANTENIMIENTO: Hacer funcionar una nueva versión del sistema global.

En cada caso, "verificación" tienen la acepción: establecer la verdad de la correspondencia entre un producto de software y su especificación. Es decir: ¿ESTAMOS CONSTRUYENDO CORRECTAMENTE EL PRODUCTO?

Los principales problemas que se han detectado en esta aproximación son debidos a que se comienza estableciendo todos los requisitos del sistema:

- En muchas ocasiones no es posible disponer de unas especificaciones correctas desde el primer momento, porque puede ser difícil para el usuario establecer al inicio todos los requisitos.
- En otras hay cambio de parecer de los usuarios sobre las necesidades reales cuando ya se ha comenzado el proyecto, siendo probables los verdaderos requisitos no se reflejen en el producto final
- Otro de los problemas de esta aproximación es que los resultados no se ven hasta muy avanzado el proyecto, por lo tanto la realización de modificaciones, si ha habido un error, es muy costosa.

Esta aproximación es la más empleada por los ingenieros informáticos, aunque ha sido muy criticada, y de hecho se ha puesto en duda su eficacia. Entre los problemas que se pueden encontrar con este modelo, se tienen:

- Los proyectos raras veces siguen el modelo secuencial que se supone. Los cambios pueden causar confusión.
- Es difícil disponer en principio de todos los requisitos. Este modelo presenta dificultades en el momento de acomodar estas incertidumbres.
- La versión operativa de los programas no está disponible hasta que el proyecto está muy avanzado. Un error importante puede ser desastroso, si se descubre al final del proceso.
- Los responsables del desarrollo siempre se retrasan innecesariamente. Algunos integrantes del equipo de desarrollo han de esperar a otros para completar tareas pendientes.

b) Aproximación prototipo

Es habitual que en un proyecto software no se identifiquen los requisitos detallados de entrada, procesamiento o salida. En otros casos no se está seguro de la eficiencia de un algoritmo, o de la forma en que se ha de implantar la interfase hombre-máquina.

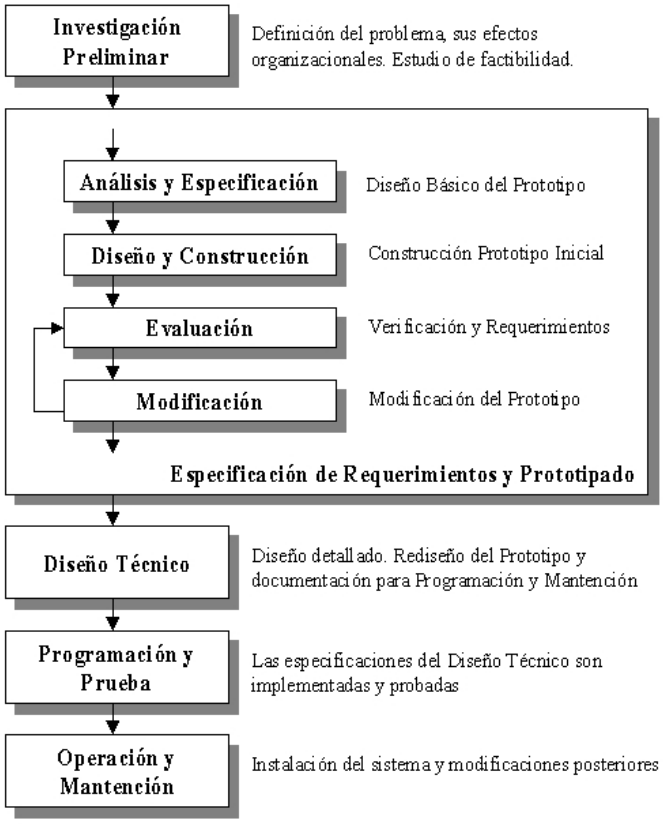


Figura 1-4 Desarrollo orientado a prototipos.

En casos así, lo habitual es construir un prototipo, que idealmente sirva como mecanismo para identificar los requisitos del software. Esta aproximación consiste en realizar la fase de definición de requisitos del sistema en base a estos tres factores:

- Un alto grado de iteración
- Un muy alto grado de interacción del usuario
- Un uso extensivo de prototipos

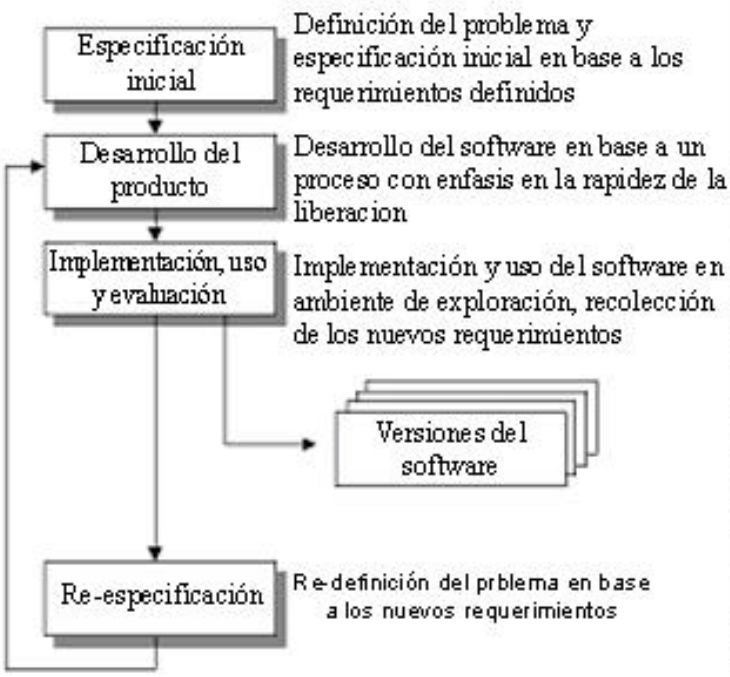
Las premisas clave de esta aproximación son:

- Que los prototipos constituyen un medio mejor de comunicación que los modelos en papel
- Que la iteración es necesaria para canalizar, en la dirección correcta, el proceso de aprendizaje. Esta aproximación se enfoca a mejorar la efectividad del proceso de desarrollo y no a mejorar la eficacia de ese proceso.
- El problema, es que los usuarios finales, ven lo que parece ser una versión de trabajo del software, sin considerar que no es la versión definitiva y por lo tanto no se han considerado aspectos de calidad o facilidad de mantenimiento. Cuando se les dice, que el producto es a partir de entonces cuando se debe de empezar a "fabricar", no lo entiende y empieza de nuevo con ajustes, lo cual hace este proceso muy lento.

c) Aproximación evolutiva

En esta aproximación el énfasis está en lograr un sistema flexible y que se pueda expandir de forma que se pueda realizar muy rápidamente una versión modificada del sistema cuando los requisitos cambien.

A diferencia de la aproximación anterior, los requisitos cambian continuamente, lo cual implicaría en el caso previo que las iteraciones no tendrían fin.



Desarrollo Evolutivo

Modelo de desarrollo que busca reemplazar a el viejo sistema con uno nuevo que tendría la propiedad de satisfacer los nuevos requerimientos lo más rápido posible. El desarrollo evolutivo asume que los requerimientos están sujetos a cambios continuos y que la estrategía para enfrentar aquello pasa por un reflujo, también continuo, de aquellos cambios.

Figura 1-5 Desarrollo evolutivo.

d) Aproximación incremental

Es un concepto muy parecido al desarrollo evolutivo, y frecuentemente comprendido en la aproximación del desarrollo evolutivo. Se comienza el desarrollo del sistema para satisfacer un subconjunto de requisitos especificados. Las últimas versiones prevén los requisitos que faltan. De esta forma se logra una rápida disponibilidad del sistema, que aunque incompleto, es utilizable y satisface algunas de las necesidades básicas de información.

La diferencia con la aproximación anterior es que en este caso cada versión parte de una previa sin cambios pero con nuevas funciones, mientras que la aproximación evolutiva cada vez se desarrolla una nueva versión de todo el sistema.

Un ejemplo de este paradigma se tiene en el desarrollo de una aplicación sencilla, como es un editor de textos. En el primer incremento se podría desarrollar con un reducido

conjunto de funciones, como las funciones básicas de gestión de archivos. En un segundo incremento, se puede incluir la gestión avanzada de textos. Y en un tercer incremento se pondría la corrección ortográfica.

e) Aproximación espiral

Nace con el objetivo de captar lo mejor de la aproximación convencional y de la de prototipo, añadiendo un nuevo componente, el análisis de riesgos. Esquemáticamente se puede ilustrar mediante una espiral, con cuatro cuadrantes que definen actividades.

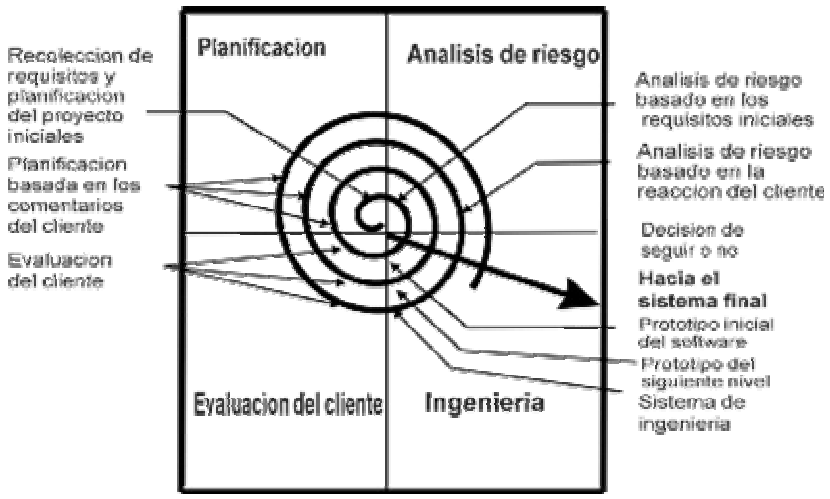


Figura 1-6 Desarrollo en espiral.

En la primera vuelta de la espiral se definen los objetivos, las alternativas y las restricciones y se analizan y se identifican los riesgos. Si como consecuencia del análisis de riesgo se observa que hay incertidumbre sobre el problema entonces en la actividad correspondiente a la Ingeniería se aplicará la aproximación prototipo cuyo beneficio principal es reducir la incertidumbre de la naturaleza del problema de información y los requerimientos que los usuarios establecen para la solución a ese problema.

Al final de esta primera vuelta alrededor de la espiral el usuario evalúa los productos obtenidos y puede sugerir modificaciones. Se comenzaría avanzando alrededor del camino de la espiral realizando las cuatro actividades indicadas a continuación. En cada vuelta de la espiral, la actividad de ingeniería se desarrolla mediante la aproximación convencional o ciclo de desarrollo en cascada o mediante la aproximación de prototipos.

- Recolección de requisitos y planificación del proyecto inicial.
- Análisis de riesgo basado en los requisitos iniciales.
- Prototipo inicial de software.
- Evaluación del cliente.
- Planificación basada en los comentarios del cliente.
- Prototipo del siguiente nivel.
- Segunda evaluación del cliente.
- Segunda planificación basada en los comentarios del cliente.
- Análisis de riesgo basado en la reacción del cliente.
- Sistema de Ingeniería.

- Tercera evaluación del cliente.
- Tercera planificación basada en los comentarios del cliente.

f) Aproximación basada en transformaciones

Con la aparición de las herramientas CASE junto con los generadores de código, el ciclo de desarrollo software en cascada ha cambiado a un ciclo de vida basado en transformaciones.

CASE (Computer Aided Software Engineering), en castellano "Ingeniería de software Asistida por Computadora", es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información.

La utilización de herramientas CASE afecta a todas las fases del ciclo de vida del software. Este ciclo de vida se puede considerar como una serie de transformaciones. Primero se definen los requisitos del sistema, seguidamente existe un proceso de transformación que hace que la especificación se convierta en un diseño lógico del sistema. Posteriormente, este sufre otro proceso de transformación para lograr un diseño físico, es decir que responda a la tecnología destino.

La tecnología CASE propone que estos procesos de transformación sean lo más automatizado posible. Sus ventajas son:
Posibilidad de comprobación de errores en etapas iniciales de desarrollo

- Posibilidad de realizar el mantenimiento en el ámbito de especificación.
- Soporte de seguimiento de los requisitos.
- Soporte de reusabilidad.
- Potencia la especificación orientada al problema.

1.4. Arquitectura de Software

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, simplemente es necesario conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actual se suele usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

1.4.1. Modelos o vistas

Toda arquitectura de software debe describir diversos aspectos, generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de los aspectos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierta integración entre ellos. Todas las vistas deben ser coherentes entre sí, dado que describen lo mismo.

Cada paradigma de desarrollo exige diferente número y tipo de vistas o modelos para describir una arquitectura. No obstante, existen al menos tres vistas absolutamente fundamentales en cualquier arquitectura:

- La visión estática: Describe qué componentes tiene la arquitectura.
- La visión funcional: Describe qué hace cada componente.
- La visión dinámica: Describe cómo se comportan los componentes a lo largo del tiempo y como interactúan entre sí.

Las vistas o modelos de una arquitectura pueden expresarse mediante uno o varios lenguajes. El más obvio es el lenguaje natural, pero existen otros lenguajes tales como los diagramas de estado, los diagramas de flujo de datos, etc. Estos lenguajes son apropiados únicamente para un modelo o vista. Afortunadamente existe cierto consenso en adoptar UML (Unified Modeling Language) como lenguaje único para todos los modelos o vistas. Sin embargo, un lenguaje general corre el peligro de no ser capaz de describir determinadas restricciones de un sistema de información (o expresarlas de manera incomprensible).

1.4.2. Arquitecturas más comunes

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

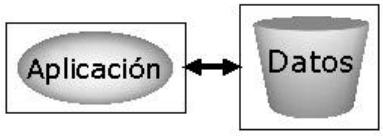
a) Monolítica

La arquitectura monolítica es considerada sin red, (finales 80's) comienza a madurar la plataforma PC tanto en su capacidad-hardware como su funcionalidad software, bajo un sistema operativo gráfico y multitarea surgen aplicaciones basadas en la elaboración de documentos, hojas de cálculo y pequeñas bases de datos (Figura 1-7), tanto el software como los datos residen en el disco duro de una computadora-cliente aislada.

Figura 1-7 Arquitectura monolítica PC.

b) Cliente/Servidor

También llamada "dos capas": las redes se ponen de moda, los servidores de red son simples depósitos de información, donde el cliente navega por sus directorios de archivos (netware). Surgen las primeras aplicaciones de red usando servidores de bases de datos (Figura 1-8). El despegue de Internet en 1993, con la primer oleada del fenómeno "Web", inaugura el servidor Web como fuente global de información navegando en hipertextos con clientes gráficos denominados navegadores Web.



Cliente Servidor

Figura 1-8 Arquitectura Cliente/Servidor.

c) Arquitectura de tres capas

La arquitectura de “tres capas” surge a mediados de los 90's. La arquitectura “dos capas” degenera en clientes pesados con enorme CPU, memoria y disco, aunado a un complejo software con la lógica y presentación de la aplicación junto con otros aspectos "administrativos" tales como enlace, multiprocesos, seguridad, transacciones, etc. Por otra parte, el denso tráfico de red que ocasiona y el acceso a recursos compartidos genera un cuello de botella. Otro problema es el mantenimiento de aplicaciones y sus versiones (en potencia por cada usuario). El siguiente paso evolutivo se basa en tres capas. Capa cliente a cargo de la presentación, capa intermedia ó middleware para la lógica y capa tres para datos, donde coexisten múltiples interfaces de usuario, tecnologías de componentes y manejadores de datos por cada servidor de aplicaciones.

Generalización de la arquitectura cliente/servidor donde la carga se divide en tres partes con un reparto claro de funciones: una capa para la presentación, otra para el cálculo y otra para el almacenamiento. Una capa solamente tiene relación con la siguiente.

Capa 1 o Cliente

Capa de presentación: es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

Capa 2 o Intermedia

La capa intermedia también llamada de negocio: es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

Capa 3 o de Datos

Es donde residen los datos. Está formada por un gestor bases de datos o más que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en una única computadora (no sería lo normal), si bien lo más usual es que haya una multitud de computadoras donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en la misma computadora, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o mas servidores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios servidores los cuales recibirán las peticiones de la computadora en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en una o más computadoras que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a

tener una serie de computadoras sobre los cuales corre la capa de datos, y otra serie de computadoras sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos Capas o Niveles no significan lo mismo ni son similares. El término capa hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación/ Lógica de Negocio/ Datos.

El termino nivel, corresponde a la forma como las capas lógicas, se encuentran distribuidos de forma física.

d) Multicapa

La arquitectura Multicapa o n-capas surge a finales de los 90's. Generaliza tres capas donde coexisten múltiples servidores de aplicaciones en un ambiente distribuido. Surge la segunda oleada del Web: sistemas basados en servicios. Se busca una arquitectura "ideal" más flexible, ligera, modular, segura, robusta, universal, multiusuario, multiplataforma, multilenguaje y escalable. En la intersección de Objetos, Internet y Sistemas Distribuidos surge la noción de Objetos Web. Se asumen el siguiente entorno de trabajo:

- Atender peticiones de clientes heterogéneos cuyo mínimo común denominador es: TCP/IP.

- No generar la interfaz de usuario para cada aplicación y sistema operativo, aceptando con todas sus virtudes y defectos al Web Browser (HTTP/HTML) como la "Interfaz de Usuario Universal".

- Ampliar los servidores Web para soportar aplicaciones Web para interactuar con clientes ligeros y tratar de garantizar un mínimo de eficiencia, funcionalidad y confiabilidad.

- Apoyarse en infraestructuras (frameworks) apropiados para servidores de aplicaciones cuyo costo sea razonable y esfuerzo de desarrollo sea equiparable al de otras arquitecturas, pero que ofrezca las bondades lo más cercana posibles a la arquitectura n-capas ideal.

1.5. Análisis, Diseño y Programación Orientada a Objetos

Se define a un objeto como "*una entidad tangible que muestra alguna conducta bien definida*". Un objeto "*es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos*".

1.5.1. Principios básicos

Actualmente, el *Análisis Orientado a Objetos* (*AOO*) va progresando como método de análisis de requisitos por derecho propio y como complemento de otros métodos de análisis. En lugar de examinar un problema mediante el modelo clásico de entrada-proceso-salida (flujo de información) o mediante un modelo derivado exclusivamente de estructuras jerárquicas de información, el AOO introduce varios conceptos nuevos, estos conceptos nuevos le parecen inusuales a mucha gente, pero son bastante naturales.

Una *clase* es una plantilla para objetos múltiples con características similares, las clases comprenden todas esas características de un conjunto particular de objetos, cuando se escribe un programa en lenguaje orientado a objetos, no se definen objetos verdaderos sino se definen clases de objetos.

Una *instancia* de una clase es otro término para un objeto real. Si la clase es la representación general de un objeto, una instancia es su representación concreta. A menudo se utiliza indistintamente la palabra objeto o instancia para referirse, precisamente a un objeto.

En los lenguajes orientados a objetos, cada clase está compuesta de dos cualidades: *atributos* (estado) y *métodos* (comportamiento o conducta). Los atributos son las características individuales que diferencian a un objeto de otro (ambos de la misma clase) y determinan la apariencia, estado u otras cualidades de ese objeto. Los atributos de un objeto incluyen información sobre su estado.

Los métodos de una clase determinan el comportamiento o conducta que requiere esa clase para que sus instancias puedan cambiar su estado interno o cuando dichas instancias son llamadas para realizar algo por otra clase o instancia. El comportamiento es la única manera en que las instancias pueden hacerse algo a sí mismas o tener que hacerles algo, los atributos se encuentran en la parte interna mientras que los métodos se encuentran en la parte externa del objeto

Para definir el comportamiento de un objeto, se crean métodos, los cuales tienen una apariencia y un comportamiento igual al de las funciones en otros lenguajes de programación, los lenguajes estructurados, pero se definen dentro de una clase. Los métodos no siempre afectan a un solo objeto; los objetos también se comunican entre sí mediante el uso de métodos. Una clase u objeto puede llamar métodos en otra clase u objeto para avisar sobre los cambios en el ambiente o para solicitarle a ese objeto que cambie su estado.

Cualquier cosa que un objeto no sabe, o no puede hacer, es excluida del objeto. Además, como se puede observar en los diagramas, las variables del objeto se localizan en el centro o núcleo del objeto. Los métodos rodean y esconden el núcleo del objeto de

otros objetos en el programa. Al empaquetamiento de las variables de un objeto con la protección de sus métodos se le llama *encapsulamiento*. Típicamente, el encapsulamiento es utilizado para esconder detalles de la puesta en práctica no importantes de otros objetos. Entonces, los detalles de la puesta en práctica pueden cambiar en cualquier tiempo sin afectar otras partes del programa.

Esta imagen conceptual de un objeto —un núcleo de variables empaquetadas en una membrana protectora de métodos— es una representación ideal de un objeto y es el ideal por el que los diseñadores de sistemas orientados a objetos luchan. A menudo, por razones de eficiencia o la puesta en práctica, un objeto puede querer exponer algunas de sus variables o esconder algunos de sus métodos.

El encapsulamiento de variables y métodos en un componente de software ordenado es una simple idea poderosa que provee dos principales beneficios a los desarrolladores de software:

- *Modularidad*: el código fuente de un objeto puede ser escrito, así como darle mantenimiento, independientemente del código fuente de otros objetos. Así mismo, un objeto puede ser transferido alrededor del sistema sin alterar su estado y conducta.
- *Ocultamiento de la información:* un objeto tiene una "interfaz publica" que otros objetos pueden utilizar para comunicarse con él. Pero el objeto puede mantener información y métodos privados que pueden ser cambiados en cualquier tiempo sin afectar a los otros objetos que dependan de ello.

Los objetos proveen el beneficio de la Modularidad y ocultar la información. Las clases proveen el beneficio de la reutilización. Los programadores de software utilizan la misma clase, y por lo tanto el mismo código, una y otra vez para crear muchos objetos.

En las implantaciones orientadas a objetos se percibe un objeto como un paquete de datos y procedimientos que se pueden llevar a cabo con estos datos. Esto encapsula los datos y el procedimiento, en la práctica es diferente, los atributos se relacionan al objeto o instancia y los métodos a la clase. ¿Por qué se hace así? Los atributos son variables comunes en cada objeto de una clase y cada uno de ellos puede tener un valor asociado, para cada variable, diferente al que tienen para esa misma variable los demás objetos. Los métodos, por su parte, pertenecen a la clase y no se almacenan en cada objeto, puesto que sería un desperdicio almacenar el mismo procedimiento varias veces y ello va contra el principio de reutilización de código.

Otro concepto muy importante en la metodología orientada a objetos es el de *herencia*. La herencia es un mecanismo poderoso con el cual se puede definir una clase en términos de otra clase; lo que significa que cuando se escribe una clase, sólo se tiene que especificar la diferencia de esa clase con otra, con lo cual, la herencia dará acceso automático a la información contenida en otra clase.

Con la herencia, todas las clases están arregladas dentro de una jerarquía estricta. Cada clase tiene una superclase (la clase superior en la jerarquía) y puede tener una o más subclases (las clases que se encuentran debajo de esa clase en la jerarquía). Se dice que las clases inferiores en la jerarquía, las clases hijas, heredan de las clases más altas, las clases padres.

Las subclases heredan todos los métodos y variables de las superclases. Es decir, en alguna clase, si la superclase define un comportamiento que la clase hija necesita, no se tendrá que redefinir o copiar ese código de la clase padre.

De esta manera, se puede pensar en una jerarquía de clase como la definición de conceptos demasiado abstractos en lo alto de la jerarquía y esas ideas se convierten en algo más concreto conforme se desciende por la cadena de la superclase.

Sin embargo, las clases hijas no están limitadas al estado y conducta provistos por sus superclases; pueden agregar variables y métodos además de los que ya heredan de sus clases padres. Las clases hijas pueden, también, sobrescribir los métodos que heredan por implementaciones especializadas para esos métodos. De igual manera, no hay limitación a un sólo nivel de herencia por lo que se tiene un árbol de herencia en el que se puede heredar varios niveles hacia abajo y mientras más niveles descienda una clase, más especializada será su conducta.

1.5.2. Historia del análisis y diseño orientado a objetos

Durante los ochenta y principios de los noventa Grady Booch, James Rumbaugh, e Ivar Jacobson trabajaban por separado en desarrollo de notaciones para el análisis y diseño de sistemas orientados a objetos.

Booch había escrito "Object-Oriented Analysis and Design with Applications" un libro de referencia en el análisis y diseño orientado a objetos desarrollando su propia notación.

Por su parte **James Rumbaugh** había desarrollado su propia notación de diseño orientado a objetos llamada OMT (Object Modeling Technique) en su libro "Object-Oriented Modeling and Design".

Por otro lado **Jacobson** se había revelado como un visionario del análisis (padre de los casos de uso) y sobre todo del diseño orientado a objetos, sorprendiendo a todo el mundo en "Object-Oriented Software Engineering: A Use Case Driven Approach".

A mediados de los noventa empezaron a intercambiar documentos y trabajar en conjunto produciendo grandes avances en el modelado de sistemas orientados a objetos.

En 1994 Rational contrató a Rumbaugh en donde ya trabajaba Booch, un año después Jacobson se unía a ellos en Rational.

En 1997 salió a la luz la versión 1.0 de UML (Unified Modeling Language).

1.5.3. Modelado de objetos

En la especificación del UML podemos comprobar que una de las partes que lo componen es un meta modelo formal. Un meta modelo es un modelo que define el lenguaje para expresar otros modelos. Un modelo en OO es una *abstracción cerrada semánticamente de un sistema y un sistema es una colección de unidades conectadas*

que son organizadas para realizar un propósito específico. Un sistema puede ser descrito por uno o más modelos, posiblemente desde distintos puntos de vista.

Una parte del UML define, entonces, una abstracción con significado de un lenguaje para expresar otros modelos (es decir, otras abstracciones de un sistema, o conjunto de unidades conectadas que se organizan para conseguir un propósito). Lo que en principio puede parecer complicado no lo es tanto si pensamos que uno de los objetivos del UML es llegar a convertirse en una manera de definir modelos, no sólo establecer una forma de modelo, de esta forma simplemente estamos diciendo que UML, además, define un lenguaje con el que podemos abstraer cualquier tipo de modelo.

El UML es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos. El modelado no es más que la construcción de un modelo a partir de una especificación. Un modelo es una abstracción de algo, que se elabora para comprender ese algo antes de construirlo. El modelo omite detalles que no resultan esenciales para la comprensión del original y por lo tanto facilita dicha comprensión.

Los modelos se utilizan en muchas actividades de la vida humana: antes de construir una casa el arquitecto utiliza un plano, los músicos representan la música en forma de notas musicales, los artistas pintan sobre el lienzo con carboncillos antes de empezar a utilizar los óleos, etc. Todos abstraen una realidad compleja sobre bocetos, modelos al fin y al cabo. La OMT, por ejemplo, intenta abstraer la realidad utilizando tres clases de modelos OO: el modelo de objetos, que describe la estructura estática; el modelo dinámico, con el que describe las relaciones temporales entre objetos; y el modelo funcional que describe las relaciones funcionales entre valores. Mediante estas tres fases de construcción de un modelo, se consigue una abstracción de la realidad que tiene en sí misma información sobre las principales características de ésta.

Los modelos, al no ser una representación que incluya todos los detalles de los originales, permiten probar más fácilmente los sistemas que modelan y determinar los errores. Según se indica en la Metodología OMT (Rumbaugh), los modelos permiten una mejor comunicación con el cliente por distintas razones:

- Es posible enseñar al cliente una posible aproximación de lo que será el producto final.
- Proporcionan una primera aproximación al problema que permite visualizar cómo quedará el resultado.
- Reducen la complejidad del original en subconjuntos que son fácilmente tratables por separado.

Se consigue un modelo completo de la realidad cuando el modelo captura los aspectos importantes del problema y omite el resto. Los lenguajes de programación que estamos acostumbrados a utilizar no son adecuados para realizar modelos completos de sistemas reales porque necesitan una especificación total con detalles que no son importantes para el algoritmo que están implementando. En OMT se modela un sistema desde tres puntos de vista diferentes donde cada uno representa una parte del sistema y una unión lo describe de forma completa. En esta técnica de modelado se utilizó una aproximación al proceso de implementación de software habitual donde se utilizan estructuras de datos (modelo de objetos), las operaciones que se realizan con ellos

tienen una secuencia en el tiempo (modelo dinámico) y se realiza una transformación sobre sus valores (modelo funcional).

UML utiliza parte de este planteamiento obteniendo distintos puntos de vista de la realidad que modela mediante los distintos tipos de diagramas que posee. Con la creación del UML se persigue obtener un lenguaje que sea capaz de abstraer cualquier tipo de sistema, sea informático o no, mediante los diagramas, es decir, mediante representaciones gráficas que contienen toda la información relevante del sistema. Un **diagrama** es una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las relaciones entre los objetos y los vértices se corresponden con los elementos del modelo. Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibujan de forma que se resaltan los detalles necesarios para entender el sistema.

1.5.4. Historia de la programación orientada a objetos

La Orientación a Objetos (O.O.) surge en Noruega en 1967 con un lenguaje llamado Simula 67, desarrollado por Krinsten Nygaard y Ole-Johan Dahl, en el centro de cálculo noruego.

Simula 67 introdujo por primera vez los conceptos de clases, corutinas y subclases (conceptos muy similares a los lenguajes Orientados a Objetos de hoy en día).

El nacimiento de la Orientación a Objetos en Europa pasó inadvertido para gran parte de los programadores. Hoy tenemos la Orientación a Objetos como un niño de 33 años al que todos quieren bautizar.

Uno de los problemas de inicio de los años setentas era que pocos sistemas lograban terminarse, pocos se terminaban con los requisitos iniciales y no todos los que se terminaban cumpliendo con los requerimientos se usaban según lo planificado. El problema consistía en cómo adaptar el software a nuevos requerimientos imposibles de haber sido planificados inicialmente.

Este alto grado de planificación y previsión es contrario a la propia realidad. El hombre aprende y crea a través de la experimentación, no de la planeación. La Orientación a Objetos brinda estos métodos de experimentación, no exige la planificación de un proyecto por completo antes de escribir la primera línea de código.

En los 70’s científicos del centro de investigación en Palo Alto Xerox (Xerox park) inventaron el lenguaje Small Talk que dio respuesta al problema anterior (investigar no planificar).

Small talk fue el primer lenguaje Orientado a Objetos puro de los lenguajes Orientados a Objetos, es decir, únicamente utiliza clases y objetos (Java usa tipos de datos primitivos, o bien los Wrappers que son clases que encapsulan tipos de datos primitivos).

Hasta este momento uno de los defectos más graves de la programación es que las variables eran visibles desde cualquier parte del código y podían ser modificadas incluyendo la posibilidad de cambiar su contenido (no existen niveles de usuarios o de seguridad, o lo que se conoce como visibilidad).

Quien tuvo la idea fue D. Parnas cuando propuso la disciplina de ocultar la información. Su idea era encapsular cada una de las variables globales de la aplicación en un solo módulo junto con sus operaciones asociadas, sólo mediante las cuales se podía tener acceso a esas variables.

El resto de los módulos (objetos) podían acceder a las variables sólo de forma indirecta mediante las operaciones diseñadas para tal efecto.

En los años 80's Bjarne Stroustrup de AT&T Labs., amplió el lenguaje C para crear C++ que soporta la programación Orientada a Objetos.

En esta misma década se desarrollaron otros lenguajes Orientados a Objetos como Objective C, Common Lisp Object System (CLOS), object Pascal, Ada y otros.

Posteriores mejoras en herramientas y lanzamientos comerciales de C++ por distintos fabricantes, justificaron la mayor atención hacia la programación Orientada a Objetos en la comunidad de desarrollo de software, siendo el detonante final el desarrollo técnico del hardware y su disminución del costo, dando como resulado más computadoras al alcance de más personas, más programadores, más problemas y más algoritmos surgieron.

En el inicio de los 90's se consolida la Orientación a Objetos como una de las mejores maneras para resolver problemas. Aumenta la necesidad de generar prototipos más rápidamente (concepto RAD Rapid Aplication Developments). Sin esperar a que los requerimientos iniciales estén totalmente precisos.

En 1996 surge un desarrollo llamado JAVA (extensión de C++). Su filosofía es aprovechar el software existente. Facilitar la adaptación del mismo a otros usos diferentes a los originales sin necesidad de modificar el código ya existente.

En 1997-1998 se desarrollan herramientas 'CASE' orientadas a objetos (como el diseño asistido por computadora).

De 1998 a la fecha se desarrolla la arquitectura de objetos distribuidos RMI, Corba, COM, DCOM.

Actualmente la orientación a objetos parece ser el mejor paradigma, no obstante, no es una solución a todos los problemas. Trata de eliminar la crisis del software.

1.6. UML – Lenguaje Unificado de Modelado

1.6.1. Introducción

Esta guía introduce el Lenguaje Unificado de Modelado (UML), versión 1.1. Analiza los diagramas que componen UML y ofrece acercamientos a casos de uso guiados sobre

cómo estos diagramas se usan para modelar sistemas. La guía también trata los mecanismos de extensibilidad de UML, los cuales permiten ampliar su notación y su semántica. También sugiere la extensión de UML mediante dos técnicas no incorporadas: tarjetas CRC (Class, Responsabilities and Collaboration) para análisis guiados por la responsabilidad, y diagramas de Entidad de Relación (ER) para modelar bases de datos relaciónales.

1.6.2. Definición

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de casos de uso para modelar los procesos 'business'.
- 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.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. 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.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un meta modelo orientado a objetos de semántica y notación estándares. UML, en su versión 1.0, fue propuesto como una respuesta a esta petición en enero de 1997. Hubo otras cinco propuestas rivales. Durante el transcurso de 1997, los seis promotores de las propuestas, unieron su trabajo y presentaron al OMG un documento revisado de UML, llamado UML versión 1.1. Este documento fue aprobado por el OMG en Noviembre de 1997. El OMG llama a este documento OMG UML versión 1.1. El OMG está actualmente en proceso de mejorar una edición técnica de esta especificación, prevista su finalización para el 1 de abril de 1999.

1.6.3. UML ofrece notación y semántica estándar

UML prescribe una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. Previamente, un diseño orientado a objetos podría haber sido modelado con cualquiera de la docena de metodologías populares, causando a los revisores tener que aprender las semánticas y notaciones de la metodología empleada antes que intentar entender el diseño en sí. Ahora con UML, diseñadores diferentes modelando sistemas diferentes pueden sobradamente entender cada uno los diseños de los otros.

1.6.4. UML no es un método

Aun así, UML no prescribe un proceso o método estándar para desarrollar un sistema. Hay varias metodologías existentes; entre las más populares se incluyen las siguientes:

Catalysis: Un método orientado a objetos que fusiona mucho del trabajo reciente en métodos orientados a objetos, y además ofrece técnicas específicas para modelar componentes distribuidos.

Objetory: Un método de caso de uso guiado para el desarrollo, creado por Ivar Jacobson.

Shlaer/Mellor: El método para diseñar sistemas de tiempo real, puesto en marcha por Sally Shlaer y Steven Mellor en dos libros de 1991, “Ciclos de vida de objetos, modelando el mundo en estados y ciclos de vida de objetos”, “Modelando el mundo en datos” (Prentice Hall). Shlaer/Mellor continúan actualizando su método continuamente (la actualización más reciente es el OOA96 report), y recientemente publicaron una guía sobre cómo usar la notación UML con Shlaer/Mellor.

Fusión: Desarrollado en Hewlett Packard a mediados de los noventa como primer intento de un método de diseño orientado a objetos estándar. Combina OMT y Booch con tarjetas CRC y métodos formales.
(www.hpl.hp.com/fusion/file/teameps.pdf)

OMT: La Técnica de modelado de objetos fue desarrollada por James Rumbaugh y otros, y publicada en el libro de gran influencia "Diseño y Modelado Orientado a Objetos" (Prentice Hall, 1991). Un método que propone análisis y diseño 'iterative', más centrado en el lado del análisis.

Booch: Parecido al OMT, y también muy popular, la primera y segunda edición de "Diseño orientado a objetos, con aplicaciones" (Benjamin Cummings, 1991 y 1994), (Object-Oriented Design, With Applications), detallan un método ofreciendo también diseño y análisis 'iterative', centrándose en el lado del diseño.

Además, muchas organizaciones han desarrollado sus propias metodologías internas, usando diferentes diagramas y técnicas con orígenes varios. Ejemplos son el método Catalyst por Computer Sciences Corporation (CSC) o el Worlwide Solution Design and Delivery Method (WSDDM) por IBM. Estas metodologías difieren, pero generalmente combinan análisis de flujo de trabajo, captura de los requisitos, y modelado de negocio

con modelado de datos, con modelado de objetos usando varias notaciones (OMT, Booch, etc), y algunas veces incluyendo técnicas adicionales de modelado de objetos como casos de uso y tarjetas CRC. La mayoría de estas organizaciones están adoptando e incorporando el UML como la notación orientada a objetos de sus metodologías.

Algunos modeladores usarán un subconjunto de UML para modelar 'what they're after', por ejemplo simplemente el diagrama de clases, o solo los diagramas de clases y de secuencia con casos de uso. Otros usarán una suite más completa, incluyendo los diagramas de estado y actividad para modelar sistemas de tiempo real, y el diagrama de implementación para modelar sistemas distribuidos. Aun así, otros no estarán satisfechos con los diagramas ofrecidos por UML, y necesitarán extender UML con otros diagramas como modelos relacionales de datos y 'CRC cards'.

1.6.5. Una perspectiva general de UML

a) Una vuelta por un caso de uso

Una vez más, UML es una notación, no un método. No prescribe un proceso para modelar un sistema. No obstante, como UML incluye los diagramas de casos de uso, se le considera estar dotado de una aproximación al diseño centrada en el problema con los casos de uso. El diagrama de caso de uso nos da el punto de entrada para analizar los requisitos del sistema, y el problema que necesitamos solucionar.

b) Casos de uso y diagramas de interacción

Un caso de uso se modela para todos los procesos que el sistema debe llevar a cabo. Los procesos se describen dentro del caso de uso por una descripción textual o una secuencia de pasos ejecutados. Los diagramas de actividad se pueden usar también para modelar escenarios gráficamente. Una vez que el comportamiento del sistema está captado de esta manera, los casos de uso se examinan y amplían para mostrar qué objetos se interrelacionan para que ocurra este comportamiento. Los diagramas de colaboración y de secuencia se usan para mostrar las relaciones entre los objetos.

c) Clases y diagramas de implementación

Conforme se van encontrando los objetos, pueden ser agrupados por tipo y clasificados en un diagrama de clase. Es el diagrama de clase el que se convierte en el diagrama central del análisis del diseño orientado a objetos, y el que muestra la estructura estática del sistema. El diagrama de clase puede ser dividido en capas: aplicación, y datos, las cuales muestran las clases que intervienen con la interfaz de usuario, la lógica del software de la aplicación, y el almacenamiento de datos respectivamente. Los diagramas de componentes se usan para agrupar clases en componentes o módulos. La distribución general del hardware del sistema se modela usando el diagrama de implementación.

d) Diagramas de estado

El comportamiento en tiempo real de cada clase que tiene comportamiento dinámico y significativo, se modela usando un diagrama de estado. El diagrama de actividad puede ser usado también aquí, esta vez como una extensión del diagrama de estado, para mostrar los detalles de las acciones llevadas a cabo por los objetos en respuesta a eventos internos. El diagrama de actividad se puede usar también para representar gráficamente las acciones de métodos de clases.

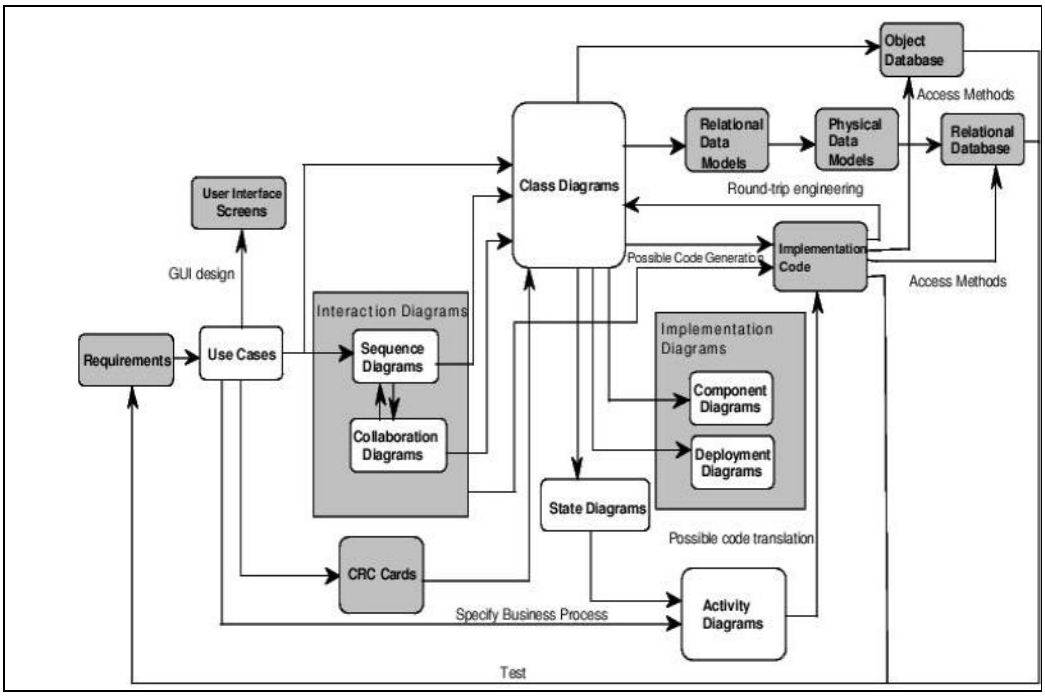


Figura 1-9 Caso de uso guiado para el desarrollo orientado a objetos con UML

e) Implementando el diseño

La implementación del sistema trata de traducir información desde múltiples modelos UML en código y estructura de bases de datos. Cuando se modela un sistema grande, es útil fragmentar el sistema en su capa 'business' (incluyendo los objetos de la interfaz de usuario), su capa de aplicación (incluyendo los objetos de implementación), y su capa de datos (incluyendo la estructura de la base de datos y el acceso a objetos).

f) Implementando la aplicación

El diagrama de clase se usa para generar una estructura base del código en el lenguaje seleccionado. Información de los diagramas de interacción, estado, y actividad, puede ofrecer detalles de la parte que describe el procedimiento del código de implementación.

g) Implementando el diseño de bases de datos

La capa de datos del diagrama de clase se puede usar para implementar directamente un diseño orientado a objetos de una base de datos, o, como extensión de UML, puede ser referenciado en un diagrama de relación de entidad para más análisis de relaciones de entidad. Está en el diagrama de relación de entidad (ER diagram, entity relationship) el cual relaciona entre entidades que pueden ser modeladas basadas en atributos clave. El diagrama de relación de entidad lógico ofrece una base desde la cual construir un diagrama físico representando las tablas y relaciones actuales de la base de datos relacional.

h) Probar teniendo en cuenta los requisitos

Los casos de uso se utilizan también para probar el sistema y ver si satisface los requisitos iniciales. Los pasos de los casos de uso van llevando a cabo para determinar si el sistema está satisfaciendo los requisitos del usuario.

1.7. Bases de datos

Una base de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

1.7.1. Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos, por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos:

a) Bases de datos jerárquicas

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.
Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

b) Bases de datos de red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

c) Bases de datos relacionales

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José, California, no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se

conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las “tuplas”, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Durante los años 1980-1989, la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.

d) Bases de datos orientadas a objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos ya vistos anteriormente.

2. Requerimientos

2.1. Concepto de requerimiento

Un requerimiento es una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal. Una representación documentada de una condición o capacidad.

Los requerimientos pueden dividirse en requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

2.2. Características de los requerimientos

Las características de un requerimiento son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo. A continuación se presentan las más importantes.

Necesario: es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
Conciso: es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
Completo: está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
Consistente: es consistente si no es contradictorio con otro requerimiento.
No ambiguo: no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.
Verificable: es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.

2.3. Definición del problema

Actualmente el Programa de Tecnología en Computo (PROTECO), se encuentra en una etapa de desarrollo importante para la cuál necesita optimizar procesos administrativos así como recursos económicos. Se requiere desarrollar un sistema, que permita al programa de la DIE (División de Ingeniería Eléctrica) organizar los procesos administrativos entre los cuales destacan:

- Administración de los cursos.
- Administración de los becarios del programa.
- Administración del material didáctico para los cursos.
- Avisos vía e-mail a los alumnos.
- Sección de alumnos.

Es necesario implementar un sistema que ofrezca un ambiente amigable de consulta, a través de la proyección clara y sencilla de dichos datos, además de facilitar la información y materiales con los que cuenta el programa a los miembros del PROTECO, también se dé a conocer y se ofrezca información al público en general, haciendo de este portal una herramienta importante para cualquier persona interesada en los diversos tópicos de la Tecnología en Computación.

2.4. Requerimientos

Administración de los cursos

PROTECO imparte varios cursos en diferentes periodos del semestre, en general son tres: los cursos sabatinos, los cursos para Prebecarios durante todo un semestre y por último los cursos que se desarrollan durante el periodo intersemestral. Todos los cursos necesitan tener una fecha de inicio y de terminación, hora de inicio y fin, un número máximo de alumnos para evitar que se saturen el curso, es necesario asignar el lugar en el que se va a impartir, existen una serie de opciones definidas previamente, estas son las salas de computo con que cuenta la DIE, así como el cubículo con que cuenta PROTECO. Cada curso tiene asignados dos o mas instructores que deben ser becarios. Para cada alumno que esta inscrito en un curso se debe tener una calificación que asigna el instructor.

Los cursos sabatinos tienen las características de que se imparte a los alumnos de la UNAM y que no tienen ningún costo, son cursos básicos para que los alumnos tengan una introducción a ciertos tópicos de computación como son los lenguajes de programación o sistemas operativos.

Para los alumnos considerados como Prebecarios se proporcionan cursos de formación de becarios, que se imparten de forma intensiva a las nuevas generaciones de becarios PROTECO.

En el caso de los Prebecarios no existe por el momento ningún costo para el alumno. y se tienen un conjunto de cursos definidos para los Prebecarios, se requiere colocar

material didáctico proporcionado por el instructor del curso, para que solo el Prebecario lo consulte.

En el caso de los cursos intersemestrales, el sistema debe permitir la inscripción a un curso cada vez que un alumno lo solicite, se le deben pedir sus datos personales: nombre completo, correo electrónico, la institución de procedencia, numero de cuenta en caso de que cuente con uno, carrera que cursa y semestre.

Los cursos intersemestrales tiene un costo dependiendo del tipo de actividad que realiza dicha persona, este costo se asigna tomando en cuenta si se trata de: un alumno de la UNAM sea cual sea su carrera o plantel, un alumno de otra institución diferente y por ultimo para el publico en general. Existe una excepción en la que no se realizara ningún costo debido a la obtención de una beca, que dependerá del procedimiento que determine el coordinador del PROTECO.

Existen un conjunto de cursos en diferentes niveles que se consideran como uno solo, a los que se les dio el nombre de paquetes, en el momento de inscribirse en el sistema se puede optar por un paquete y se asegura la inscripción en cada uno de los cursos que el paquete tiene asociado, obteniendo como ventaja un menor precio en comparación con la inscripción de los cursos individualmente.

Después de concluir con la inscripción del alumno de los cursos intersemestrales el alumno tendrá por impreso un comprobante de pago con los datos del curso y su costo, el cual debe presentar en caja para efectuar su pago, lo siguiente es entregar una copia del comprobante de pago en el cubículo del PROTECO para terminar con el proceso de inscripción. Se debe registrar el folio del comprobante de pago para tener un seguimiento de los ingresos que percibe el programa por esta actividad. Para el caso de la inscripción dentro de un paquete solo se entrega la ficha de pago que considera a todos los cursos que el paquete incluye y el costo total del paquete.

Para evitar grupos con más alumnos de los que se consideran para cada sala, se debe limitar la inscripción de alumnos a cursos, este límite esta definido por el número máximo de alumnos que se consideran para cada curso.

La aplicación debe mostrar una lista con todos los alumnos inscritos junto con el estado en el que se encuentra el registro del comprobante de pago, se debe ver cuales alumnos solo están inscritos en el curso y cuales ya efectuaron el pago pertinente.

Es necesario generar un reporte de todos aquellos alumnos que tienen calificación mayor o igual a ocho en el curso, esta calificación es registrada por cualquiera de los instructores del curso y se utiliza para hacer una constancia para el alumno que acreditado el curso.

Al término del periodo de cursos intersemestral se procede al trámite de cobro de beca para lo cual es necesario tener una relación de todos los folios que se percibieron en ese periodo.

Administración de los Becarios del programa.

Se desea tener una agenda con la información de todos los becarios del programa como es su nombre, e-mail, teléfono, carrera en la que está inscrito, número de cuenta, porcentaje de la carrera, promedio, generación, fecha de nacimiento, número de celular, dirección.
Para facilitar el trámite de beca para cada instructor se debe generar un recibo de beca con el formato establecido y los siguientes datos del programa: Jefe de Proyecto, Jefe del Departamento, Secretario, Proyecto, Periodo y monto de la beca tanto en número como en letra.

Administración del material didáctico para los cursos.

Se debe tener la opción de asociar material didáctico para los cursos, este material debe poder ser consultado por solo alumnos o becarios. En el caso específico de los manuales del curso debe permitir la lectura o impresión del mismo, sin modificar el contenido.
El material estará disponible para los alumnos durante un periodo de tiempo, el cual será especificado por el instructor del curso, y para los becarios este material se encuentra siempre disponible.

Inscripción a cursos de PROTECO sin condición

Se requiere facilitar el proceso de inscripción a los cursos que imparte PROTECO, como un primer objetivo se necesita que el alumno ingrese sus datos en el sistema y seleccione el curso o cursos que le interesan. El alumno debe recibir la ficha de pago correspondiente de modo que no pueda cambiar el costo del curso. El lugar que esta solicitando el alumno dentro del grupo se mantiene reservado en espera de que el alumno entregue su recibo de pago en el cubículo de PROTECO.

Avisos vía e-mail a los alumnos

En el caso de que los alumnos no acudan a entregar el comprobante de pago se debe enviar un mensaje al e-mail del alumno para solicitar que se entregue el mismo a la brevedad posible y terminar con el proceso de inscripción.

Para los alumnos con derecho a recibir una constancia se les debe avisar a través de e-mail, que pasen a recoger su constancia.
Se desea enviar la información de fechas, horarios y costo de los cursos que se impartirán en fechas próximas utilizando el registro previo de alumnos, obteniendo de esta forma una mayor difusión dentro del grupo de alumnos que ya han tomado un curso.

Sección de alumnos

En esta sección el alumno puede obtener el material del curso y su calificación. Para que el alumno tenga acceso a esta sección debe proporcionar su nombre de usuario y contraseña. Tiene la opción de enviar mensajes de texto a sus instructores o bien sus compañeros.

3. Análisis del sistema

3.1. Análisis de requerimientos

El análisis de requerimientos es la tarea que plantea la asignación de software a nivel de sistema y el diseño de programas (Figura 3-1). El análisis de requerimientos facilita al ingeniero de sistemas especificar la función y comportamiento de los programas, indicar la interfaz con otros elementos del sistema y establecer las ligas de diseño que debe cumplir el programa. El análisis de requerimientos permite al ingeniero refinar la asignación de software y representar el dominio de la información que será tratada por el programa. El análisis de requerimientos da al diseñador la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministra al técnico y al cliente, los medios para valorar la calidad de los programas, una vez que se haya construido.

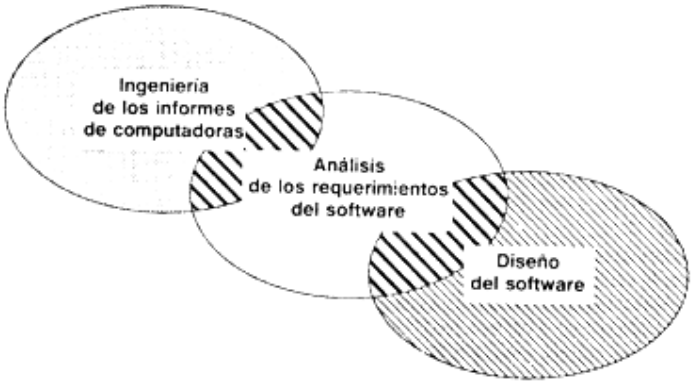


Figura 3-1 Análisis de requerimientos

3.1.1. Tareas del análisis

El análisis de requerimientos puede dividirse en cuatro áreas:

- Reconocimiento del problema
- Evaluación y síntesis
- Especificación
- Revisión.

Inicialmente, el analista estudia la especificación del sistema (si existe) y el plan de proyecto. Es importante comprender el contexto del sistema y revisar el ámbito de los programas que se usó para generar las estimaciones de la planificación. A continuación, debe establecerse la comunicación necesaria para el análisis, de forma que se asegure el reconocimiento del problema.

El analista debe establecer contacto con el equipo técnico y de gestión del usuario/cliente y con la empresa que vaya a desarrollar el software. El gestor del programa puede servir como coordinador para facilitar el establecimiento de los

caminos de comunicación. El objetivo del analista es reconocer los elementos básicos del programa tal como lo percibe el usuario/cliente.

La evaluación del problema y la síntesis de la solución es la siguiente área principal de trabajo en el análisis. El analista debe evaluar el flujo y estructura de la información, refinar en detalle todas las funciones del programa, establecer las características de la interfase del sistema y descubrir las ligas del diseño, cada una de las tareas sirve para descubrir el problema de forma que pueda sintetizarse un enfoque o solución global.

Las tareas asociadas con el análisis y especificación existen para dar una representación del programa que pueda ser revisada y aprobada por el cliente. En un mundo ideal el cliente desarrolla una especificación de requerimientos del software completamente por sí mismo. Esto se presenta raramente en el mundo real. En el mejor de los casos, la especificación se desarrolla conjuntamente entre el cliente y el analista.

Una vez que se hayan descrito las funcionalidades básicas, comportamiento, interfase e información, se especifican los criterios de validación para demostrar una comprensión de una correcta implementación de los programas. Estos criterios sirven como base para hacer una prueba durante el desarrollo de los programas. Para definir las características y atributos del software se escribe una especificación de requerimientos formal. Además, para los casos en los que se desarrolle un prototipo se realiza un manual de usuario preliminar.

Puede parecer innecesario realizar un manual de usuario en una etapa tan temprana del proceso de desarrollo, pero de hecho, este borrador del manual de usuario obliga al analista a tomar el punto de vista del usuario del software. El manual permite al usuario/cliente revisar el software desde una perspectiva de ingeniería humana y frecuentemente produce el comentario: "La idea es correcta pero esta no es la forma en que pensé que se podría hacer esto". Es mejor descubrir tales comentarios lo más tempranamente posible en el proceso.

Los documentos del análisis de requerimiento (especificación y manual de usuario) sirven como base para una revisión conducida por el cliente y el técnico. La revisión de los requerimientos casi siempre produce modificaciones en la función, comportamiento, representación de la información, ligas o criterios de validación. Además, se realiza una nueva apreciación del plan del proyecto de software para determinar si las primeras estimaciones siguen siendo validas después del conocimiento adicional obtenido durante el análisis.

3.2. UML - Nivel conceptual

El Lenguaje Unificado de Modelado (UML) es una notación gráfica para dibujar diagramas de conceptos de software. Se puede utilizar para dibujar diagramas de un dominio del problema, un diseño de software propuesto o una implementación de un software ya completado. Fowlier [1] describe estos tres niveles diferentes como *Conceptual*, de *Especificación* y de *Implementación*.

[1] **[Fowlier 1999a]:** Martin Fowler and Kendall Scout, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd ed. Reading, Mass.: Addison-Wesley,1999.

Los diagramas en el nivel Conceptual no están tan fuertemente relacionados con el código fuente, solo describen el problema.

3.2.1. Actor.

Es un usuario del sistema, que necesita o usa algunos de los casos de uso. Se representa mediante un símbolo como el de la Figura 3-2 , acompañado de un nombre significativo, si es necesario.

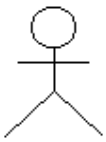


Figura 3-2 Actor

Para cubrir los requerimientos de PROTECO se identificaron cinco actores, los cuales se encuentran descritos en la tabla siguiente y que realizan las actividades especificadas en la sección de casos de uso.

Actor	Descripción
Coordinador PROTECO	Usuario que tiene la facultad de ver los reportes genéralas del programa.
Alumno	Usuario que esta relacionado a un curso.
Becario	Usuario que es un becario del PROTECO.
Instructor	Usuario que realiza las actividades de un instructor en un curso.
Coordinador de cursos	Usuario que realiza las tareas de administración de los cursos.
Administrador	Usuario que realiza la administración de los becarios.

Tabla 3-1 Actores del sistema

3.2.2. Caso de uso.

Cada caso de uso es una operación completa desarrollada por los actores y por el sistema en un diálogo. El conjunto de casos de uso representa la totalidad de operaciones desarrolladas por el sistema.

Coordinador PROTECO

Nombre: Consultar la lista de alumnos
Descripción: Muestra una lista de alumnos inscritos en un curso.
Curso principal:
1. El usuario selecciona la opción "Consultar listas".
2. Se muestra una lista con todos los cursos que están en proceso de inscripción.
3. Selecciona de la lista de los cursos.
4. Se debe mostrar una tabla que contenga los alumnos inscritos hasta ese momento en el curso.

Nombre: Consultar reportes de los cursos
Descripción: Consultar diferentes reportes del estado en el que se encuentra el programa.
Curso principal:
1. El usuario selecciona la opción "Reportes".
2. Se muestran los semestres disponibles en el sistema, junto con una lista con los reportes disponibles.
3. Se selecciona un número de semestre y un reporte.
4. Se muestra la tabla con los datos solicitados en el reporte.

Nombre: Ver lista de becarios
Descripción: Muestra una lista de los becarios del programa. Al seleccionar un becario se muestran los datos del becario.
Datos del becario :
- Nombre.
- E-mail.
- Teléfono.
- Carrera.
- Número de cuenta.
- Porcentaje de la carrera.
- Generación de becarios a la que pertenece.
- Fecha de nacimiento.
- Número telefónico del celular.
- Dirección.

Curso principal:
1. El usuario selecciona la opción "Becarios".
2. Se muestran una lista de las generaciones de becarios del programa.

3. Se selecciona un número de generación de becarios.
4. Se muestra la lista de becarios de esa generación.
5. Se selecciona uno de los becarios.
6. Se muestra la información general del becario.

Alumno

Nombre:	Consultar apuntes
Descripción:	Permite consultar y descargar las notas del curso, estos notas son archivos que deben estar disponibles para los alumnos que lo soliciten, pero solo se permite descargar aquellas notas, que son de los cursos en los que está inscrito el alumno.
Curso principal:	1. El usuario selecciona la opción "Notas del cursos". 2. Se muestra una lista con las notas que están disponibles para el curso en el que está inscrito. 3. Selecciona una de las notas que aparecen en la lista. 4. Se permite la operación de descarga del archivo.

Nombre:	Consultar calificación
Descripción:	Muestra las calificaciones del alumno.
Curso principal:	1. El usuario selecciona la opción "Calificación(es)". 2. Se muestra una lista con el nombre del curso o de los cursos en los que está registrado el alumno junto con su calificación final.

Nombre:	Consultar instructores
Descripción:	Muestra las la lista de instructores que tiene el alumno en el curso o curso que selecciono.
Curso principal:	1. El usuario selecciona la opción "Instructores". 2. Se muestra una lista con el nombre los becarios que participan como instructores en el curso.

Becario

Nombre:	Modificar datos personales
Descripción:	El usuario modifica sus datos personales.
Curso principal:	1. El usuario selecciona la opción "Datos personales". 2. Se muestran los datos actuales del becario. 3. Se selecciona la opción "Modificar datos". 4. Se capturan los nuevos datos personales. 5. Se actualiza la información del usuario con los nuevos datos que se capturaron

Nombre:	Inscribir alumnos
Descripción:	El usuario registra los datos del alumno y selecciona el

curso que el alumno desea tomar.

Curso principal:
1. El usuario selecciona la opción "Inscripción".
2. Se capturan los datos personales del alumno.
3. Se muestra una lista de los cursos disponibles.
4. Se selecciona el curso o cursos que desea tomar.

Nombre: Ver lista de alumnos.
Descripción: Al usuario se le muestran los alumnos inscritos un en curso.

Curso principal:
1. El usuario selecciona la opción "Consultar listas".
2. Se muestra una lista de todos los cursos.
3. Se selecciona un curso
4. Se muestra una lista con los alumnos inscritos en el cursos

Nombre: Ver datos de un alumno.
Descripción: Al usuario se le muestran los datos del alumno junto con los cursos en los que se encuentra inscrito.

Curso principal:
1. Se usa "*Ver lista de alumnos*".
2. Se selección un alumno.
3. Se muestra los datos del alumno y los datos del curso.

Nombre: Generar comprobante de pago.
Descripción: El usuario genera el comprobante de pago. Para cada curso se debe generar una ficha de pago con el formato que se maneja en PROTECO. Es necesario que se mantenga la integridad de los datos que se presentan la ficha de pago.

Curso principal:
1. Se usa "Ver datos de un alumno".
2. Se muestran los datos del alumno junto con los cursos en los que se encuentra inscrito.
3. Se selecciona la opción "Imprimir comprobante". Se debe generar la ficha de pago con los datos del alumno y del curso.

Nombre: Modificar datos del alumno
Descripción: El usuario modifica los datos del alumno.

Curso principal:
1. Se usa "Ver datos de un alumno".
2. Se muestran los datos del alumno junto con los cursos en los que se encuentra inscrito.
3. Se selecciona la opción "Modificar datos".
4. Se capturan los nuevos datos personales del alumno.
5. Se actualiza la información del usuario con los nuevos datos que se capturaron.

Nombre:	Registrar número de folio
Descripción:	El usuario registra el número de folio que se encuentra en el comprobante de pago.
Curso principal:	1. Se usa "Ver datos de un alumno". 2. Se muestran los datos del alumno junto con los cursos en los que se encuentra inscrito. 3. Se selecciona el curso en el cual el usuario registrara el número de folio. 4. Se selecciona la opción "Registrar folio". 5. Si el alumno tiene derecho a una beca : 5.1. Se captura el nombre de la persona que otorgo la beca. en caso contrario 5.2. Se captura el número de folio y el costo del curso.

Nombre:	Eliminar alumno
Descripción:	El usuario elimina un alumno de un curso.
Curso principal:	1. Se usa "Ver datos de un alumno". 2. Se muestran los datos del alumno junto con los cursos en los que se encuentra inscrito. 3. Se selecciona el curso que se desea eliminar. 4. Se elimina el alumno del curso seleccionado.

Nombre:	Consultar reportes
Descripción:	Al usuario se le muestran los reportes disponibles de los cursos, por el momento se necesita conocer todos los alumnos con folio y los alumnos con una calificación mayor o igual a ocho. Estos reportes son por cada semestre.
Curso principal:	1. El usuario selecciona la opción "Reportes". 2. Se muestra una lista de los reportes disponibles. 3. Se selecciona el semestre. 4. Se selecciona un reporte. 5. Se muestra una tabla con los datos que contiene el reporte.

Instructor

Nombre:	Publicar material
Descripción:	El usuario coloca un documento relacionado al curso en el cual se encuentra asignado como instructor, este documento debe estar disponible solo para que los alumnos del curso lo descarguen.

Curso principal:
1. El usuario selecciona la opción "Administrar material".
2. Se selecciona la opción "Subir material".
3. Se captura el curso en el cual se desea agregar este material.
4. Se capturan una descripción general del material.
5. Se selección el archivo.
6. Se procede a almacena el material.

Nombre: Eliminar Material
Descripción: El usuario tiene que ser el instructor del curso para realizar la tarea de eliminar el material que esta asociado al curso.
Curso principal:
1. El usuario selecciona la opción "Administrar material".
2. Se selecciona la opción "Eliminar material".
3. Se muestra una lista con todo el material disponible en el curso.
4. Se selección el material que se desea eliminar.
5. Se elimina el material.

Nombre: Modificar material
Descripción: El usuario puede modificar la descripción que acompaña al material.
Curso principal:
1. El usuario selecciona la opción "Administrar material".
2. Se selecciona la opción "Modificar material".
3. Se muestra una lista con todo el material disponible en el curso.
4. Se selección el material que se desea modificar.
5. Se elimina el material dicha materia.

Nombre: Calificar alumnos
Descripción: El usuario debe registrar las calificaciones de los alumnos, se necesita hacer el registro para cada uno de los cursos en los que el alumno se encuentra como instructor.
Curso principal:
1. El usuario selecciona la opción "Calificar alumnos".
2. Se muestra una lista con los cursos en los que el usuario se encuentra asignado como instructor.
3. Se selecciona un curso.
4. Se muestra una lista de todos los alumnos inscritos el curso.
5. Se captura la calificación que debe ser un número decimal de cero a diez.
6. Se almacena la calificación del alumno.

Coordinador de cursos

Nombre:	Alta de un curso
Descripción:	El usuario registra los datos para dar de alta un curso
Curso principal:	1. El usuario selecciona la opción “Cursos”. 2. Selecciona la opción “Alta”. 3. Captura los siguientes datos del curso: • Nombre del curso. • Horario. • Fecha de inicio. • Fecha de terminación. • Semestre. • Sala. • Tipo: intersemestral, para Prebecario o sabatino. • Costo. • Número máximo de alumnos. 4. Se registra el curso

Nombre:	Baja de un curso
Descripción:	El usuario da de baja un curso.
Curso principal:	1. El usuario selecciona la opción “Cursos”. 2. Selecciona la opción “Baja”. 3. Se muestra una lista con todos los cursos. 4. Selecciona un curso. 5. Se elimina el curso.

Nombre:	Alta elementos de catálogo
Descripción:	El usuario da de alta un elemento de cualquiera de los catálogos que conforman el sistema.
Curso principal:	1. El usuario selecciona la opción “Catálogos”. 2. Se muestra una lista de los catálogos disponibles. 3. Se selecciona el catálogo a editar. 4. Se agrega el elemento del catálogo seleccionado. 5. Se guardan los cambios realizados.

Nombre:	Baja elementos de catálogo
Descripción:	El usuario elimina un elemento de cualquiera de los catálogos que conforman el sistema.
Curso principal:	1. El usuario selecciona la opción “Catálogos”. 2. Se muestra una lista de los catálogos disponibles. 3. Se selecciona el catálogo a editar. 4. Se elimina el elemento del catálogo seleccionado. 5. Se guardan los cambios realizados.

Nombre:	Edita curso
Descripción:	El usuario se encarga de hacer las tareas de alta, baja, o cambio de un curso. Para esta tarea se debe seleccionar el curso a modificar sus datos.

Curso principal:
1. El usuario selecciona la opción “Cursos”.
2. Selecciona la opción “Modificar”.
3. Se muestra una lista con todos los cursos.
4. Selecciona un curso.
5. Se muestran los datos del curso.
6. Se captura los nuevos datos.
7. Se actualiza la información.

Nombre: Alta de un instructor

Descripción: El usuario asigna a los instructores en los cursos, es necesario que cada uno de los instructores este registrado previamente como un becario. Los cursos tienen dos tipos de instructores el titular y un adjunto. Existe casos con más de un adjunto, el máximo es de cuatro.

Curso principal:
1. El usuario selecciona la opción “Instructores”.
2. Selecciona la opción “Alta”.
3. Se muestra la lista de becarios registrados y la lista de cursos.
4. Se selecciona el becario y el curso.
5. Se selecciona el tipo de instructor: Titular o Adjunto.

Nombre: Alta de un instructor Prebecario

Descripción: El usuario se desea de asignar a los instructores en los cursos de prebecario, es necesario que cada uno de los instructores este registrado previamente como un becario. Los cursos tienen dos tipos de instructores el titular y un adjunto. Existe casos con más de un adjunto, el máximo es de cuatro.

Curso principal:
1. El usuario selecciona la opción “Prebecarios”.
2. Selecciona la opción “Alta”.
3. Se muestra la lista de becarios registrados y la lista de cursos.
4. Se selecciona el becario y el curso.
5. Se selecciona el tipo de instructor: Titular o Adjunto.

Nombre: Baja de un instructor

Descripción: El usuario elimina la asignación de un becario como instructor de un curso.

Curso principal:
1. El usuario selecciona la opción “Instructores”.
2. Selecciona la opción “Baja”.
3. Se muestra la lista de los instructores de cada curso.
4. Se selecciona el instructor que se desea dar de

baja.
5. Se elimina el la asignación que tiene el becario.

Nombre:	Baja de un instructor Prebecario
Descripción:	El usuario elimina la asignación de un becario como instructor de un curso de prebecario.
Curso principal:	1. El usuario selecciona la opción “Prebecarios”. 2. Selecciona la opción “Eliminar”. 3. Se muestra la lista de los instructores de cada curso. 4. Se selecciona el instructor que se desea dar de baja. 5. Se elimina el la asignación que tiene el becario.

Nombre:	Imprimir recibos beca
Descripción:	El usuario desea imprimir los recibos de beca de los instructores para ejercer las becas de los instructores de los cursos intersemestrales. Se solicita información previa a la impresión como el nombre de los jefes en turno de la facultad.
Curso principal:	1. El usuario selecciona la opción “Instructores”. 2. Selecciona la opción “Imprimir recibos”. 3. Se piden los datos para la impresión de recibos. 4. Se seleccionan los instructores. 5. Se imprimen los recibos en PDF.

Nombre:	Recordatorio
Descripción:	El usuario desea enviar un recordatorio vía email, a los alumnos que no han realizado sus pagos.
Curso principal:	1. El usuario selecciona la opción “Recordatorio”. 2. Selecciona la opción “Enviar recordatorio”. 3. Se selecciona a los alumnos para enviar el recordatorio. 4. Enviar recordatorio.

Nombre:	Alta curso
Descripción:	El usuario desea dar de alta un curso para impartir.
Curso principal:	1. El usuario selecciona la opción “Cursos”. 2. Selecciona la opción “Alta”. 3. Se selecciona el estado del curso, el tipo de curso, el nombre del curso, el nombre del paquete al que pertenece, las fechas de inicio y término, el número máximo de alumnos, semestre, etc. 4. Guardar curso.

Nombre:	Alta curso prebecario
Descripción:	El usuario desea dar de alta un curso para impartir.
Curso principal:	1. El usuario selecciona la opción “Prebecarios”.

2. Selecciona la opción “Alta Curso”.
3. Se selecciona el estado del curso, el tipo de curso, el nombre del curso, el nombre del paquete al que pertenece, las fechas de inicio y término, el número máximo de alumnos, semestre, etc.
4. Guardar curso.

Nombre: Edita curso
Descripción: El usuario desea editar un curso para impartir.
Curso principal:
1. El usuario selecciona la opción “Cursos”.
2. Selecciona la opción “Edición”.
3. Se edita la opción u opciones que se desean editar.
4. Actualizar curso.

Nombre: Edita curso prebecario
Descripción: El usuario desea editar un curso para impartir.
Curso principal:
1. El usuario selecciona la opción “Prebecarios”.
2. Selecciona la opción “Edición curso”.
3. Se edita la opción u opciones que se desean editar.
4. Actualizar curso.

Nombre: Elimina curso
Descripción: El usuario desea eliminar un curso.
Curso principal:
1. El usuario selecciona la opción “Cursos”.
2. Selecciona la opción “Eliminar curso”.
3. Se selecciona el curso a eliminar.
4. Eliminar curso.

Nombre: Elimina curso prebecario
Descripción: El usuario desea eliminar un curso.
Curso principal:
1. El usuario selecciona la opción “Cursos”.
2. Selecciona la opción “Eliminar curso”.
3. Se selecciona el curso a eliminar.
4. Eliminar curso.

Nombre: Eliminar alumnos
Descripción: El usuario desea eliminar alumnos.
Curso principal:
1. El usuario selecciona la opción “Alumnos”.
2. Selecciona la opción “Eliminar”.
3. Se selecciona el curso en donde se encuentra el alumno a eliminar.
4. Eliminar alumno.

Nombre: Eliminar prebecarios
Descripción: El usuario desea eliminar prebecarios.
Curso principal:
1. El usuario selecciona la opción “Prebecarios”.
2. Selecciona la opción “Eliminar”.

3. Se selecciona el curso en donde se encuentra el prebecario a eliminar.
4. Eliminar prebecario.

Nombre: Bloquear alumnos
Descripción: El usuario desea bloquear alumnos.
Curso principal:
1. El usuario selecciona la opción “Alumnos”.
2. Selecciona la opción “Bloquear”.
3. Se selecciona el curso en donde se encuentra el alumno a bloquear.
4. Bloquear alumno.

Nombre: Bloquear prebecarios
Descripción: El usuario desea bloquear prebecarios.
Curso principal:
1. El usuario selecciona la opción “Prebecarios”.
2. Selecciona la opción “Bloquear”.
3. Selecciona el curso en donde se encuentra el prebecario a bloquear.
4. Bloquear prebecario.

Nombre: Alta prebecarios
Descripción: El usuario desea dar de alta prebecarios.
Curso principal:
1. El usuario selecciona la opción “Prebecarios”.
2. Selecciona la opción “Alta”.
3. Se llenan los datos del prebecario.
4. Dar de alta prebecario.

Nombre: Convertir prebecario en becario
Descripción: El usuario desea convertir al prebecario en becario.
Curso principal:
1. El usuario selecciona la opción “Prebecarios”.
2. Selecciona la opción “Convertir en becario”.
3. Selecciona a los prebecarios a cambiar.
4. Actualizar datos.

Nombre: Ver lista prebecarios
Descripción: El usuario desea la lista de los prebecarios.
Curso principal:
1. El usuario selecciona la opción “Prebecarios”.
2. Selecciona la opción “Ver lista”.

Administrador

Nombre: Administrar becarios
Descripción: Se encarga de dar de alta un becario, baja o modificar la información de este, además de actualizar el login y

password de estos.

Curso principal:
1. El usuario selecciona la opción “Becarios” del menú
2. Selecciona las opciones de Alta, Baja, Edición de becario etc. Dependiendo de lo que requiera.

Nombre: Alta administrador

Descripción: Se da de alta al nuevo administrador de una lista de becarios.

Curso principal:
1. El usuario selecciona la opción “Administrador” del menú
2. Selecciona al nuevo becario que será administrador.

Nombre: Alta coordinador

Descripción: Se da de alta al nuevo coordinador de una lista de becarios.

Curso principal:
1. El usuario selecciona la opción “Coordinador” del menú
2. Selecciona al nuevo becario que será coordinador.

3.2.3. Diagrama de casos de uso.

Un diagrama de casos de uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

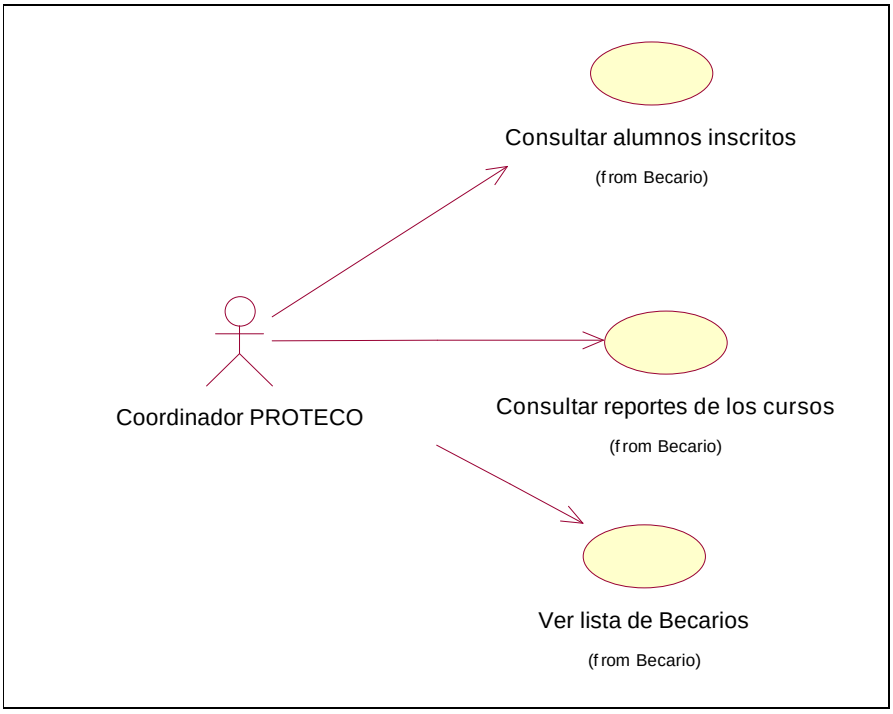


Figura 3-3 Diagrama de casos de uso del coordinador PROTECO.

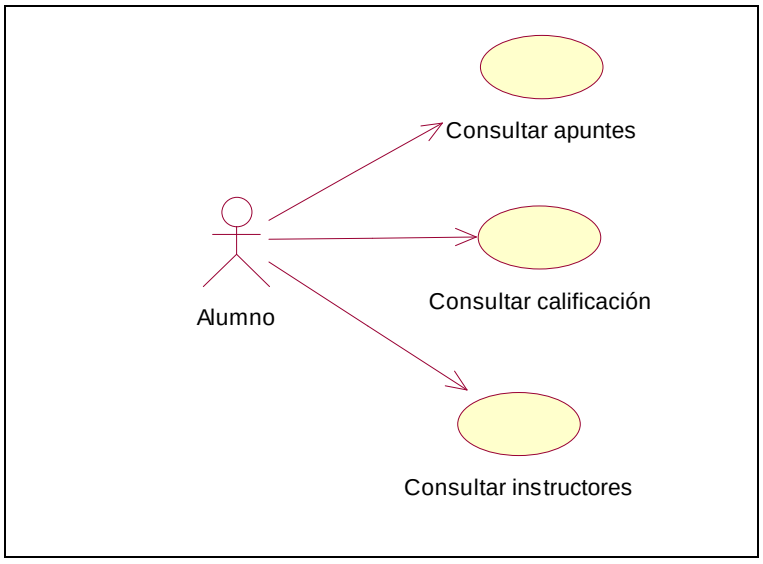


Figura 3-4 Diagrama de casos de uso del alumno.

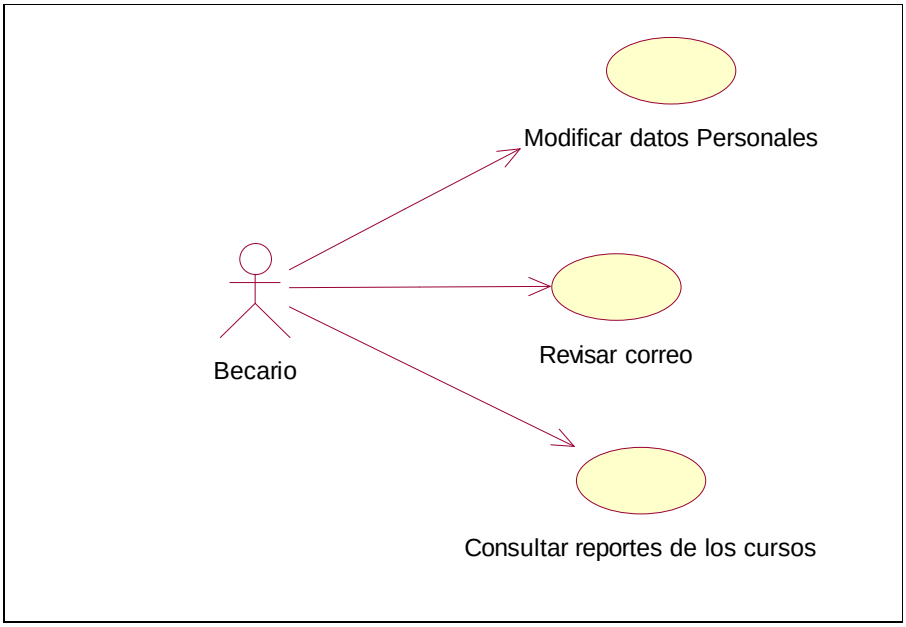


Figura 3-5 Diagrama de casos de uso del becario.

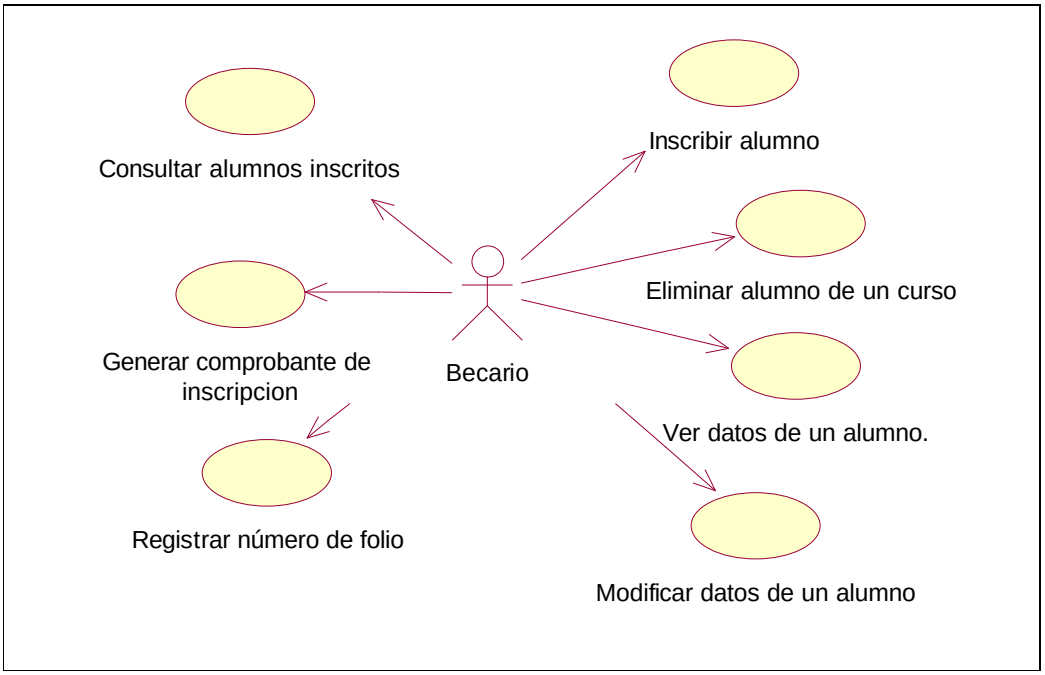


Figura 3-6 Diagrama de casos de uso del becario (administrar alumno).

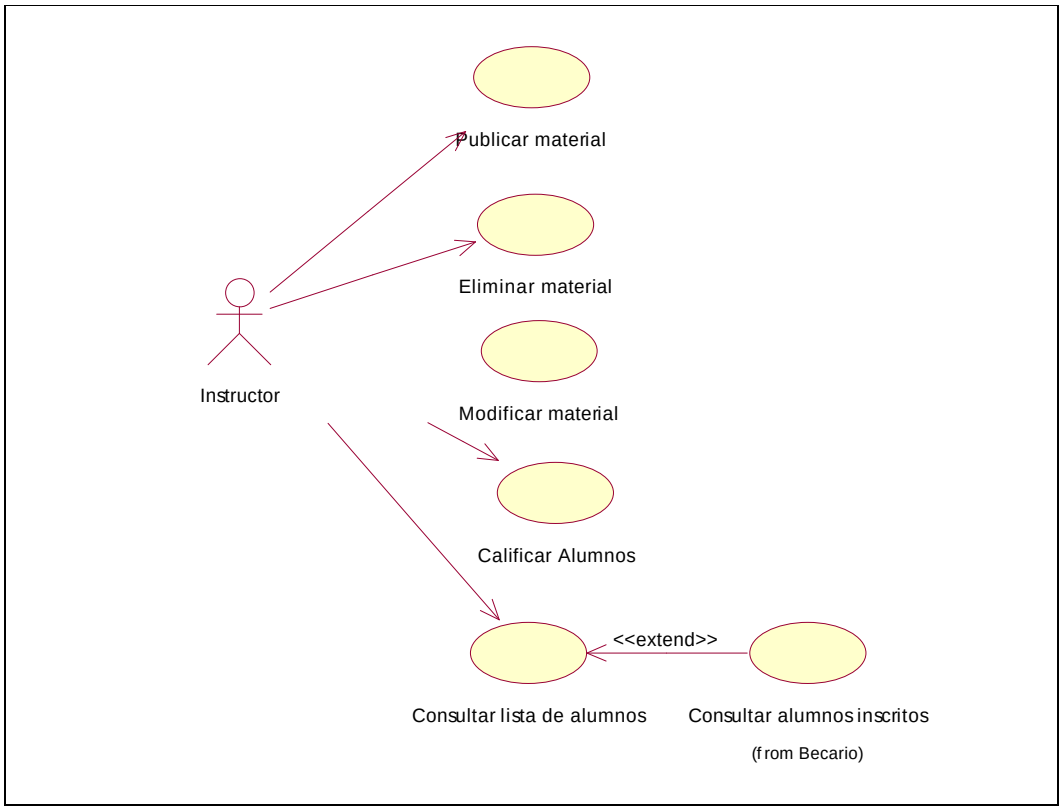


Figura 3-7 Diagrama de casos de uso del instructor.

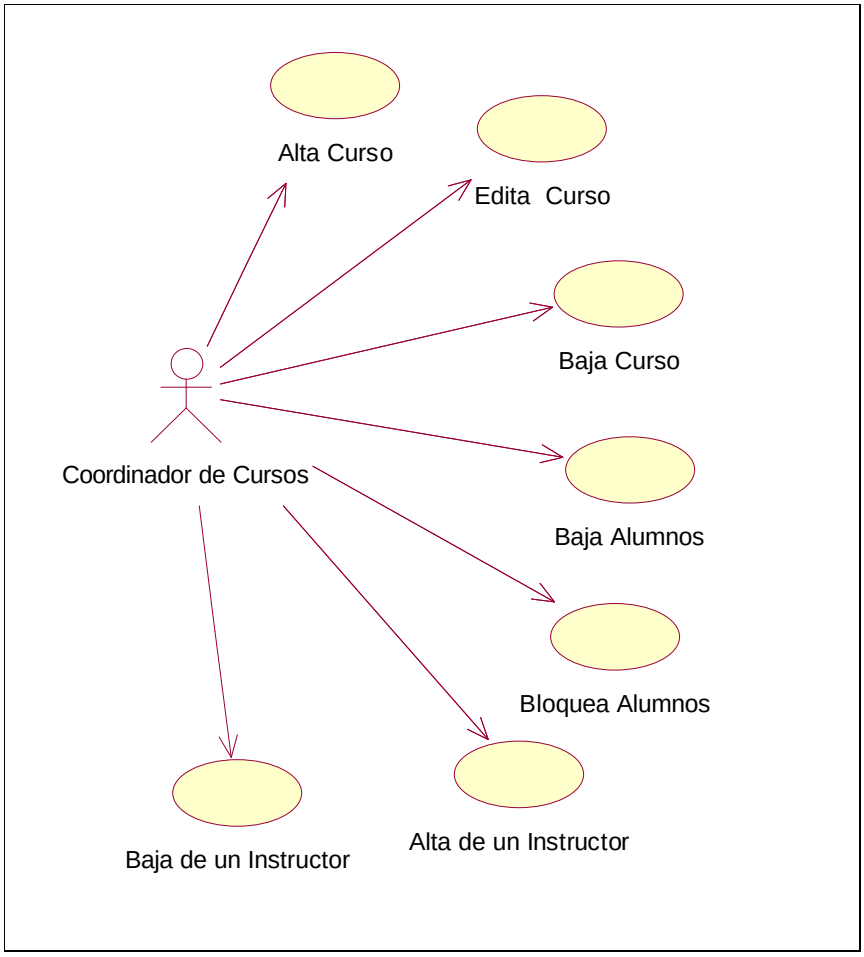


Figura 3-8 Diagrama de casos de uso del coordinador de cursos (sección de cursos).

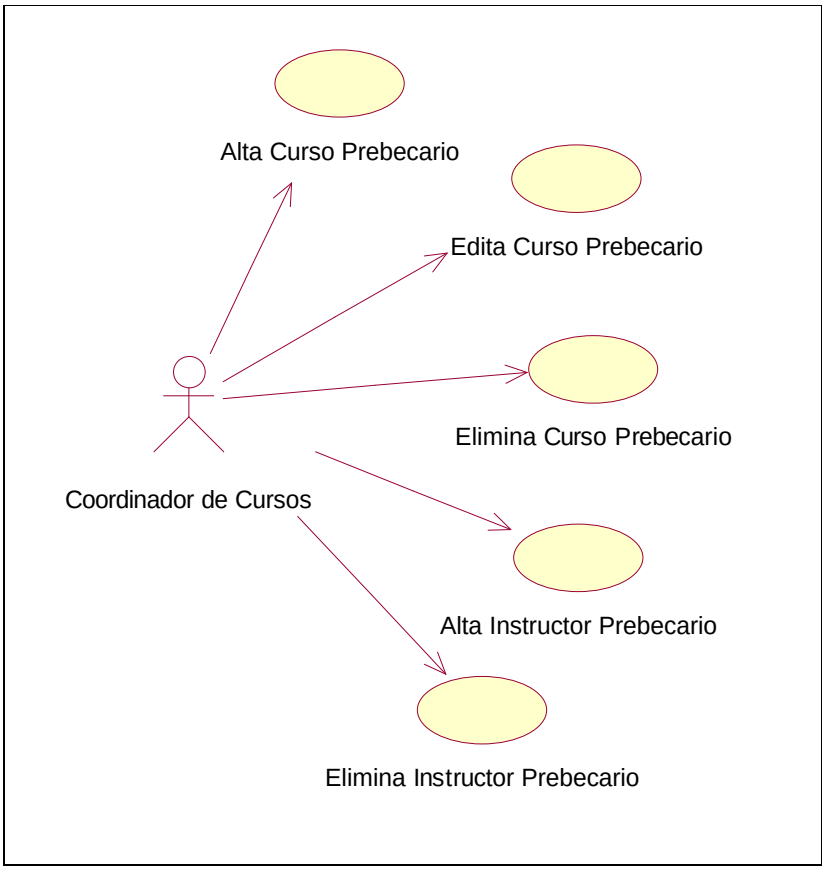


Figura 3-9 Diagrama de casos de uso del coordinador de cursos (cursos para prebecarios).

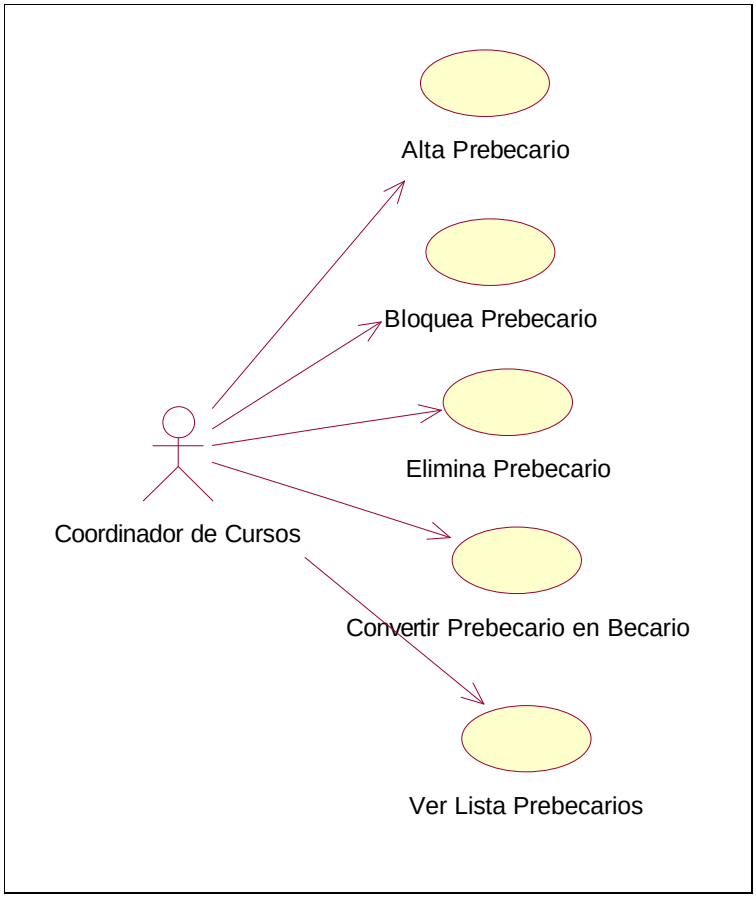


Figura 3-10 Diagrama de casos de uso del coordinador de cursos (sección de prebecarios).

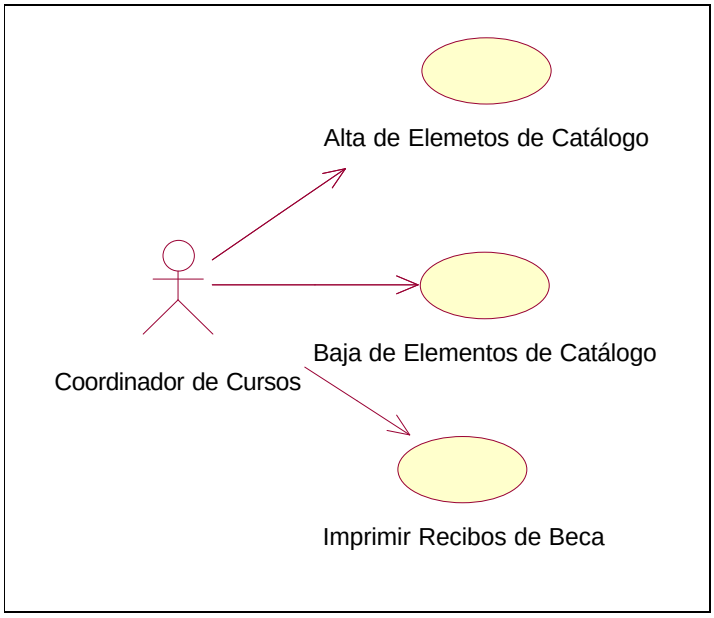


Figura 3-11 Diagrama de casos de uso del coordinador de cursos (varios).

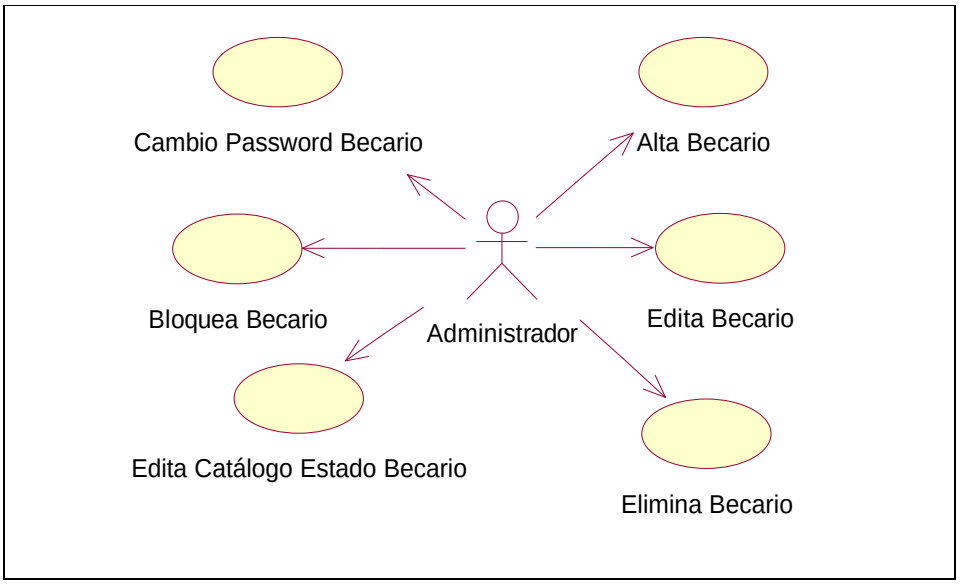


Figura 3-12 Diagrama de casos de uso del administrador (sección de becarios).

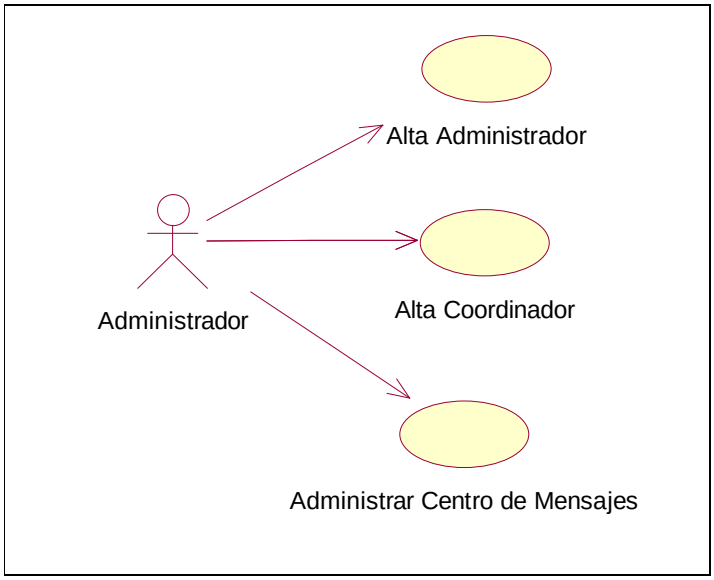


Figura 3-13 Diagrama de casos de uso del administrador (varios).

3.3. UML - Nivel de especificación

El propósito es que un diagrama del nivel de especificación se transforme en código fuente.

3.3.1. Diagramas de clases.

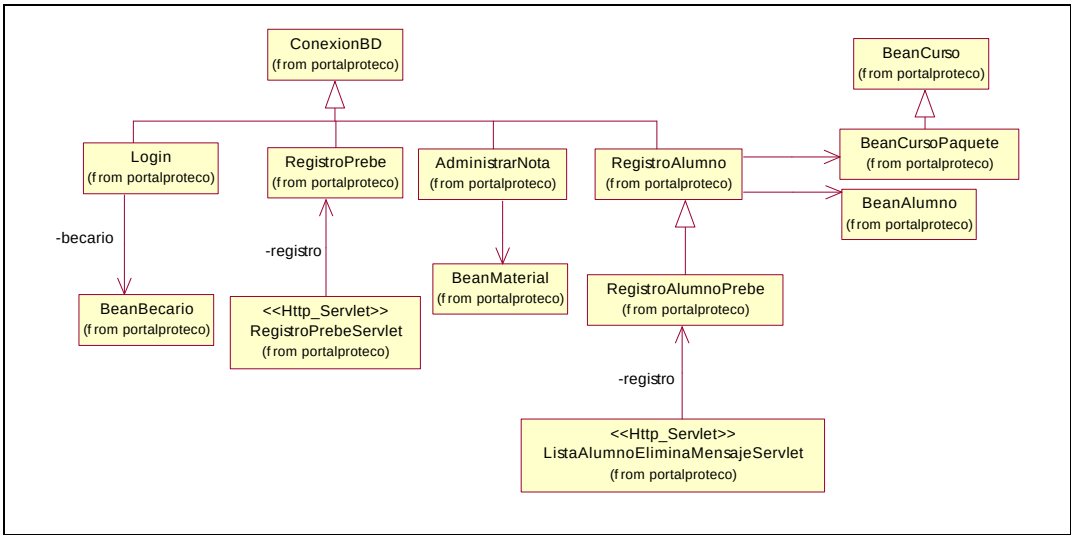


Figura 3-14 Diagrama de general de clases.

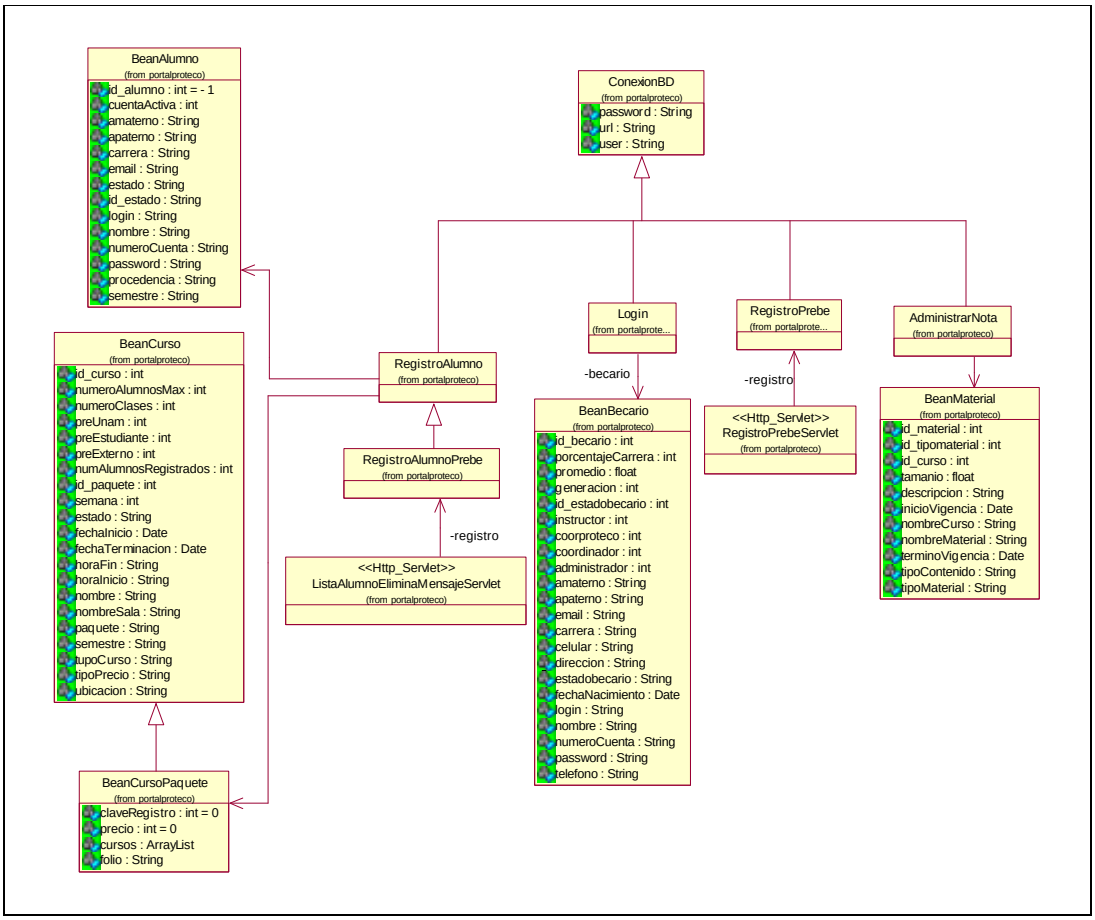


Figura 3-15 Diagrama general de clases (con atributos).

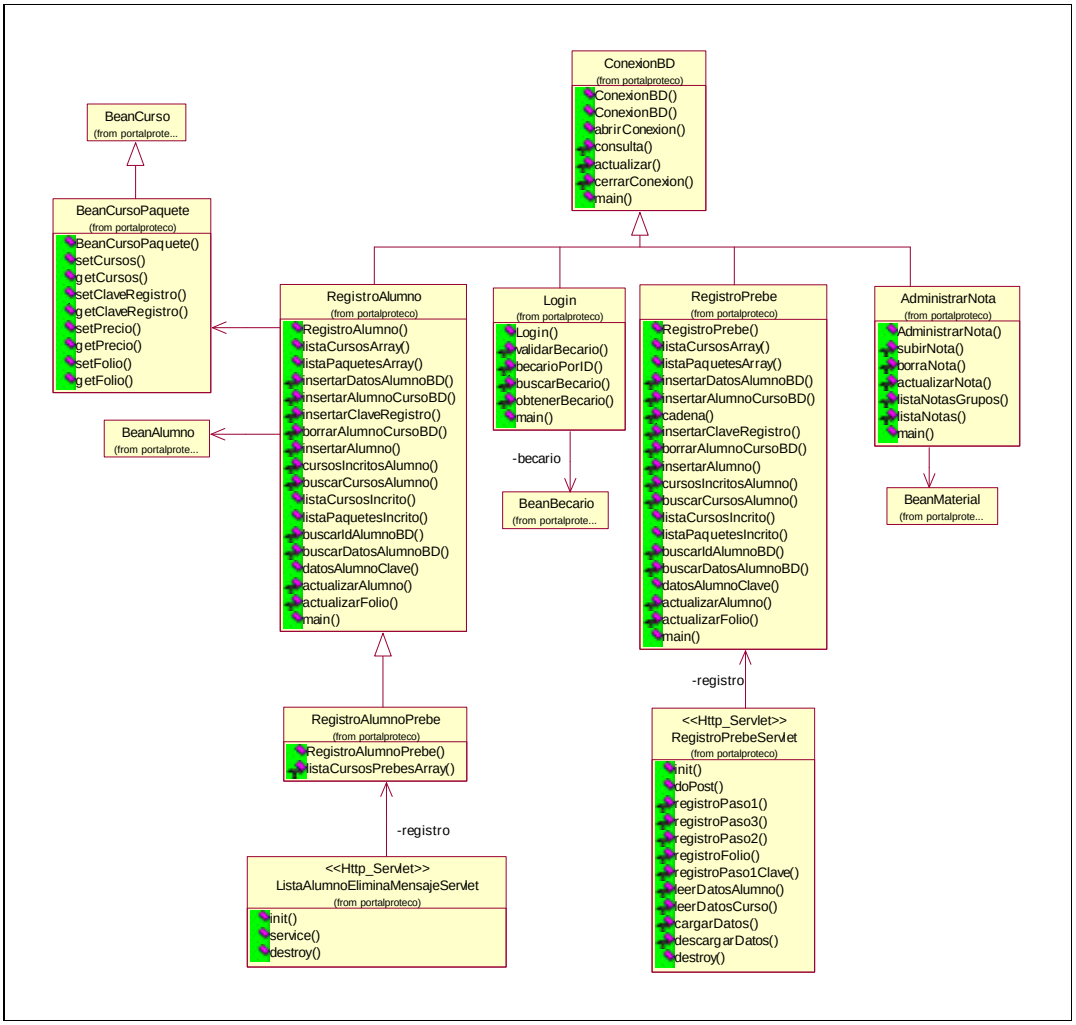


Figura 3-16 Diagrama general de clases (con métodos).

3.4. Especificación de requerimientos

La especificación de requisitos de software es la actividad en la cual se genera el documento, con el mismo nombre, que contiene una descripción completa de las necesidades y funcionalidades del sistema que será desarrollado; describe el alcance del sistema y la forma en como hará sus funciones, definiendo los requerimientos funcionales y los no funcionales.

En la Especificación de Requerimientos de Software (con las siglas en ingles SRS) se definen todos los requerimientos de hardware y software, diagramas, modelos de sistemas y cualquier otra información que sirva de soporte y guía para fases posteriores.

Es importante destacar que la especificación de requisitos es el resultado final de las actividades de análisis y evaluación de requerimientos; este documento resultante será utilizado como fuente básica de comunicación entre los clientes, usuarios finales, analistas de sistema, personal de pruebas, y todo aquel involucrado en la implementación del sistema.

Los clientes y usuarios utilizan la SRS para comparar si lo que se está proponiendo, coincide con las necesidades de la empresa. Los analistas y programadores la utilizan para determinar el producto que debe desarrollarse. El personal de pruebas elaborará las pruebas funcionales y de sistema en base a este documento. Para el administrador del proyecto sirve como referencia y control de la evolución del sistema.

La SRS posee las mismas características de los requerimientos: completa, consistente, verificable, no ambigua, factible, modificable, rastreable, precisa, entre otras. Para que cada característica de la SRS sea considerada, cada uno de los requerimientos debe cumplirlas; por ejemplo, para que una SRS se considere verificable, cada requerimiento definido en ella debe ser verificable; para que una SRS se considere modificable, cada requerimiento debe ser modificable y así sucesivamente. Las características de la SRS son verificadas en la actividad de validación.

La estandarización de la SRS es fundamental pues ayudará, entre otras cosas, a facilitar la lectura y escritura de la misma. Será un documento familiar para todos los involucrados, además de asegurar que se cubren todos los tópicos importantes.

Requerimientos de usabilidad

Nuestro principal criterio para hacer el sistema usable es la dificultad de realizar cada caso de uso de alta frecuencia. La dificultad depende del número de pasos, el conocimiento que el usuario debe tener en cada paso, las decisiones que el usuario debe realizar en cada paso, y la mecánica de cada paso.

La interfaz del usuario del sistema deberá ser tan familiar como sea posible a los usuarios que han usado otras aplicaciones Web o aplicaciones de escritorio.

Se requiere un fácil proceso de registro del alumno dentro de un curso, para el cual se sugieren dos pasos, el primero es introducir los datos personales del alumno en el sistema y el segundo será seleccionar solamente el curso o cursos que están disponibles o en estado de inscripción.

Para generar el formato que tiene la ficha de pago se deben obtener los datos del alumno y del curso que se registraron en el sistema previamente. Y la ficha debe estar disponible para su re-impresión en cualquier momento.

La tarea de registro de folio del comprobante de pago debe permitir seleccionar fácilmente al alumno y el curso, evitando errores al escribir el nombre del usuario.

Requerimientos de seguridad

El acceso será controlado con nombres de usuario y contraseñas para cada becario del PROTECO.

Solo los usuarios con derechos de administrador podrán acceder las funciones administrativas, los usuarios normales no podrán.

El acceso de un alumno al material del curso será controlado con un nombre de usuario y una contraseña.

Las contraseñas deberán tener de 6 a 20 caracteres de longitud.

El tiempo de tolerancia estando inactivo dentro del sistema para realizar cualquier tarea será menor a 10 minutos.

No se permitirá que se revisen secciones que no corresponden a la actividad que realiza el usuario, por ejemplo un alumno no tendrá acceso a la sección de becarios.

Requerimientos de desempeño y escalabilidad

Las expectativas de desempeño para el sistema se basan principalmente en el tiempo de respuesta al solicitar información hablando de un tiempo promedio de 3 segundos.

El sistema debe permitir a múltiples usuarios estar conectados y realizando las operaciones.

Se requiere que el sistema permita incluir nuevos módulos, y que permita inclusive el interactuar con otras aplicaciones compartiendo información, a través uno o varios medios utilizados en el momento en el que se escribe este documento, un ejemplo simple de ello seria el XSL.

Requerimientos de mantenimiento

Se requiere presentar un respaldo de todos los datos registrados de una manera periódica y con la posibilidad de solicitar en cualquier momento un respaldo.

Requerimientos funcionales

- Realizar el proceso de inscripción a un curso.
- Generar ficha de inscripción.
- Registrar el folio del comprobante de pago del alumno.
- Mostrar información de un alumno al igual que los cursos en los que esta inscrito.
- Modificar los datos del alumno.
- Lista de alumnos de cada grupo.
- Registrar las calificaciones de los alumnos.
- Lista de alumnos con derecho a constancia.
- Lista con todos los folios registrados para el control administrativo del PROTECO.
- Envío de e-mail recordando que se debe registrar el folio que tiene el recibo de pago.
- Administrar el material de los cursos.
- Manejar el alta de nuevos cursos y administrar los ya existentes.
- Registrar la información de todos los becarios del PROTECO.

- Generar el recibo de beca para los becarios que impartieron curso.
- Generar el recibo de beca para becarios en un concepto diferente al de los cursos.
- Registrar nuevos becarios en el sistema.
- Modificar los datos de los becarios.

Requerimientos no funcionales

- Interfase de usuario amigable.
- Manual de usuario.
- Documentación del sistema.

4. Diseño del Sistema

Para cubrir los requerimientos descritos en el capitulo anterior se tomaron en cuenta las siguientes arquitecturas que se pueden utilizar para generar la aplicación:

- Aplicación de escritorio para proceso simple.
- Cliente-servidor.
- Aplicación Web: servidor Web/servidor de aplicaciones, base de datos.
- Servicio Web simple: servidor de Web, base de datos.
- Cliente-a-Cliente sin servidor central.
- Malla de computadoras / servidores distribuidos.

Se opto por una Arquitectura de aplicación Web de tres capas, como la que se muestra en la Figura 4-1.

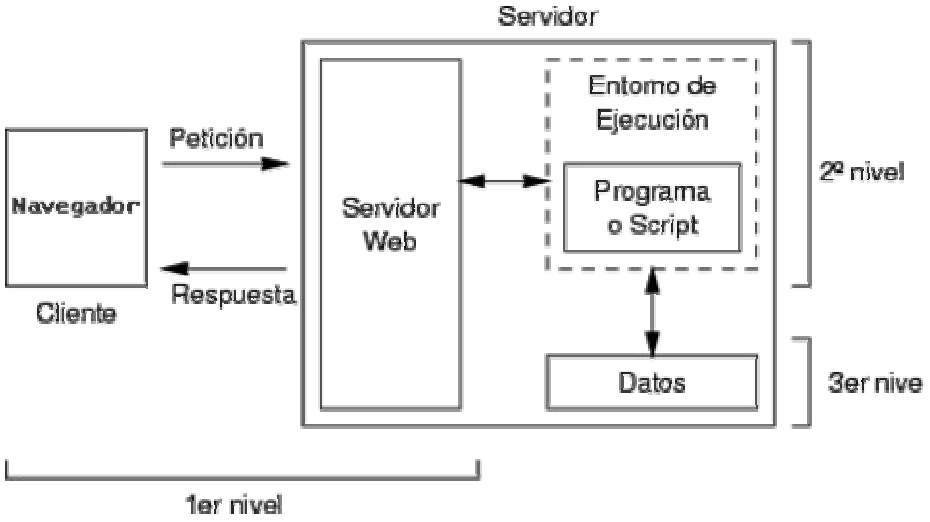


Figura 4-1 Arquitectura Web de tres capas.

La decisión se basa en la necesidad de permitir que las inscripciones a un curso se lleven a cabo sin la necesidad de presentarse en el cubículo del PROTECO y de manera inmediata se obtenga un recibo correspondiente para presentar el pago en la caja de la Facultad de Ingeniería.
Una ventaja más que nos da el Web, es el acceso en todo momento al material de los cursos para los alumnos que lo soliciten, dicho material estará en formato digital cumpliendo con la condición de no permitir su modificación.

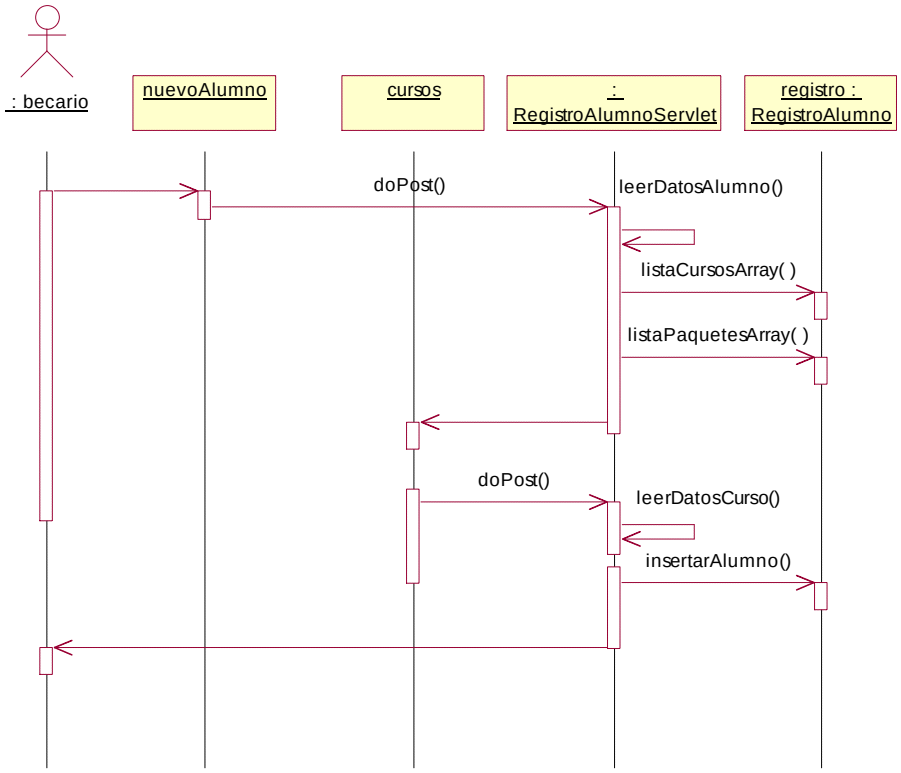
4.1. UML - Nivel de implementación

El propósito es que un diagrama del nivel de implementación describa un código fuente existente.

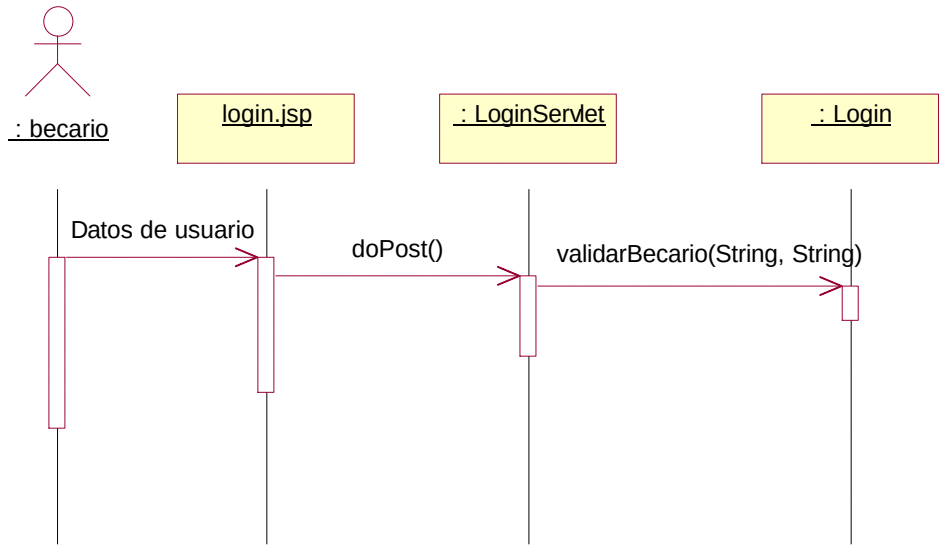
4.1.1. Diagrama de secuencia.

a) Diagramas de secuencia para los casos de uso de becario.

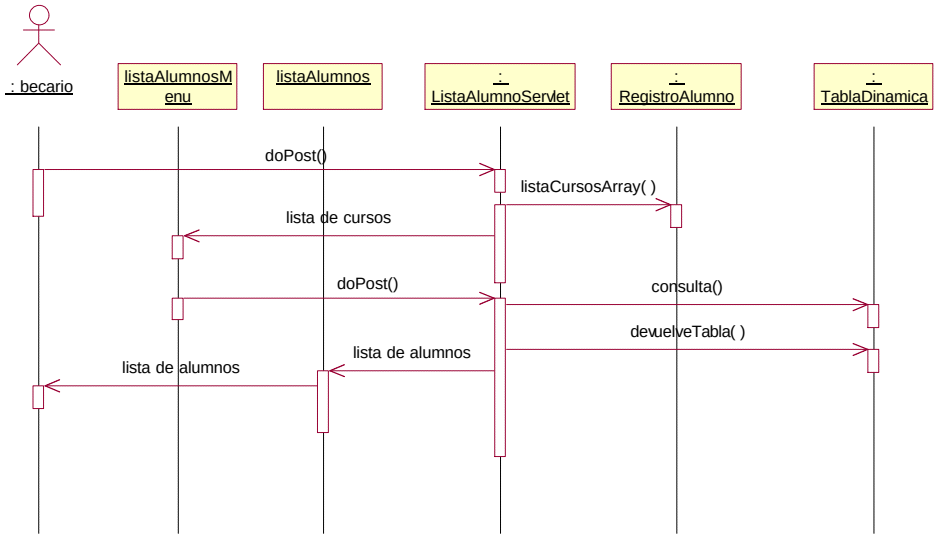
Inscribir alumno.



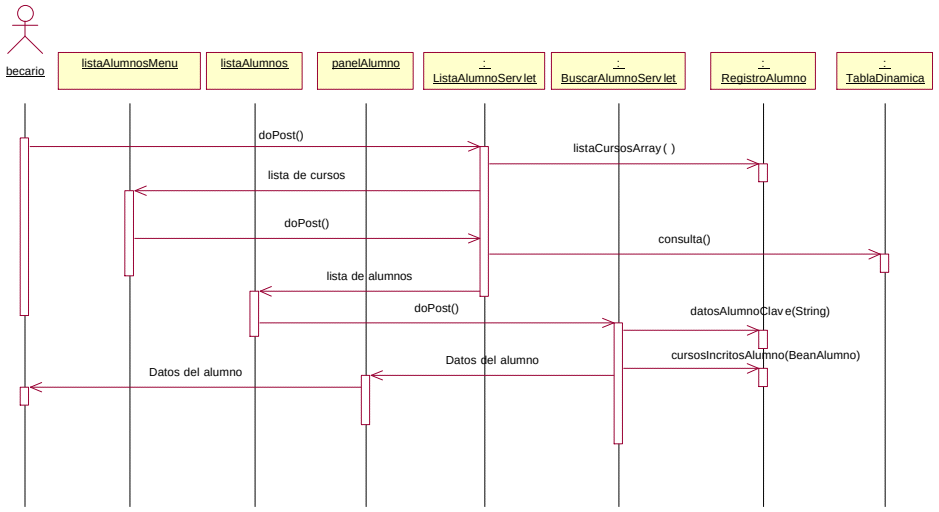
Validar becario.



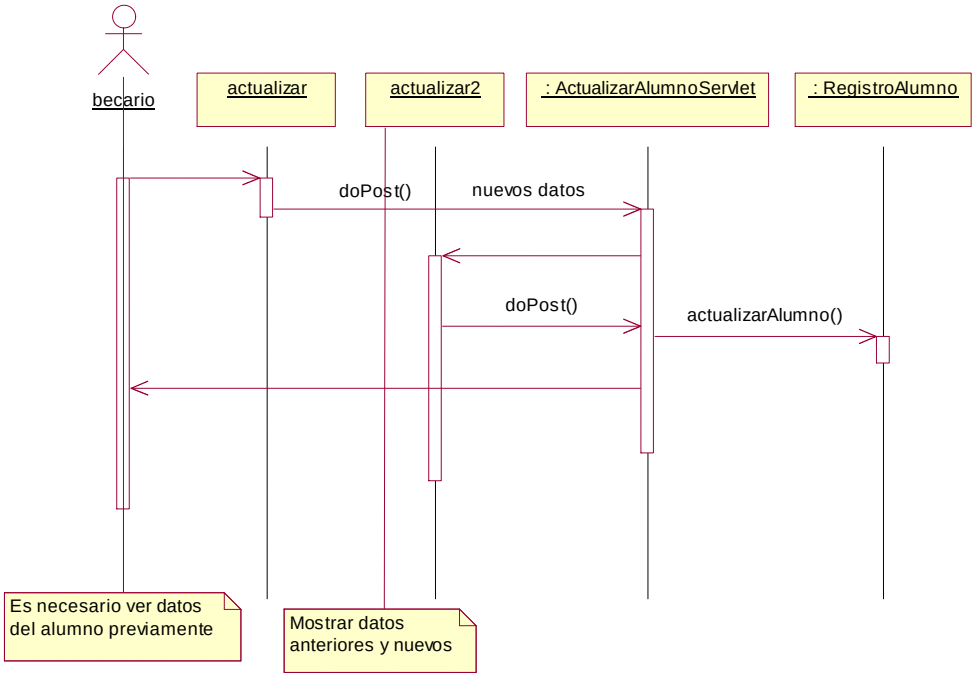
Consultar alumnos inscritos.



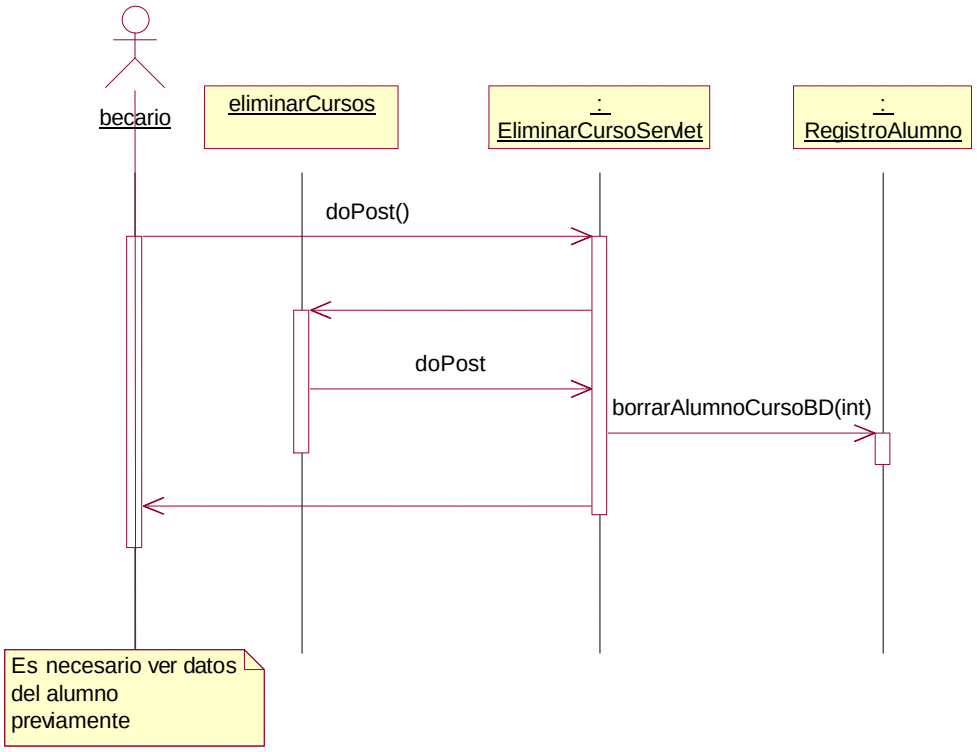
Ver datos del alumno.



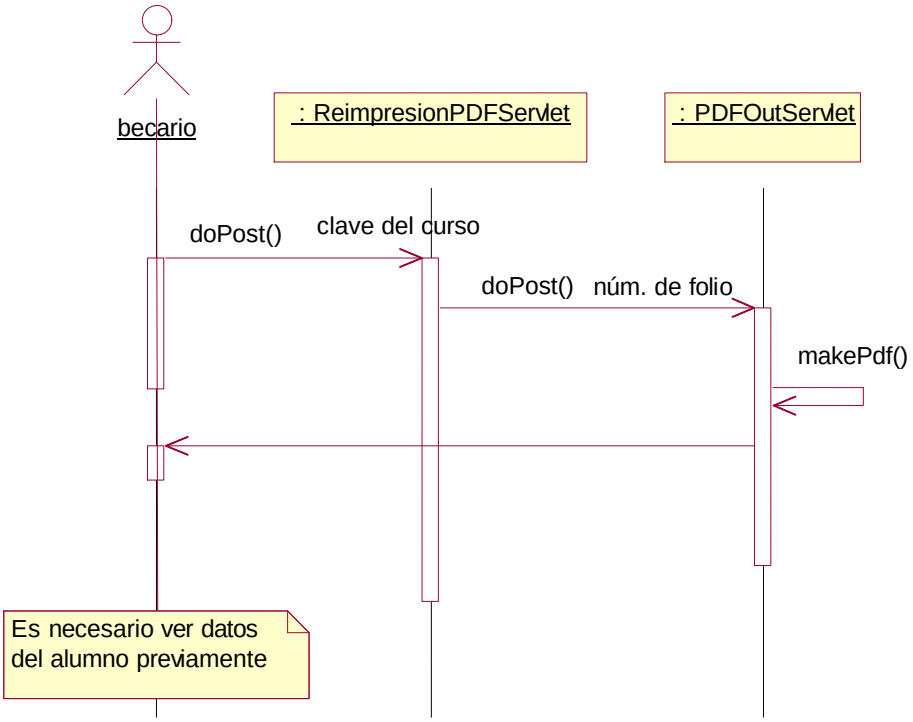
Actualizar datos de alumno.



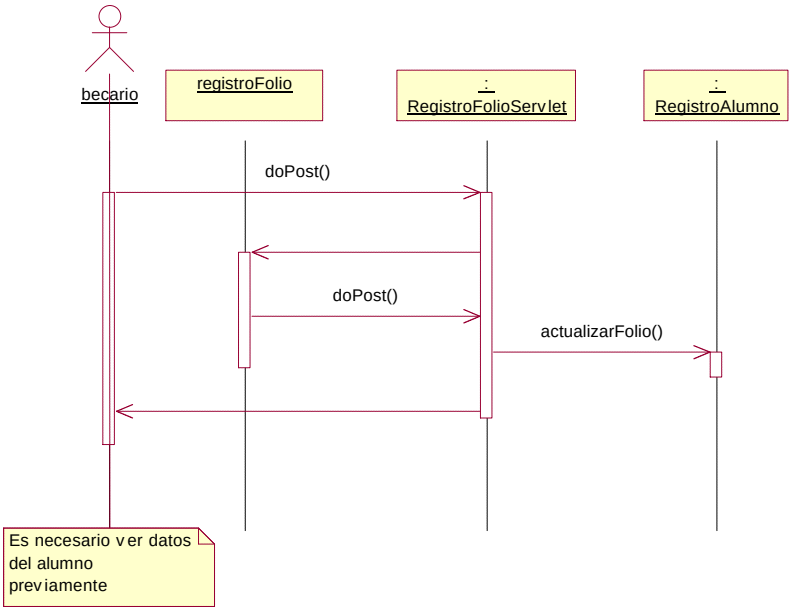
Eliminar alumno de un curso.



Imprimir ficha de pago.

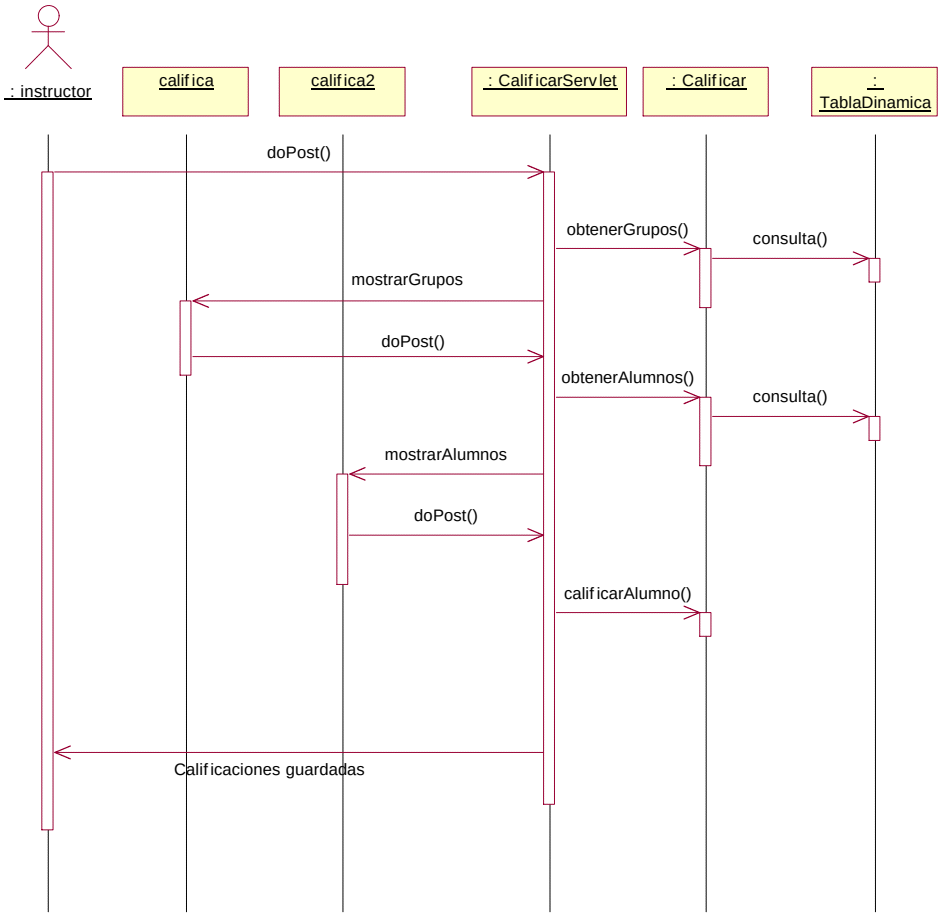


Registrar número de folio.

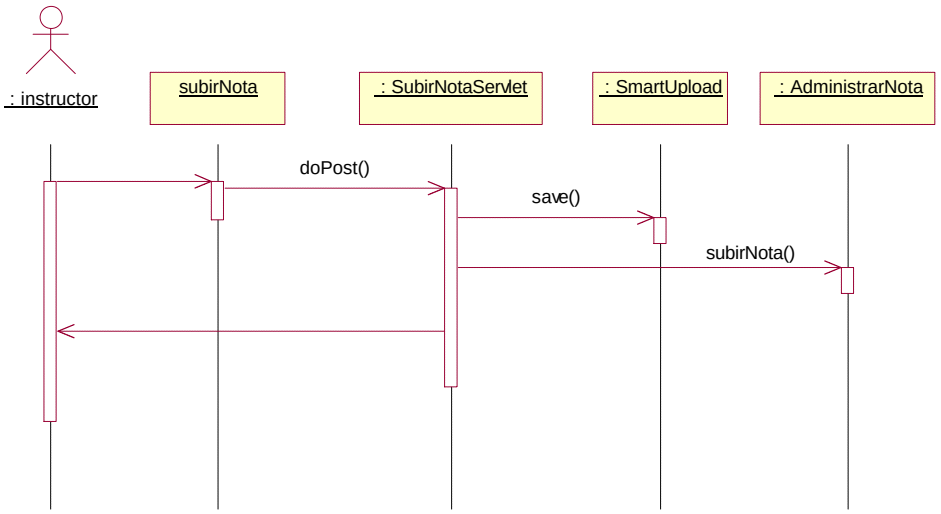


b) Diagramas de secuencia para los casos de uso del instructor.

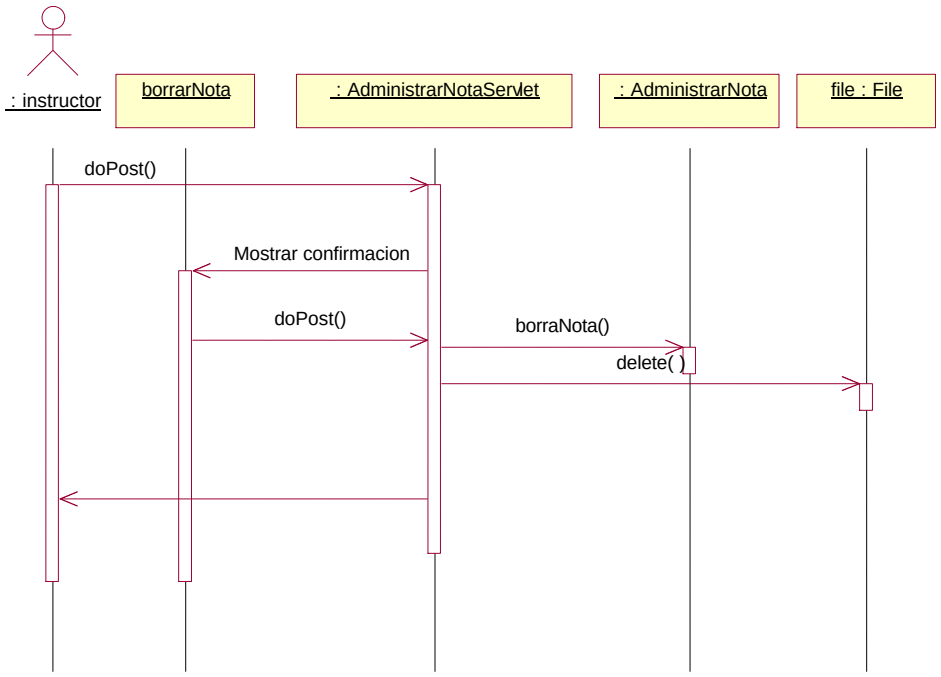
Califica alumnos del curso.



Publicar material.

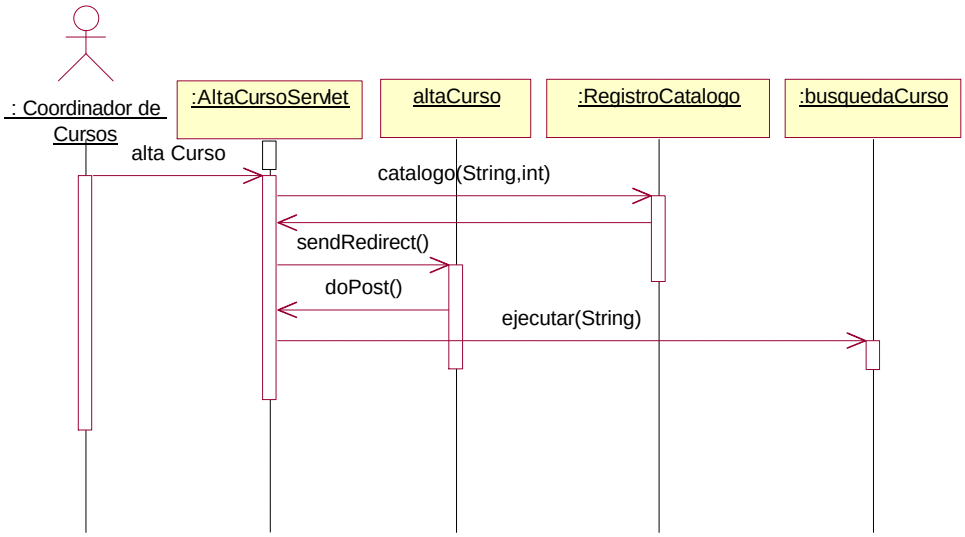


Eliminar material.

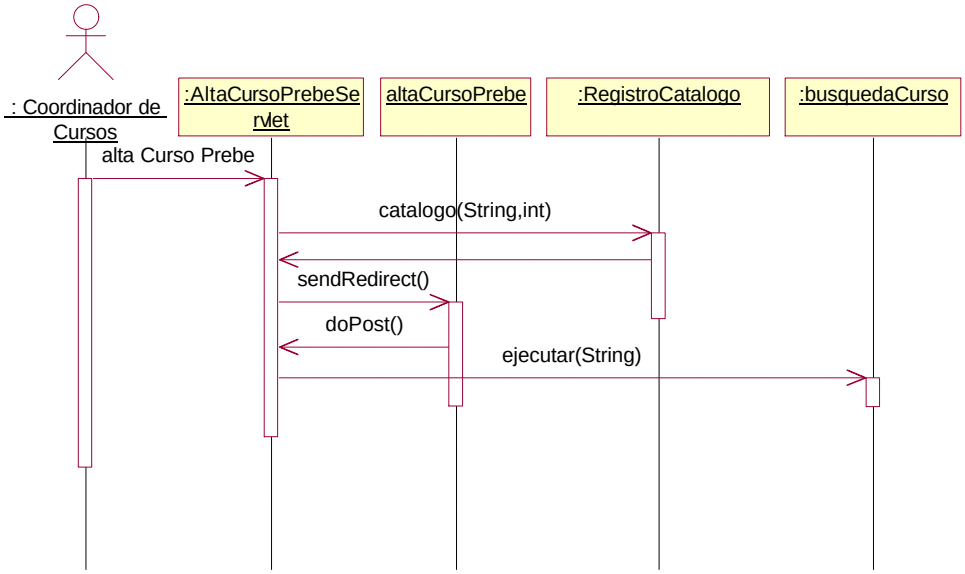


c) Diagramas de secuencia para los casos de uso de coordinador de curso

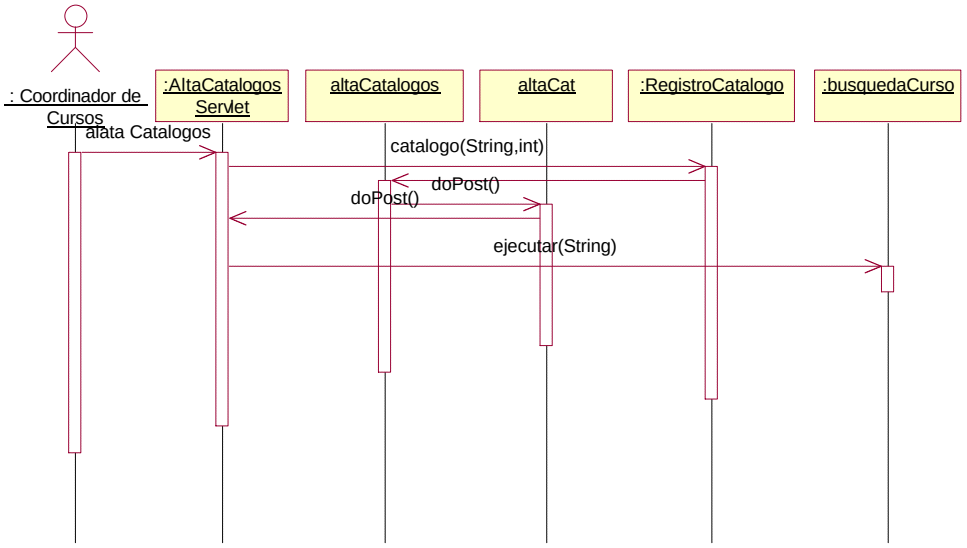
Alta curso



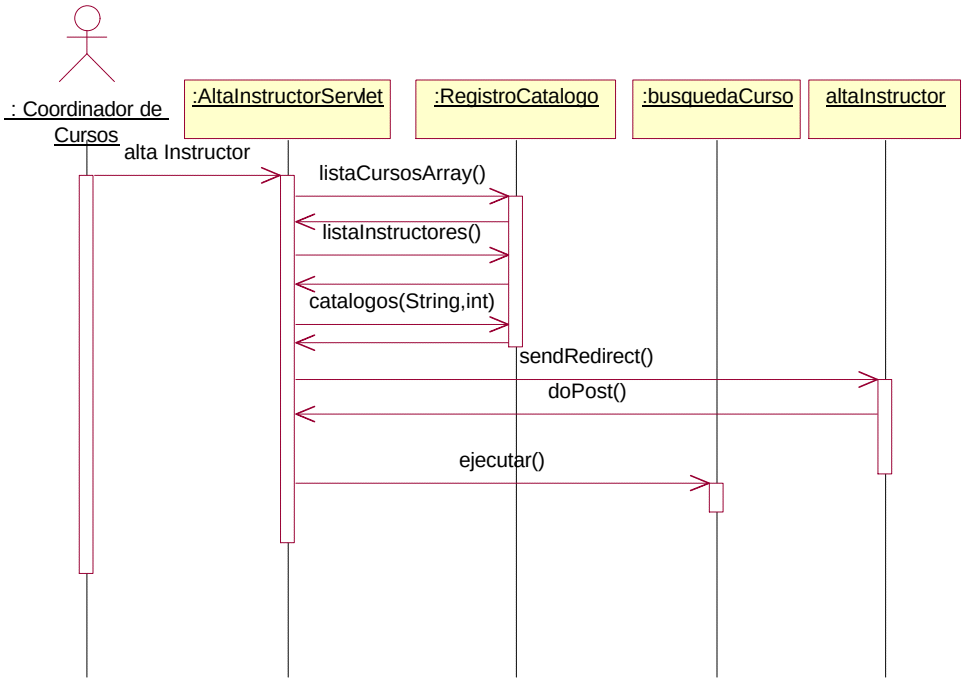
Alta curso prebecario.



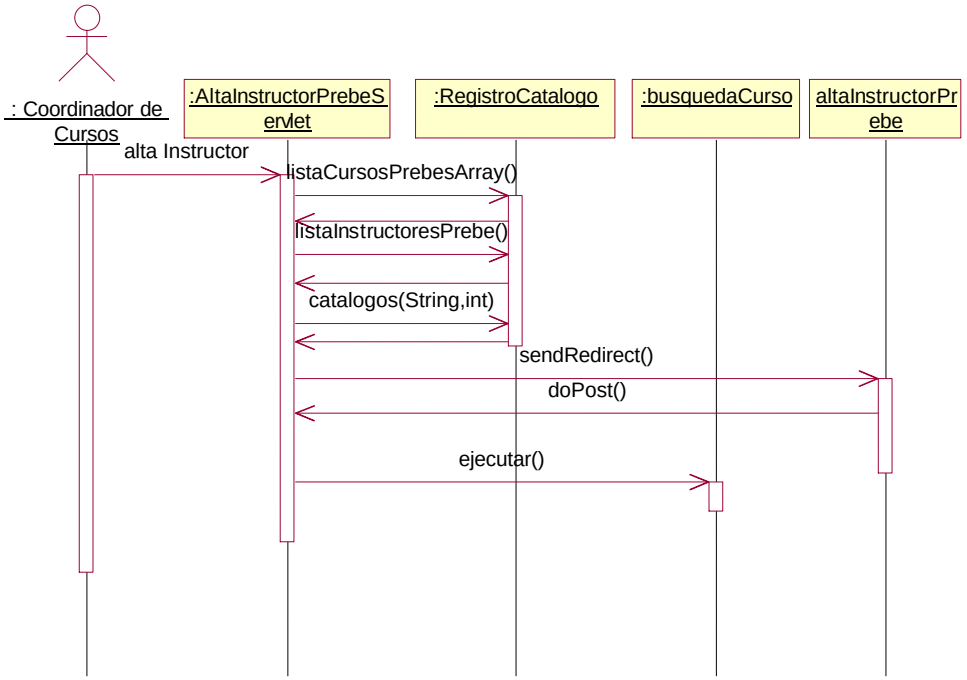
Alta elementos de catálogo.



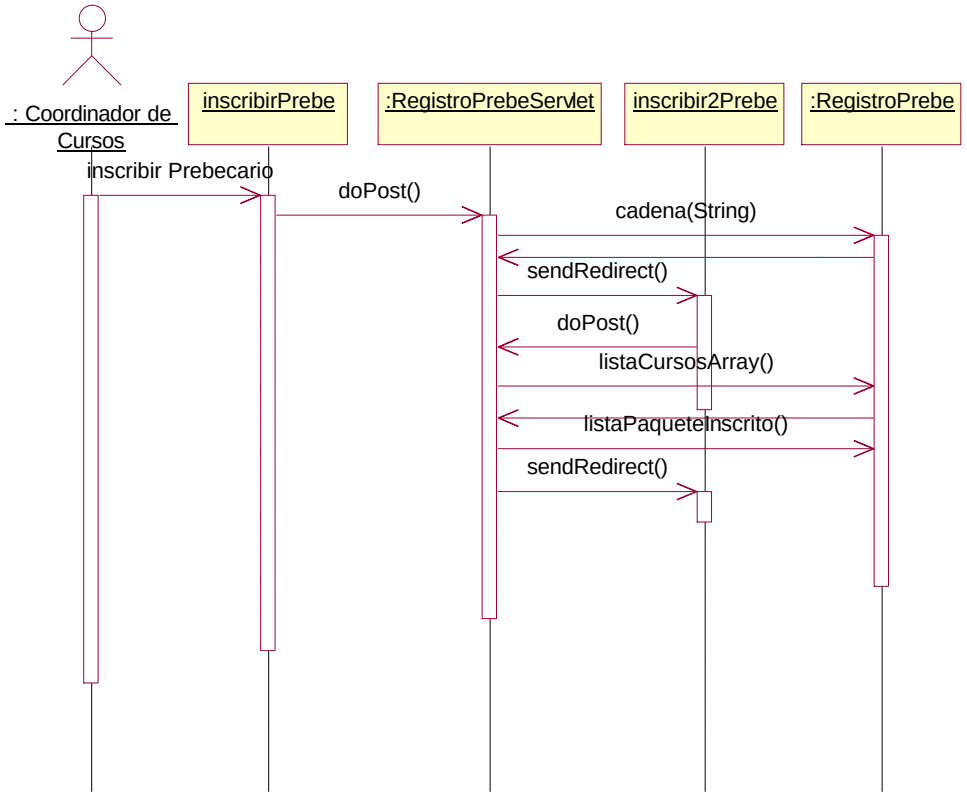
Alta de un instructor.



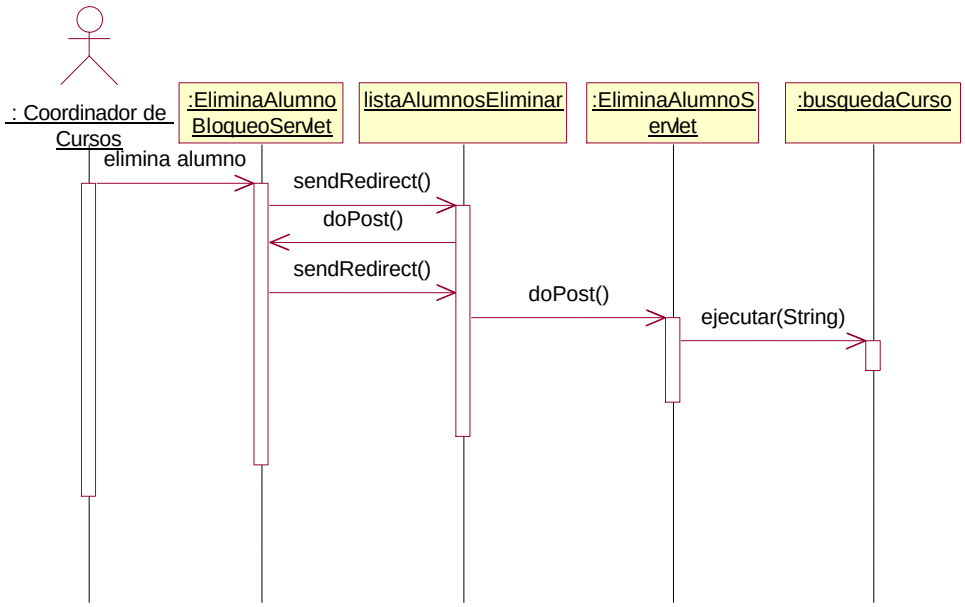
Alta instructor prebecario.



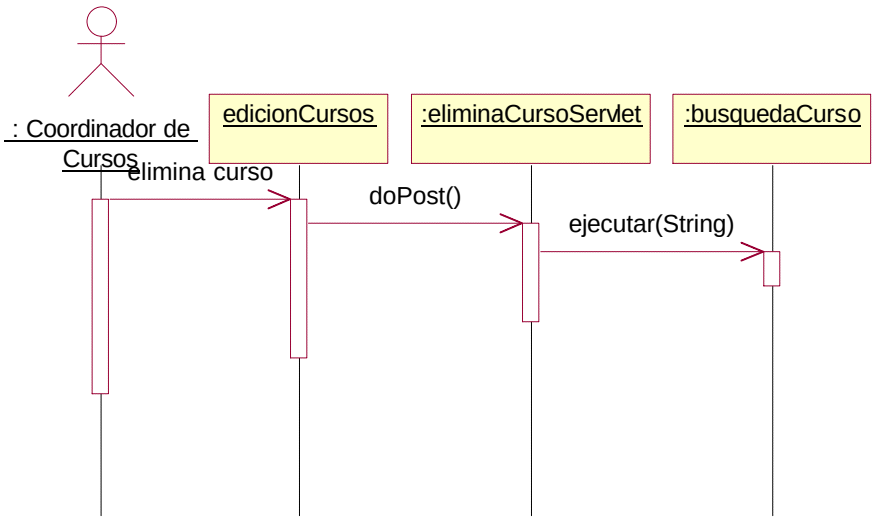
Alta prebecario.



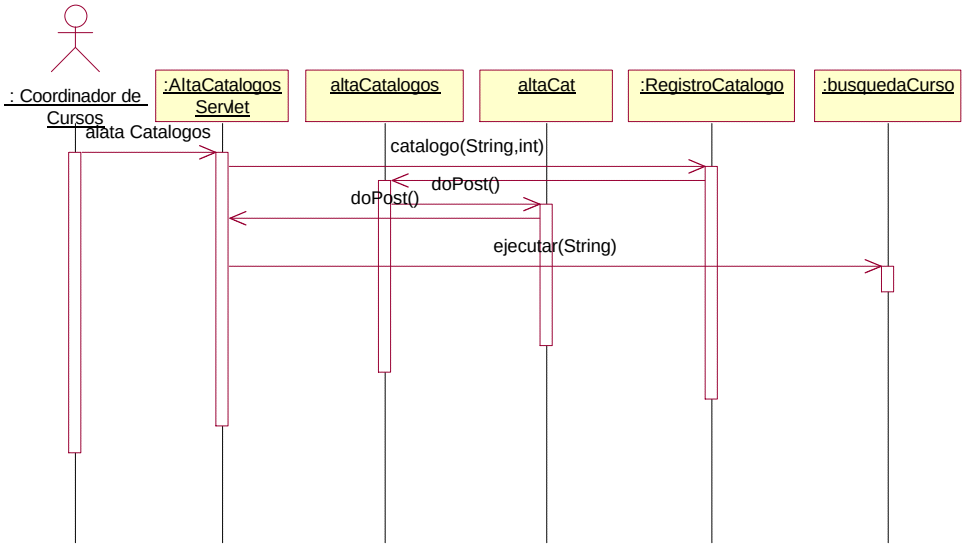
Baja alumnos.



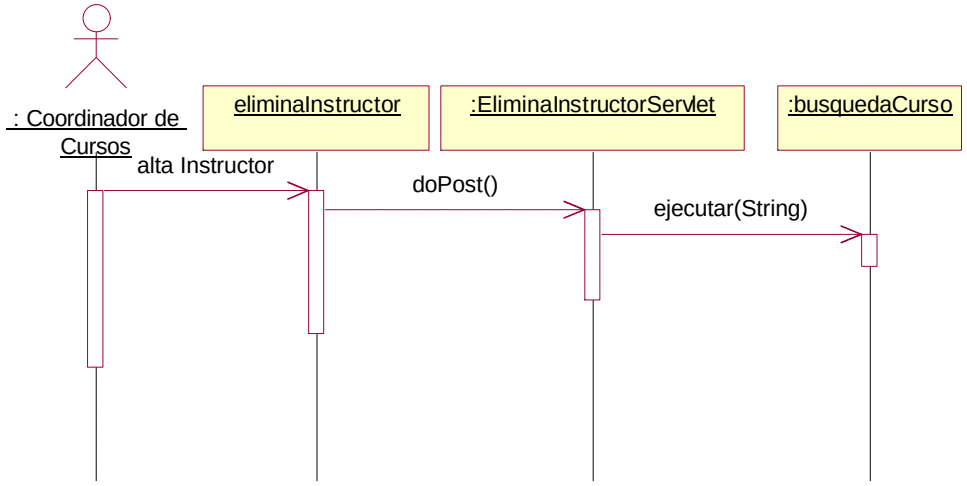
Baja curso.



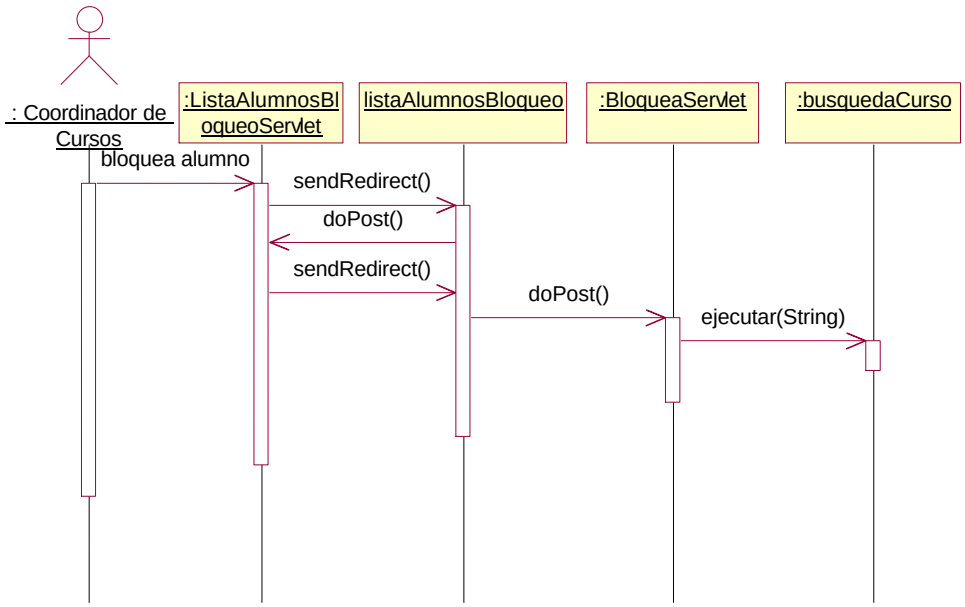
Baja de elementos de catálogo.



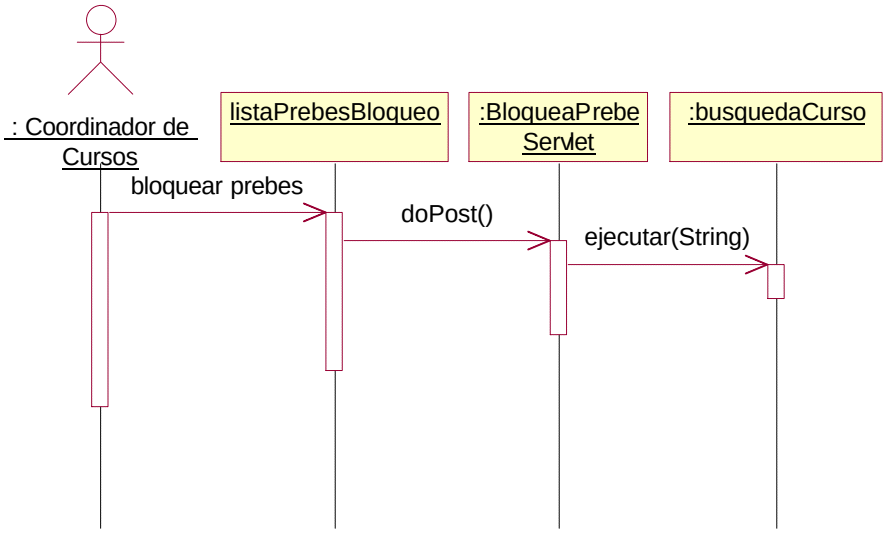
Baja de un instructor.



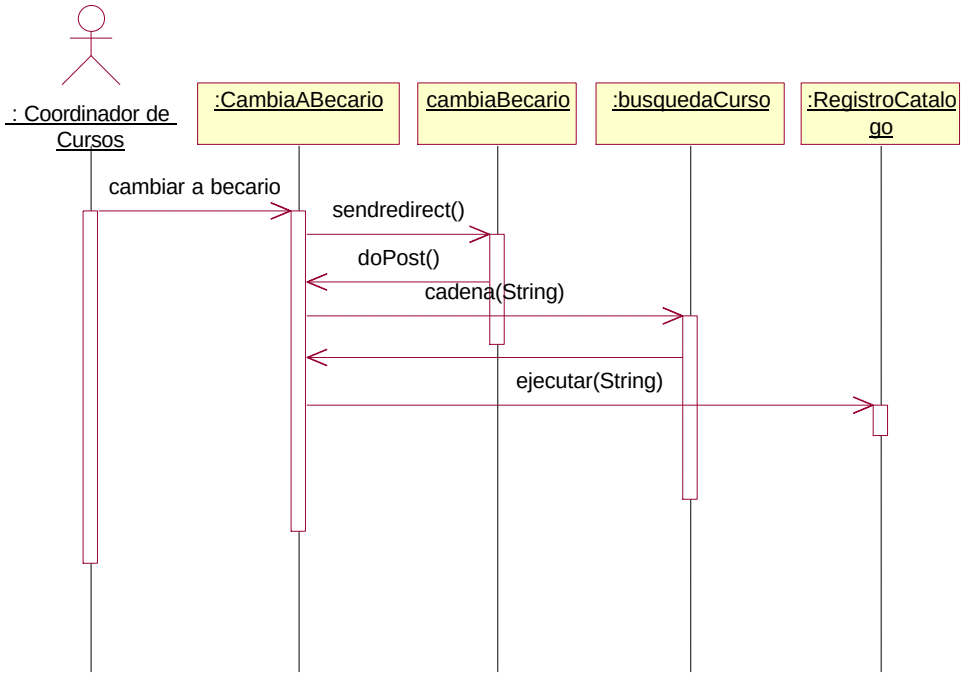
Bloquea alumnos.



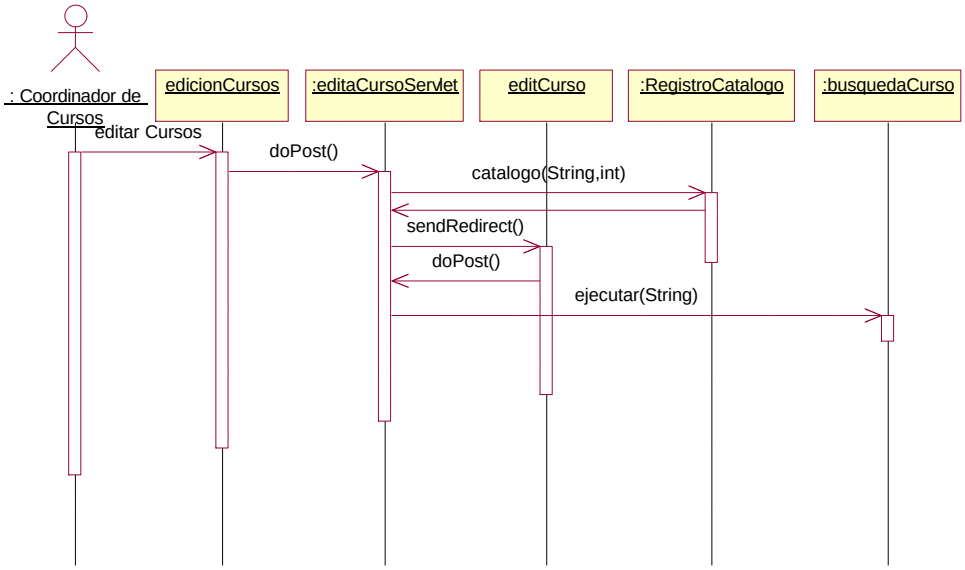
Bloquea prebecario.



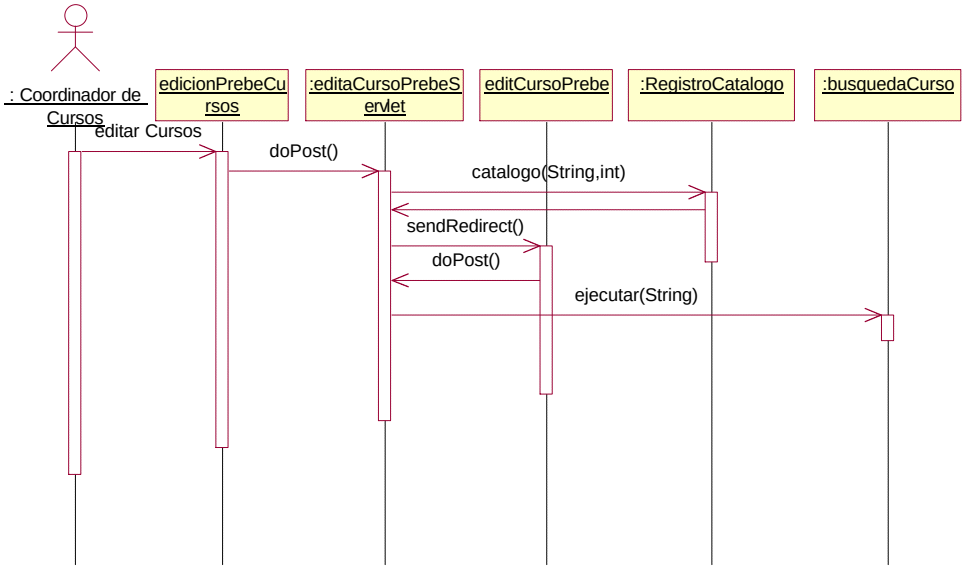
Convertir prebecario en becario.



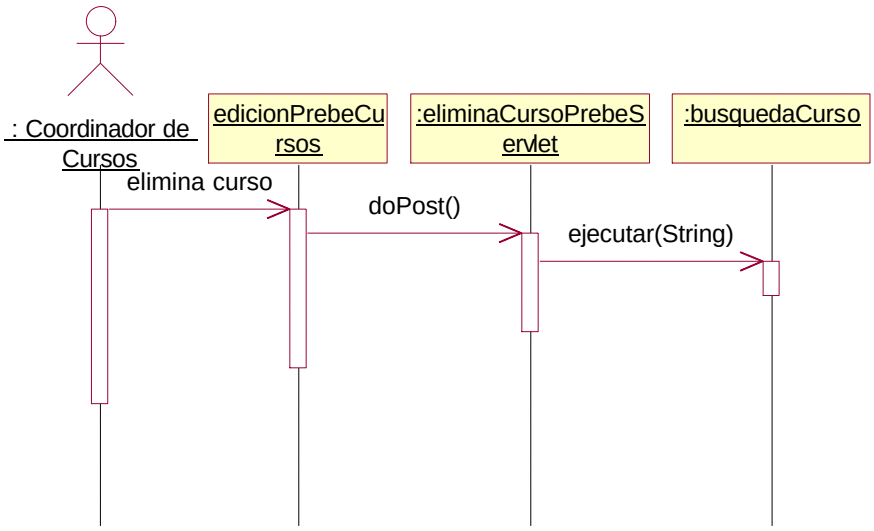
Edita curso.



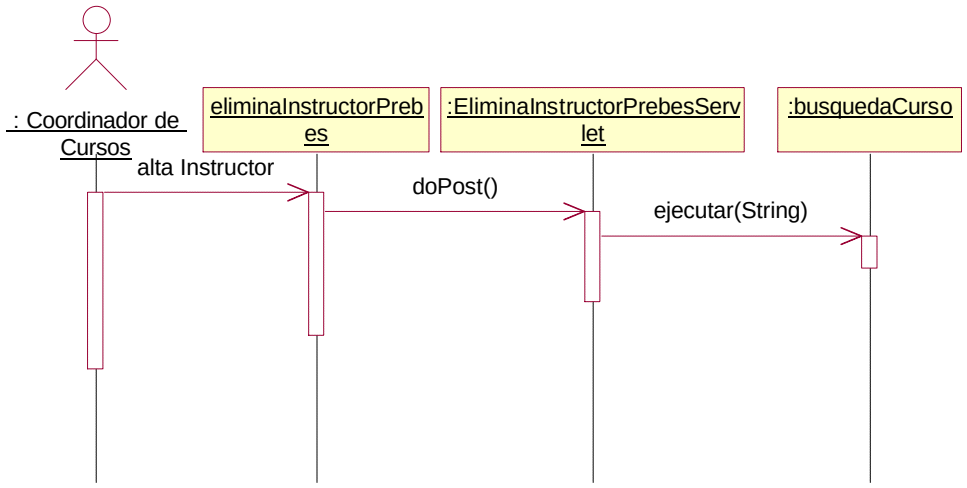
Edita curso prebecario.



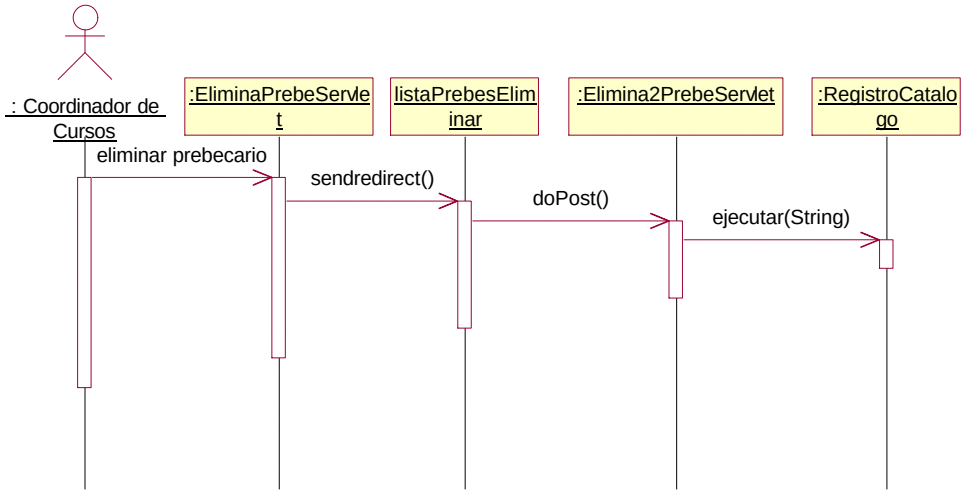
Elimina curso prebecario.



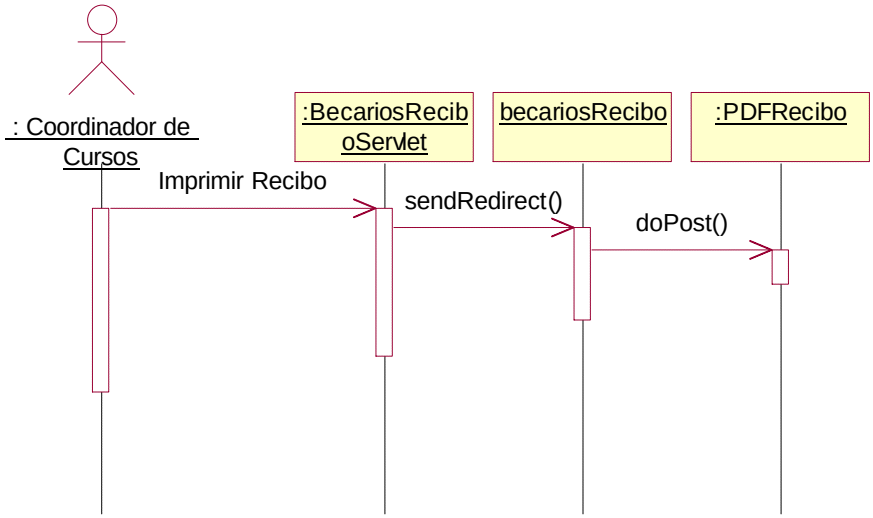
Elimina instructor prebecario.



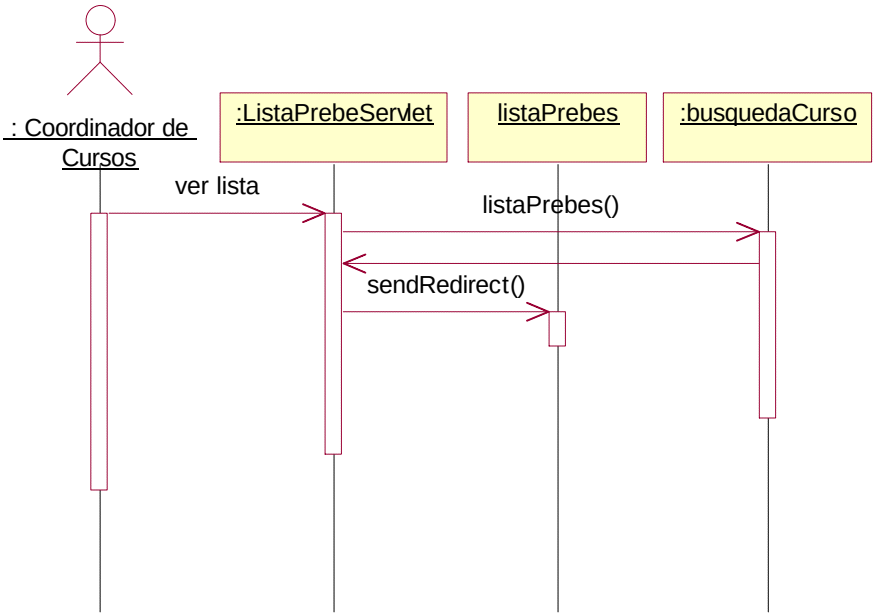
Elimina prebecario.



Imprimir recibos de becario.

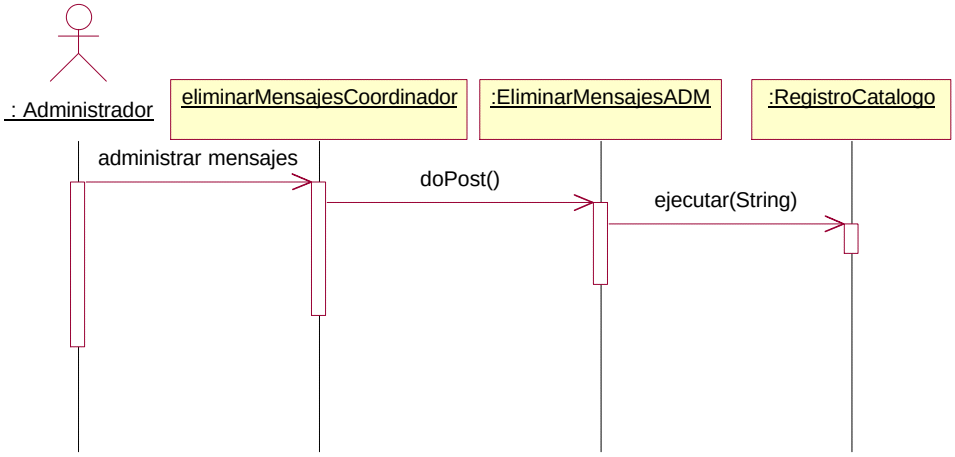


Ver lista prebecarios.

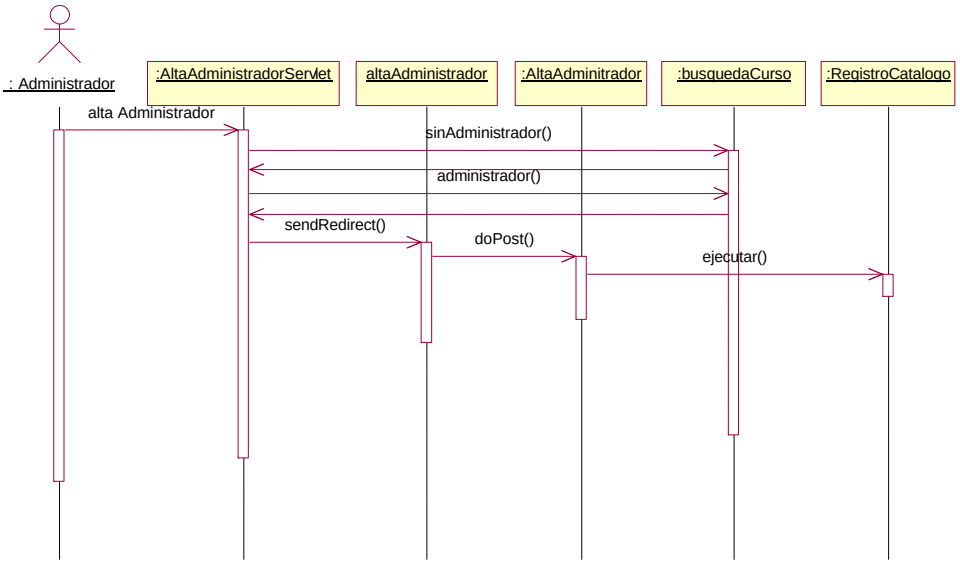


d) Diagramas de secuencia de los casos de uso de administrador.

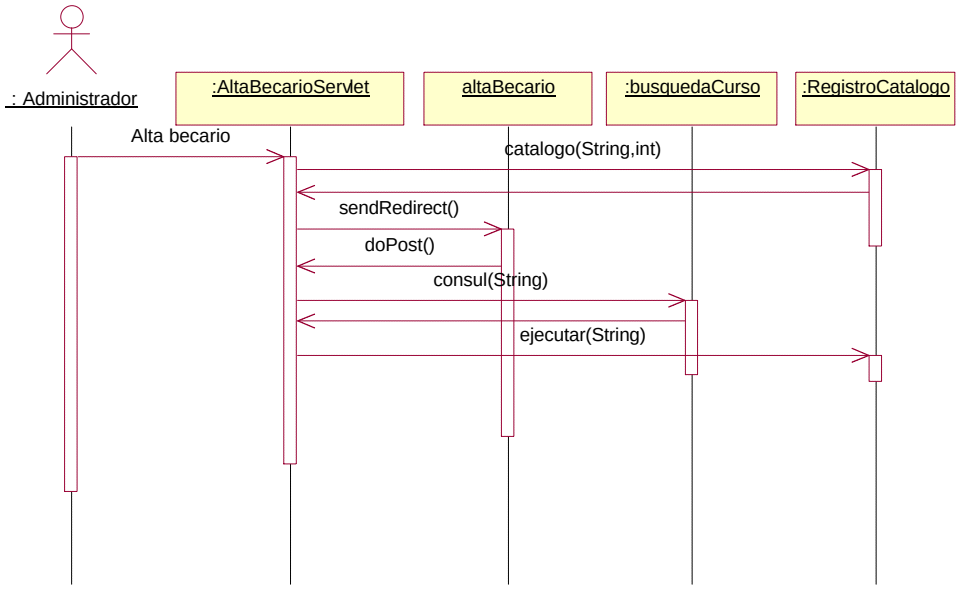
Administrar centro de mensajes.



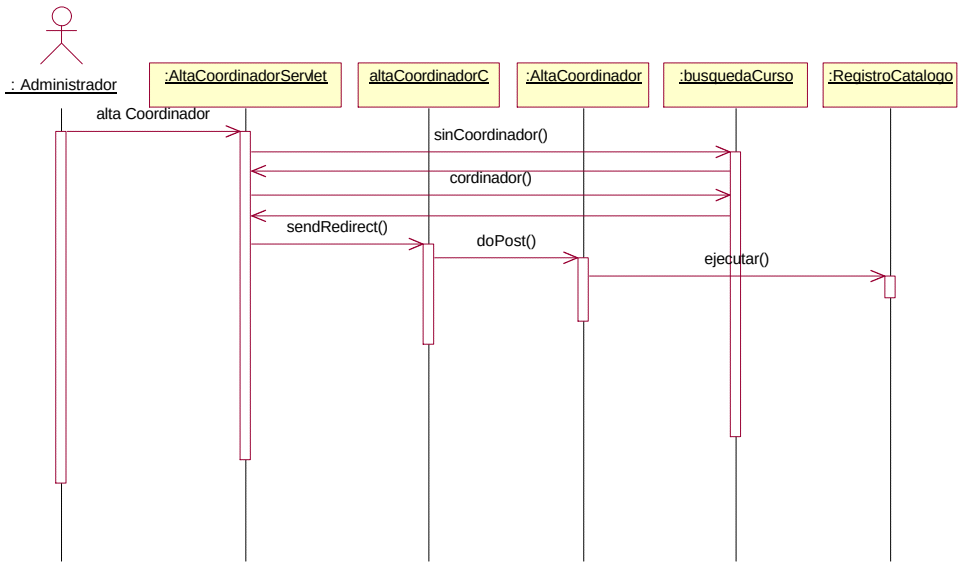
Alta administrador.



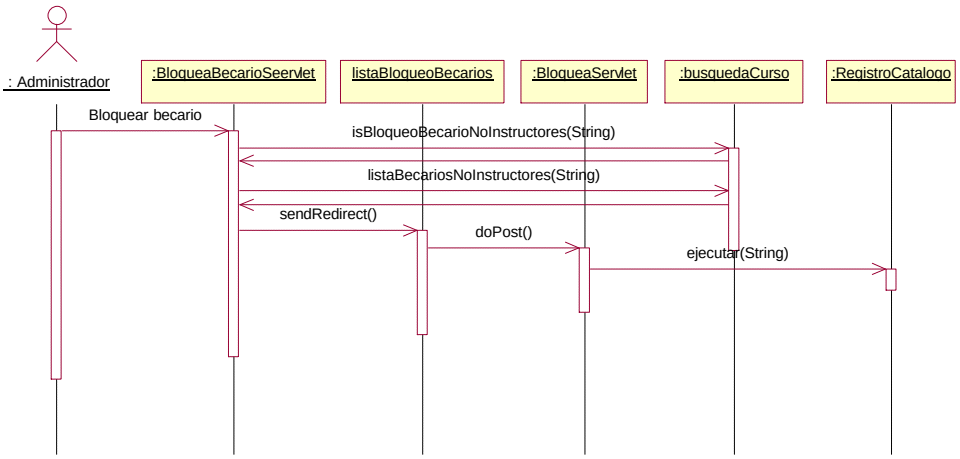
Alta de becario.



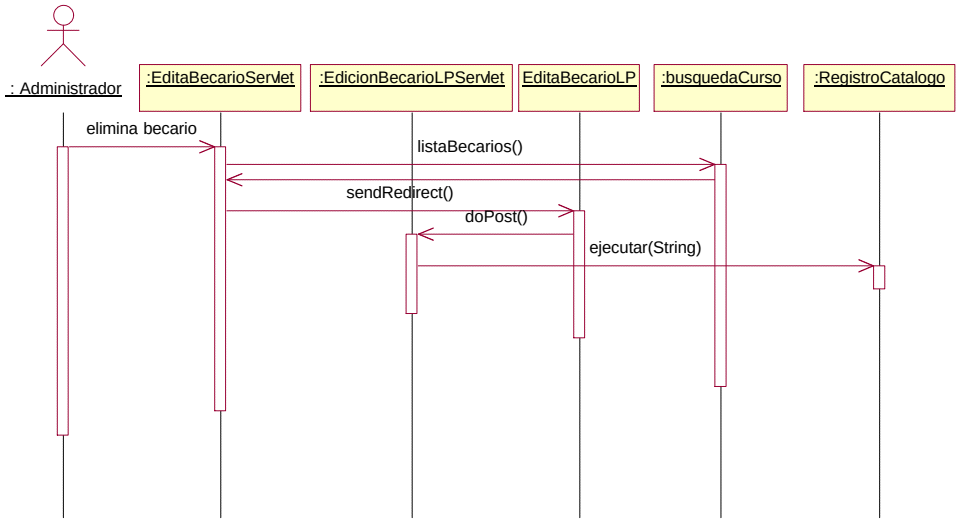
Alta del coordinador.



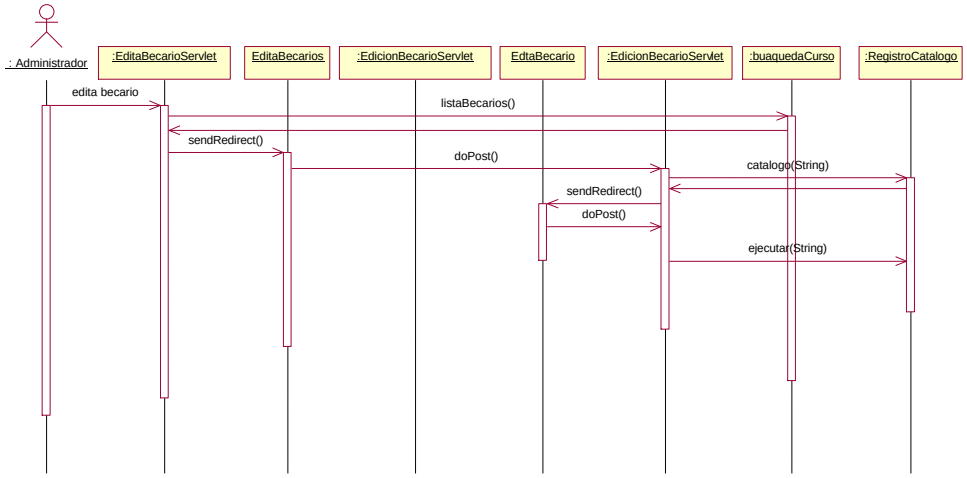
Bloquea de becario.



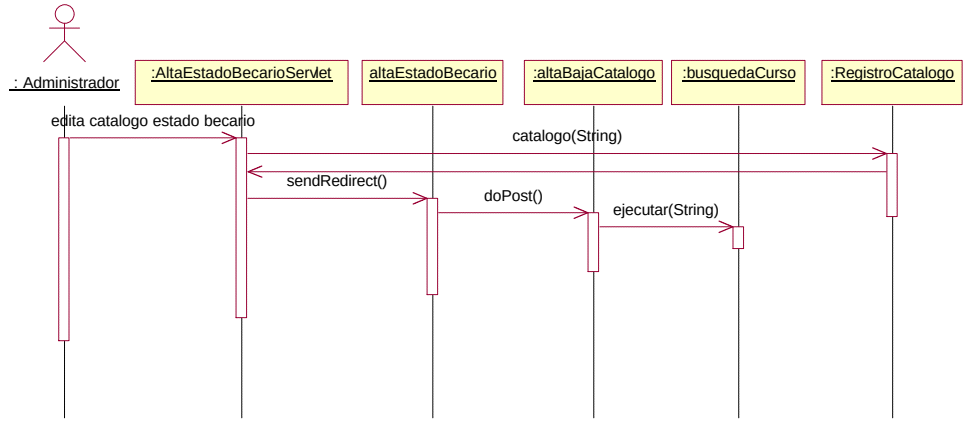
Cambio de contraseña del becario.



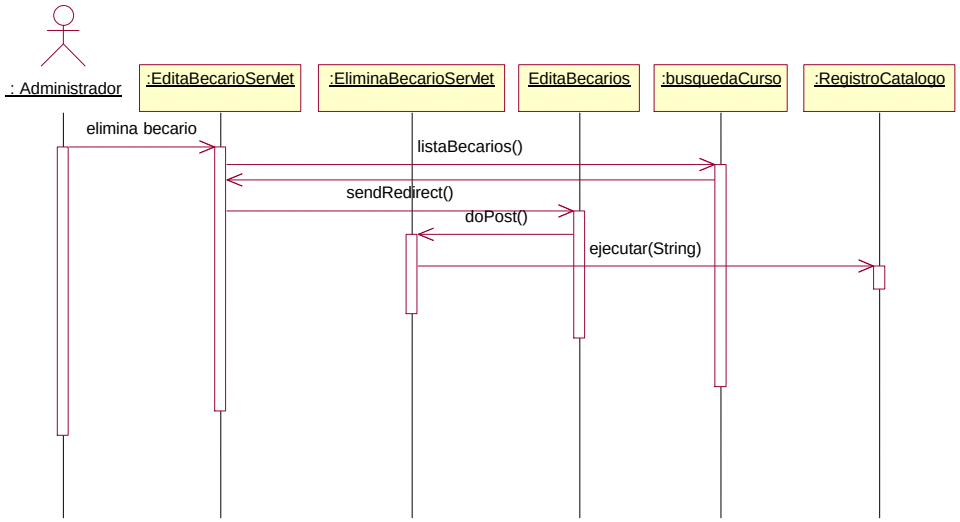
Editar becario.



Editar el catálogo estado becario.



Elimina becario.



4.1.2. Diagrama de actividades.

Diagrama de actividades del proceso de preinscripción.

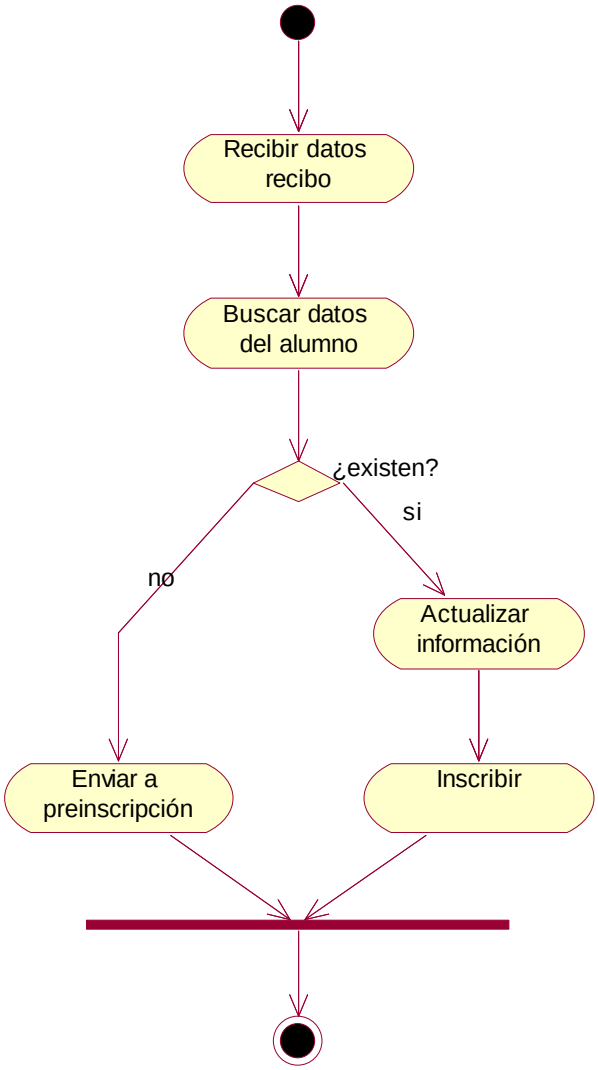
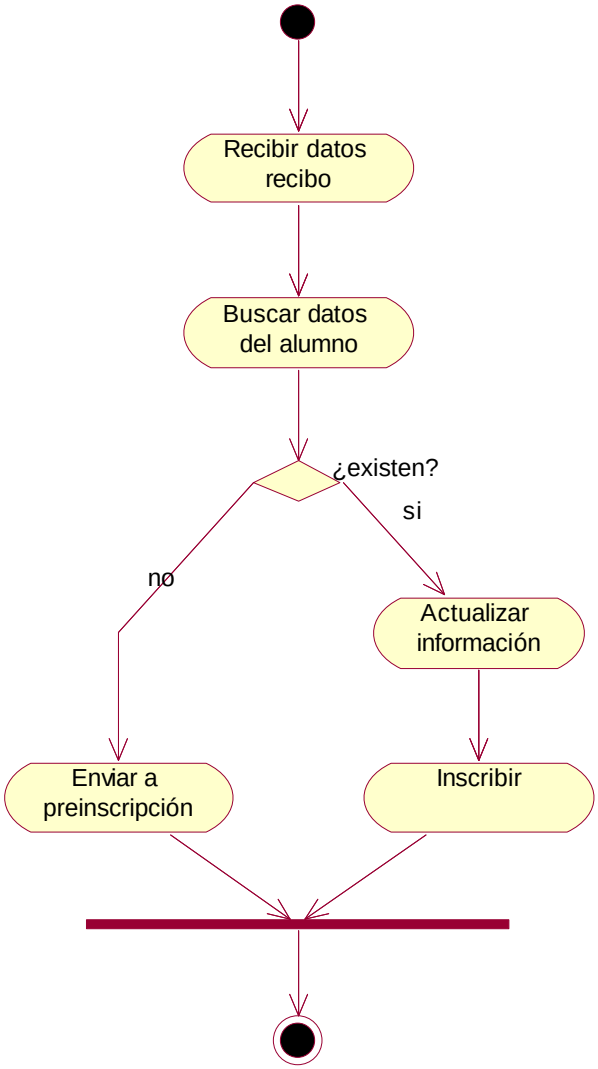
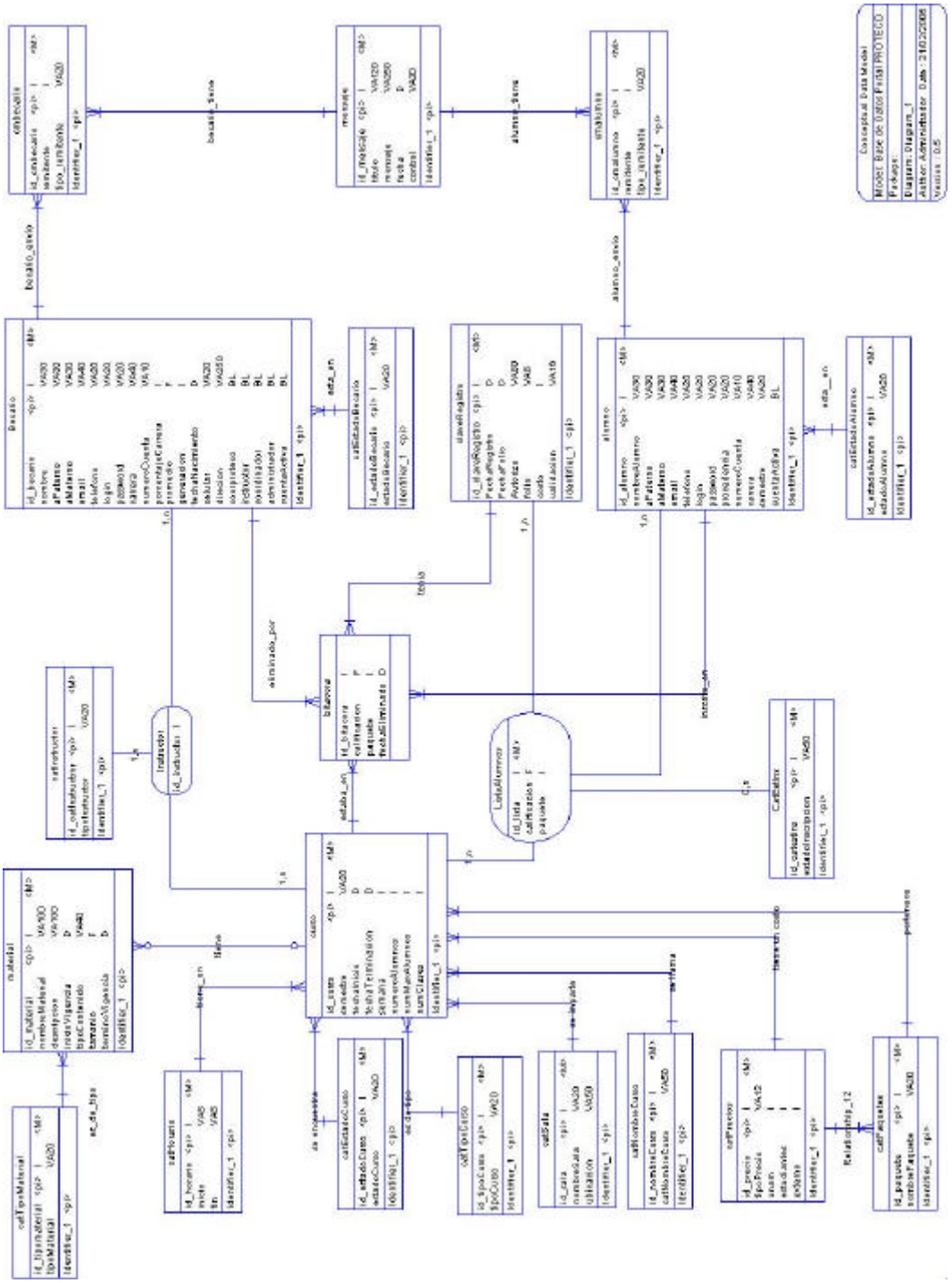


Diagrama de actividades del proceso de inscripción.

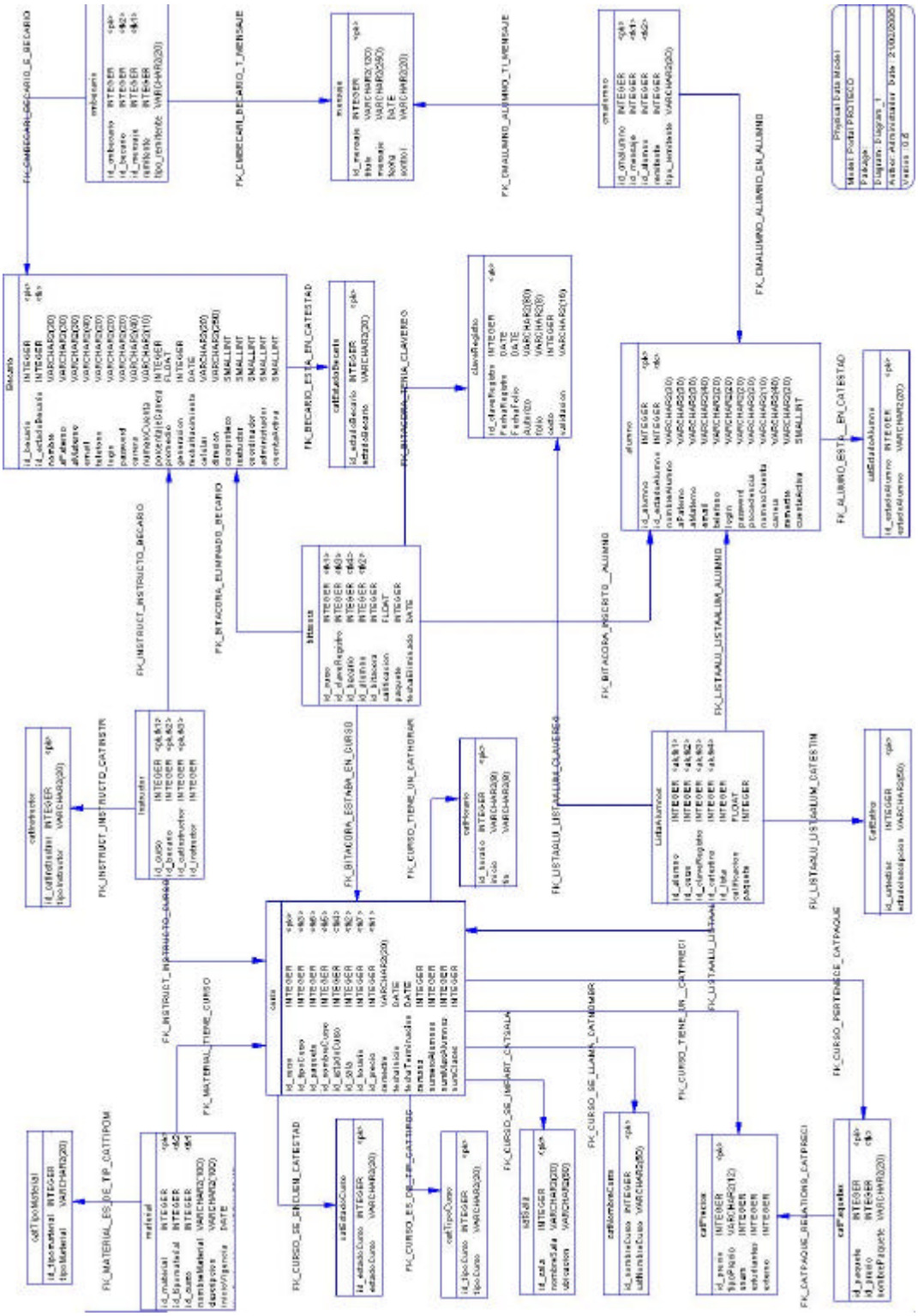


4.2. SQL

4.2.1. Diagrama conceptual.



4.2.2. Diagrama físico de la base de datos.



4.2.3. Diccionario de datos.

TABLA	**MENSAJE**		
CAMPO	***DESCRIPCION***	***TIPO DE DATO***	
ID_MENSAJE	Identificador.	INTEGER	PK
TITULO	Título del mensaje.	VARCHAR2(120)	NULL
MENSAJE	Texto completo del mensaje.	VARCHAR2(250)	NULL
FECHA	Fecha en que se envía el mensaje.	DATE	NULL
CONTROL	Bandera de control para envío masivo de mensajes.	VARCHAR2(20)	NULL
ID_MATERIAL	Identificador.	INTEGER	PK
ID_TIPOMATERIAL	Tipo de material, tiene relación con la tabla CATTIPOMATERIAL.	INTEGER	FK
ID_CURSO	Curso en el cual esta asignado el material, tiene relación con la tabla CURSO.	INTEGER	FK
TIPOCONTENIDO	Describe el formato del archivo.	VARCHAR2(40)	NULL
TERMINOVIGENCIA	Fecha final de distribución del documento.	DATE	NULL
TAMANIO	Tamaño en KBytes del archivo.	FLOAT	NULL
NOMBREMATERIAL	Nombre del archivo.	VARCHAR2(100)	NULL
INICIOVIGENCIA	Fecha inicial de distribución del documento.	DATE	NULL
DESCRIPCION	Descripción del documento.	VARCHAR2(100)	NULL

TABLA	**LISTAALUMNOS**		
CAMPO	***DESCRIPCION***	***TIPO DE DATO***	
ID_LISTA	Identificador.	INTEGER	PK
ID_CURSO	Curso en el que esta inscrito el alumno, tiene relación con la tabla CURSO.	INTEGER	PK
ID_CLAVEREGISTRO	Clave de registro, tiene relación con la tabla CLAVEREGISTRO.	INTEGER	PK
ID_ALUMNO	Alumno inscrito en el curso, tiene relación con la tabla ALUMNO.	INTEGER	PK
PAQUETE	Número del paquete inscrito o -1 en caso contrario.	INTEGER	NULL
CALIFICACION	Calificación final del alumno en el curso.	FLOAT	NULL

TABLA	**INSTRUCTOR**		
CAMPO	***DESCRIPCION***	***TIPO DE DATO***	
ID_INSTRUCTOR	Identificador.	INTEGER	PK
ID_CURSO	Curso en el que el becario esta asignado como instructor, tiene relación con la tabla CURSO.	INTEGER	PK
ID_CATINSTRUCTROR	Tipo de instructor ya sea Titular, Adjunto, etc. Tiene relación con la tabla CATINSTRUCTOR.	INTEGER	PK
ID_BECARIO	Becario asignado como instructor, tiene relación con la tabla BECARIO.	INTEGER	PK

TABLA	CURSO		
CAMPO	***DESCRIPCION***	***TIPO DE DATO***	
ID_CURSO	Identificador.	INTEGER	PK
ID_TIPOCURSO	Tipo de curso: sabatino, intersemestral, para prebecarios, etc. Tiene relación con la tabla CATTIPOCURSO.	INTEGER	FK
ID_SALA	Sala en la que se tiene planeado el curso, tiene relación con la tabla CATSALA.	INTEGER	FK
ID_PRECIO	Precio del curso, tiene relación con la tabla CATPRECIOS.	INTEGER	FK
ID_PAQUETE	Nombre del grupo de cursos al que esta asociado y se denomina paquete. Tiene relación con la tabla CATPAQUETES.	INTEGER	FK
ID_NOMBRECURSO	Nombre del curso, tiene relación con la tabla CATNOMBRECURSO.	INTEGER	FK
ID_HORARIO	Horario del curso, tiene relación con la tabla CATHORARIO.	INTEGER	FK
ID_ESTADOCURSO	Estado en el que se encuentra el curso: inscripción, cancelado, terminado, etc. Tiene relación con la tabla CATESTADOCURSO.	INTEGER	FK
SEMESTRE	Semestre en el que se imparte el curso.	VARCHAR2(20)	NULL
SEMANA	Número de semana en el que esta programado para el caso de estar en un paquete.	INTEGER	NULL
NUMMAXALUMNOS	Número máximo de alumnos en el curso.	INTEGER	NULL
NUMEROALUMNOS	Número de alumnos inscritos en el curso.	INTEGER	NULL
NUMCLASES	Número de clases del curso.	INTEGER	NULL
FECHATERMINACION	Fecha de terminación del curso.	DATE	NULL
FECHAINICIO	Fecha de inicio del curso.	DATE	NULL

TABLA	CMBECARIO		
CAMPO	***DESCRIPCION***	***TIPO DE DATO***	
ID_CMBECARIO	Identificador.	INTEGER	PK
ID_MENSAJE	Identificador tabla mensaje.	INTEGER	FK
ID_BECARIO	Identificador destinatario (becario).	INTEGER	FK
TIPO_REMITENTE	Tipo de remitente (alumno o becario).	VARCHAR2(20)	NULL
REMITENTE	Identificador del remitente del mensaje.	INTEGER	NULL
ID_CMALUMNO	Identificador.	INTEGER	PK
ID_MENSAJE	Identificador tabla mensaje.	INTEGER	FK
ID_ALUMNO	Identificador destinatario (alumno).	INTEGER	FK

TABLA	CLAVEREGISTRO

CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_CLAVEREGISTRO	Identificador.	INTEGER	PK
FOLIO	Número de folio del comprobante de pago.	VARCHAR2(8)	NULL
FECHAREGISTRO	Fecha de registro al curso.	DATE	NULL
FECHAFOLIO	Fecha de registro del folio.	DATE	NULL
COSTO	Costo que aparece en el comprobante de pago.	INTEGER	NULL
AUTORIZO	Nombre de la persona que autorizo la beca.	VARCHAR2(80)	NULL

TABLA	**CATTIPOCURSO**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_TIPOMATERIAL	Identificador.	INTEGER	PK
TIPOMATERIAL	Tipo de material: manual, ejercicio, examen, etc.	VARCHAR2(20)	NULL
ID_TIPOCURSO	Identificador.	INTEGER	PK
TIPOCURSO	Tipo de curso :sabatino, intersemestral, para prebecarios, etc.	VARCHAR2(20)	NULL

TABLA	**CATSALA**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_SALA	Identificador.	INTEGER	PK
UBICACION	Descripción de la localización física de la sala.	VARCHAR2(50)	NULL
NOMBRESALA	Nombre de la sala.	VARCHAR2(20)	NULL

TABLA	**CATPRECIOS**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_PRECIO	Identificador.	INTEGER	PK
UNAM	Costo para alumnos de la UNAM.	INTEGER	NULL
TIPOPRECIO	Tipo de precio para el curso, es decir, la categoría en la que se encuentra A, B, etc.	VARCHAR2(12)	NULL
EXTERNO	Costo para personas que no son alumnos.	INTEGER	NULL
ESTUDIANTES	Costo para alumnos de otra institución.	INTEGER	NULL

TABLA	**CATPAQUETES**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_PAQUETE	Identificador.	INTEGER	PK
ID_PRECIO	Precio del paquete.	INTEGER	FK
NOMBREPAQUETE	Nombre del conjunto de curso denominado paquete.	VARCHAR2(20)	NULL

TABLA	**CATNOMBRECURSO**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_NOMBRECURSO	Identificador.	INTEGER	PK
CATNOMBRECURSO	Nombre del curso.	VARCHAR2(50)	NULL

TABLA	**CATINSTRUCTOR**

CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_CATINSTRUCTROR	Identificador.	INTEGER	PK
TIPOINSTRUCTOR	Tipo de instructor: Titular, Adjunto, etc.	VARCHAR2(20)	NULL

TABLA	**CATHORARIO**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_HORARIO	Identificador.	INTEGER	PK
INICIO	Hora de inicio del curso.	VARCHAR2(8)	NULL
FIN	Hora de terminación del curso.	VARCHAR2(8)	NULL

TABLA	**CATESTADOCURSO**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_ESTADOCURSO	Identificador.	INTEGER	PK
ESTADOCURSO	Estado en el que se encuentra el curso: programado, inscripción, terminado, cancelado, etc.	VARCHAR2(20)	NULL

TABLA	**CATESTADOBECARIO**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_ESTADOBECARIO	Identificador.	INTEGER	PK
ESTADOBECARIO	Estado en el que se encuentra el becario: becario, ex becario, etc.	VARCHAR2(20)	NULL
ID_ESTADOALUMNO	Identificador.	INTEGER	PK
ESTADOALUMNO	Estado en el que se encuentra el alumno: preinscripción, inscripción, etc.	VARCHAR2(20)	NULL

TABLA	**BECARIO**		
CAMPO	*DESCRIPCION*	*TIPO DE DATO*	
ID_BECARIO	Identificador.	INTEGER	PK
ID_ESTADOBECARIO	Estado en el que se encuentra el becario. Tiene relación con la tabla CATESTADOBECARIO.	INTEGER	FK
TELEFONO	Número telefónico del becario.	VARCHAR2(20)	NULL
PROMEDIO	Calificación promedio del becario.	FLOAT	NULL
PORCENTAJECARRERA	Porcentaje cursado de la carrera.	INTEGER	NULL
PASSWORD	Contraseña del becario.	VARCHAR2(20)	NULL
NUMEROCUENTA	Número de cuenta del becario.	VARCHAR2(10)	NULL
NOMBRE	Nombre del becario.	VARCHAR2(30)	NULL
LOGIN	Nombre de inicio de sesión del becario.	VARCHAR2(20)	NULL
INSTRUCTOR	Bandera de identificación como instructor.	SMALLINT	NULL
GENERACION	Número de generación del programa de becarios.	INTEGER	NULL
FECHANACIMIENTO	Fecha de nacimiento del becario.	DATE	NULL
EMAIL	Dirección de correo electrónica.	VARCHAR2(40)	NULL
DIRECCION	Dirección del becario.	VARCHAR2(250)	NULL
CUENTAACTIVA	Bandera para activar o des activar el acceso al sitio.	SMALLINT	NULL
COORPROTECO	Bandera de identificación como coordinador PROTECO.	SMALLINT	NULL
COORDINADOR	Bandera de identificación como coordinador de cursos.	SMALLINT	NULL

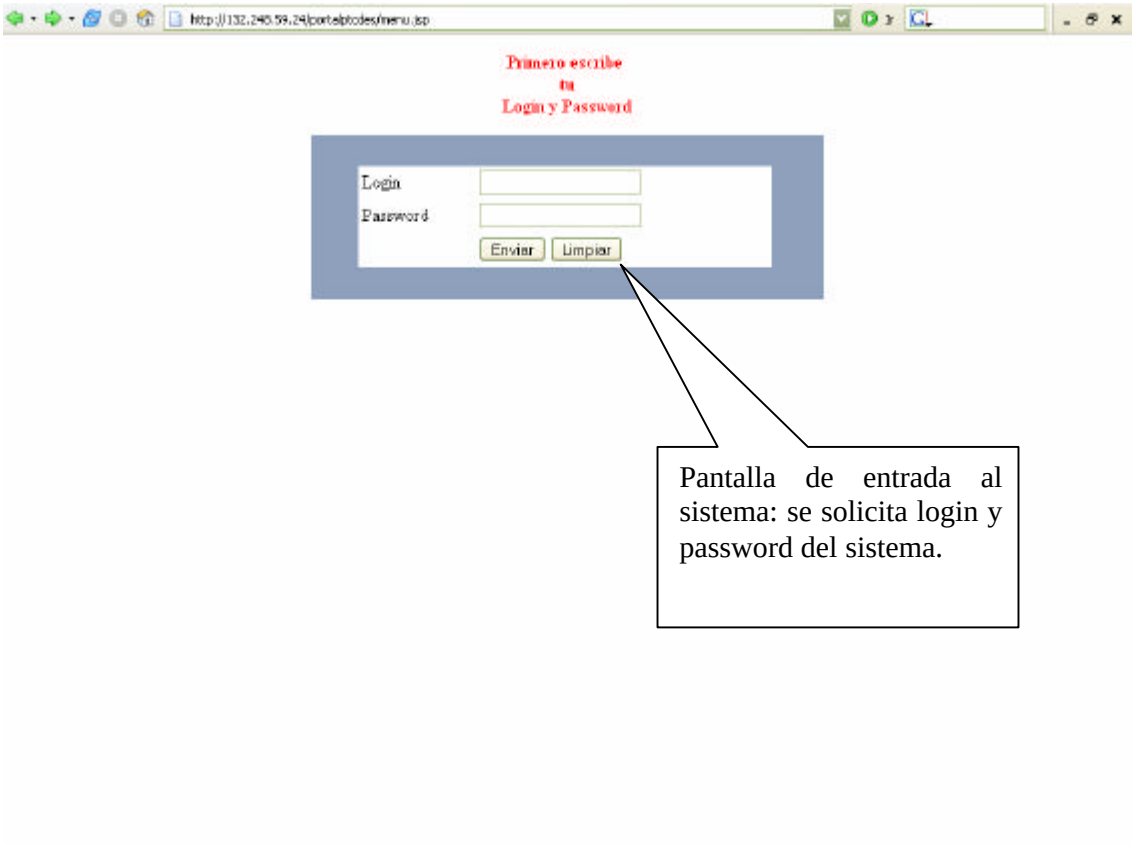
CELULAR	Número telefónico del celular del becario.	VARCHAR2(20)	NULL
CARRERA	Nombre de la carrera a la que pertenece el becario.	VARCHAR2(40)	NULL
APATERNO	Apellido paterno del becario.	VARCHAR2(30)	NULL
AMATERNO	Apellido materno del becario.	VARCHAR2(30)	NULL
ADMINISTRADOR	Bandera de identificación como administrador.	SMALLINT	NULL

TABLA	**ALUMNO**		
CAMPO	***DESCRIPCION***	***TIPO DE DATO***	
ID_ALUMNO	Identificador.	INTEGER	PK
ID_ESTADOALUMNO	Estado en el que se encuentra el alumno. Tiene relación con la tabla CATESTADOALUMNO.	INTEGER	FK
TELEFONO	Número telefónico del alumno.	VARCHAR2(20)	NULL
SEMESTRE	Semestre en el que esta inscrito el alumno en caso de ser alumno de la UNAM u otra institución.	VARCHAR2(20)	NULL
PROCEDENCIA	Institución de procedencia en el caso de ser un alumno y en caso contrario se considera la procedencia general.	VARCHAR2(20)	NULL
PASSWORD	Contraseña del alumno.	VARCHAR2(20)	NULL
NUMEROCUENTA	Número de cuenta del alumno.	VARCHAR2(10)	NULL
NOMBREALUMNO	Nombre del alumno.	VARCHAR2(30)	NULL
LOGIN	Nombre de inicio de sesión del alumno.	VARCHAR2(20)	NULL
EMAIL	Dirección electrónica del alumno.	VARCHAR2(40)	NULL
CUENTAACTIVA	Bandera para activar o desactivar el inicio de sesión del alumno.	SMALLINT	NULL
CARRERA	Nombre de la carrera a la que pertenece el alumno.	VARCHAR2(40)	NULL
APATERNO	Apellido paterno del alumno.	VARCHAR2(30)	NULL
AMATERNO	Apellido materno del alumno.	VARCHAR2(30)	NULL

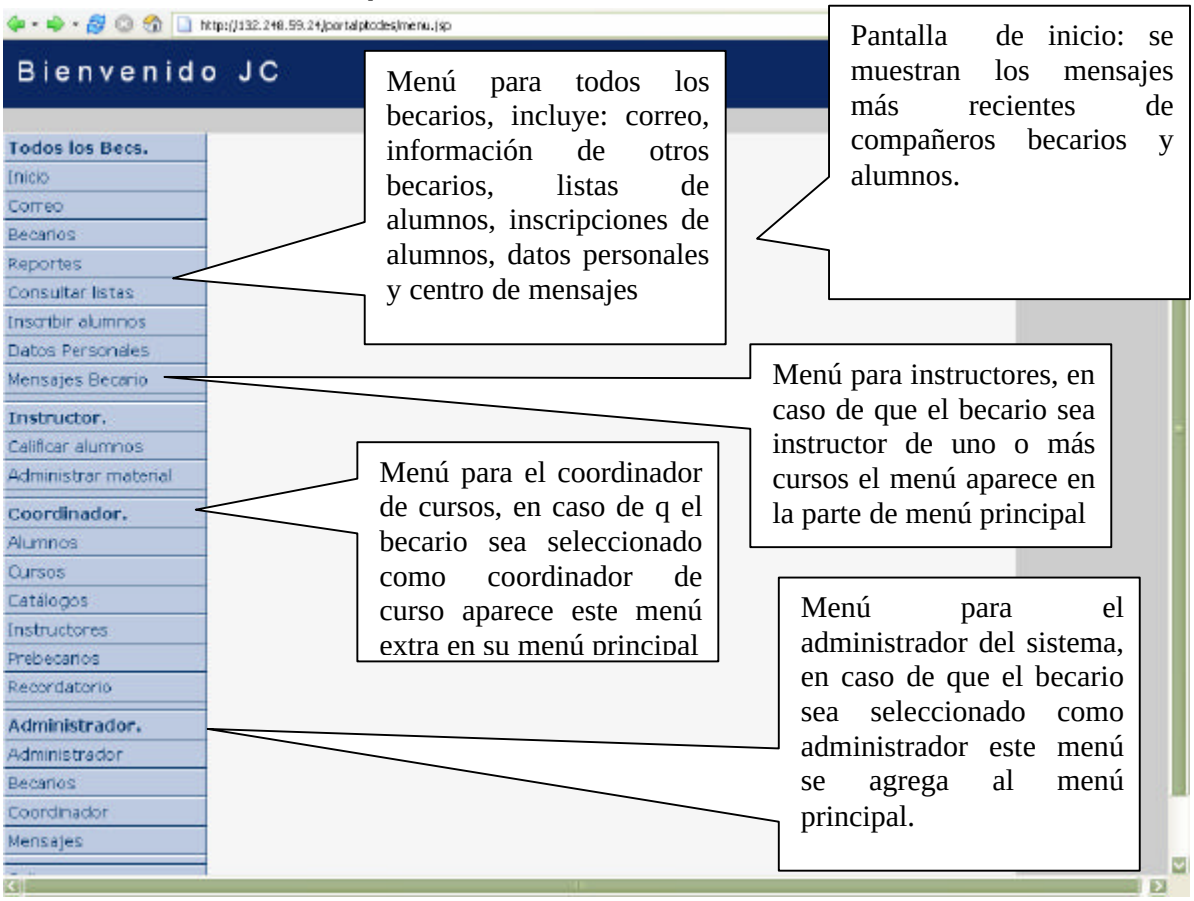
TABLA	**BITACORA**		
CAMPO	***DESCRIPCION***	***TIPO DE DATO***	
ID_BITACORA	Índice de cada proceso de eliminación.	INTEGER	NOT NULL
CALIFICACION	Calificación final del alumno en el curso.	FLOAT	NULL
FECHAELIMINADO	Fecha en se elimino el registro de la tabla LISTAALUMNOS.	DATE	NOT NULL
ID_ALUMNO	Identificador de alumno. Tiene relación con la tabla ALUMNO.	INTEGER	FK
ID_BECARIO	Identificador de becario. Tiene relación con la tabla BECARIO.	INTEGER	FK
ID_CLAVEREGISTRO	Identificador de becario. Tiene relación con la tabla BECARIO.	INTEGER	FK
ID_CURSO	Identificador de becario. Tiene relación con la tabla CURSO.	INTEGER	FK
PAQUETE	Número del paquete inscrito o -1 en caso contrario.	INTEGER	NOT NULL

4.3. Secuencia de pantallas

4.3.1. Entrada al sistema.



4.3.2. Menú Principal

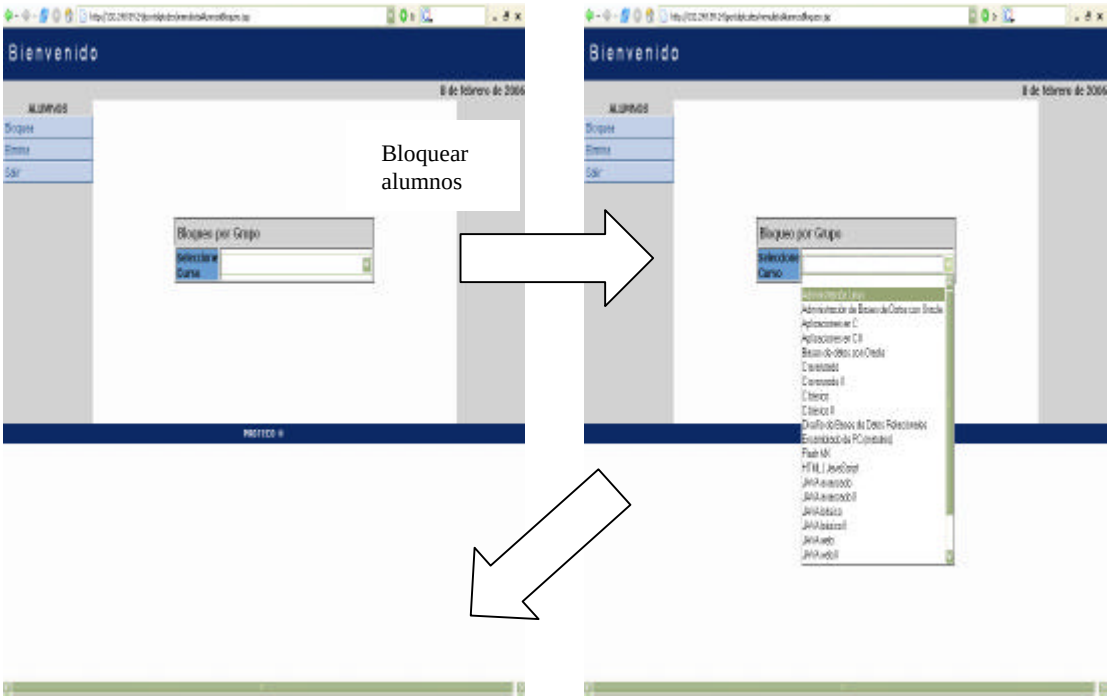


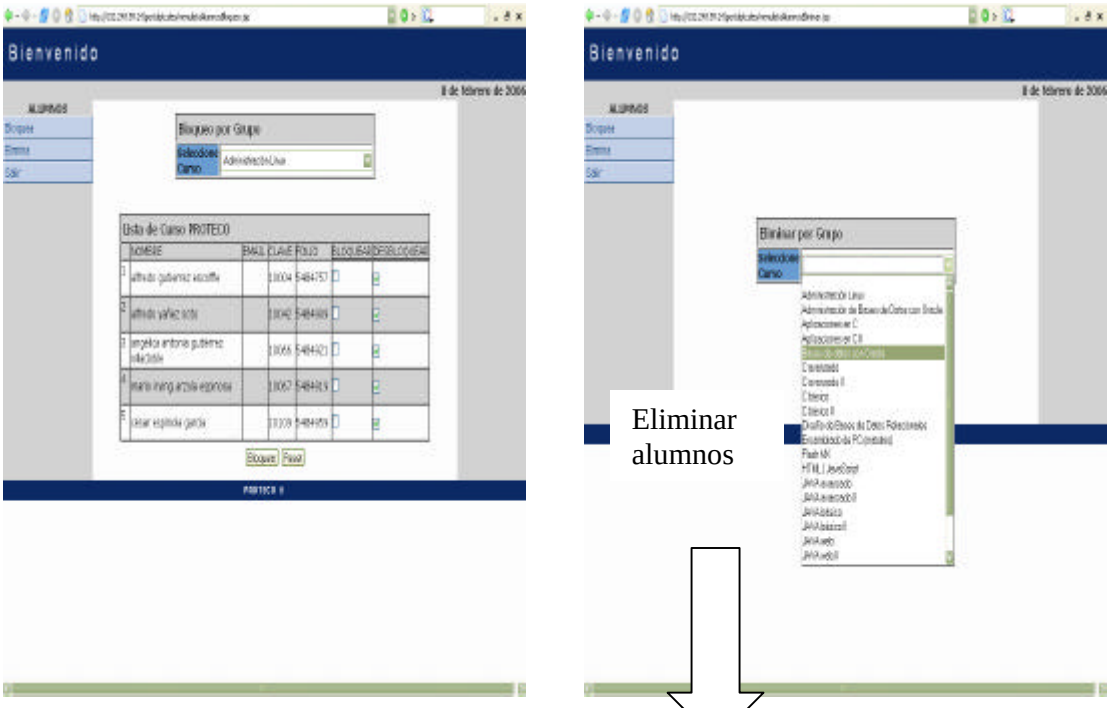
Procedimiento

• Escribir el login y el password, en caso de no contar con una cuenta de acceso, acudir con el administrador del sistema.
• Hacer clic en el botón de *Enviar*.
• Hacer clic en cualquier opción del menú principal, el menú principal varía dependiendo del usuario.

4.3.3. Menú coordinador

- Alumnos.

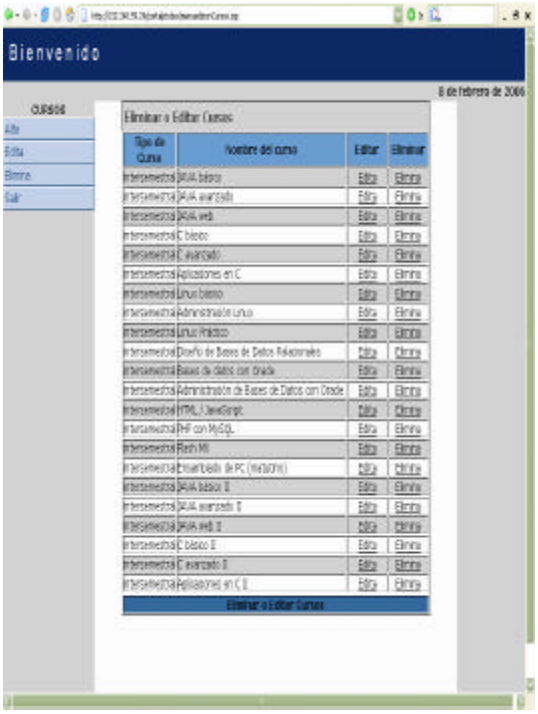


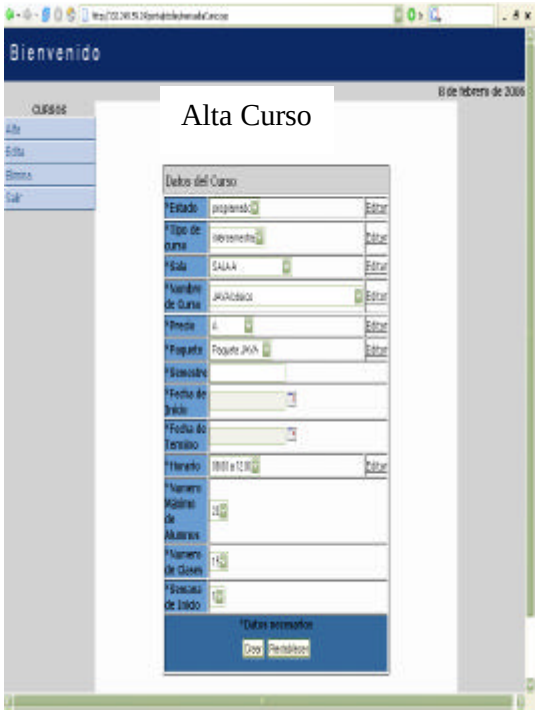


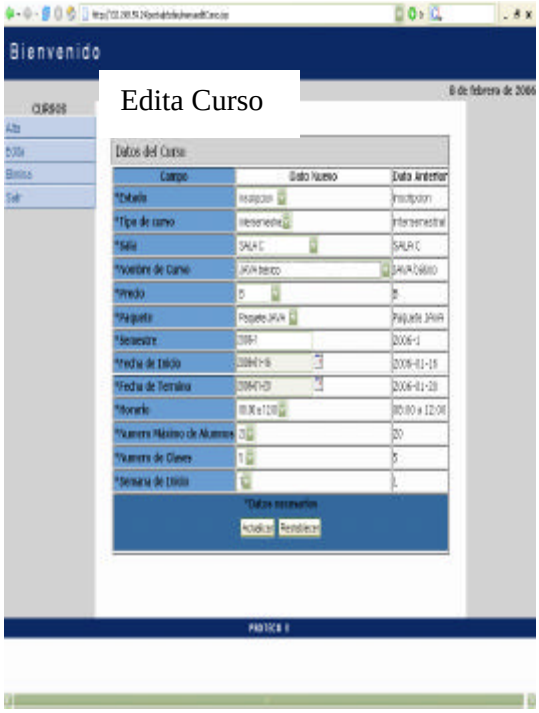
Procedimiento

• Seleccionar en el menú ***Alumnos***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Bloquea***, ***Elimina*** y ***Salir***
• Para bloquear a un alumno se da clic en el botón ***Bloquea***, se selecciona el grupo del menú en pantalla, se da clic en la opción correspondiente. Aparecerá la lista completa de alumnos que se encuentran inscritos en ese grupo, se selecciona al alumno o alumnos a bloquear y se da clic en ***Enviar***.
• Para eliminar a un alumno se da clic en el botón ***Elimina***, se selecciona el grupo del menú en pantalla, se da clic en la opción correspondiente. Aparecerá la lista completa de alumnos que se encuentran inscritos en ese grupo, se selecciona al alumno o alumnos a eliminar y se da clic en el botón de ***Eliminar***.
• Para regresar al menú principal se da clic en ***Salir.***

- Cursos


Bienvenido
Eliminar o Editar Cursos

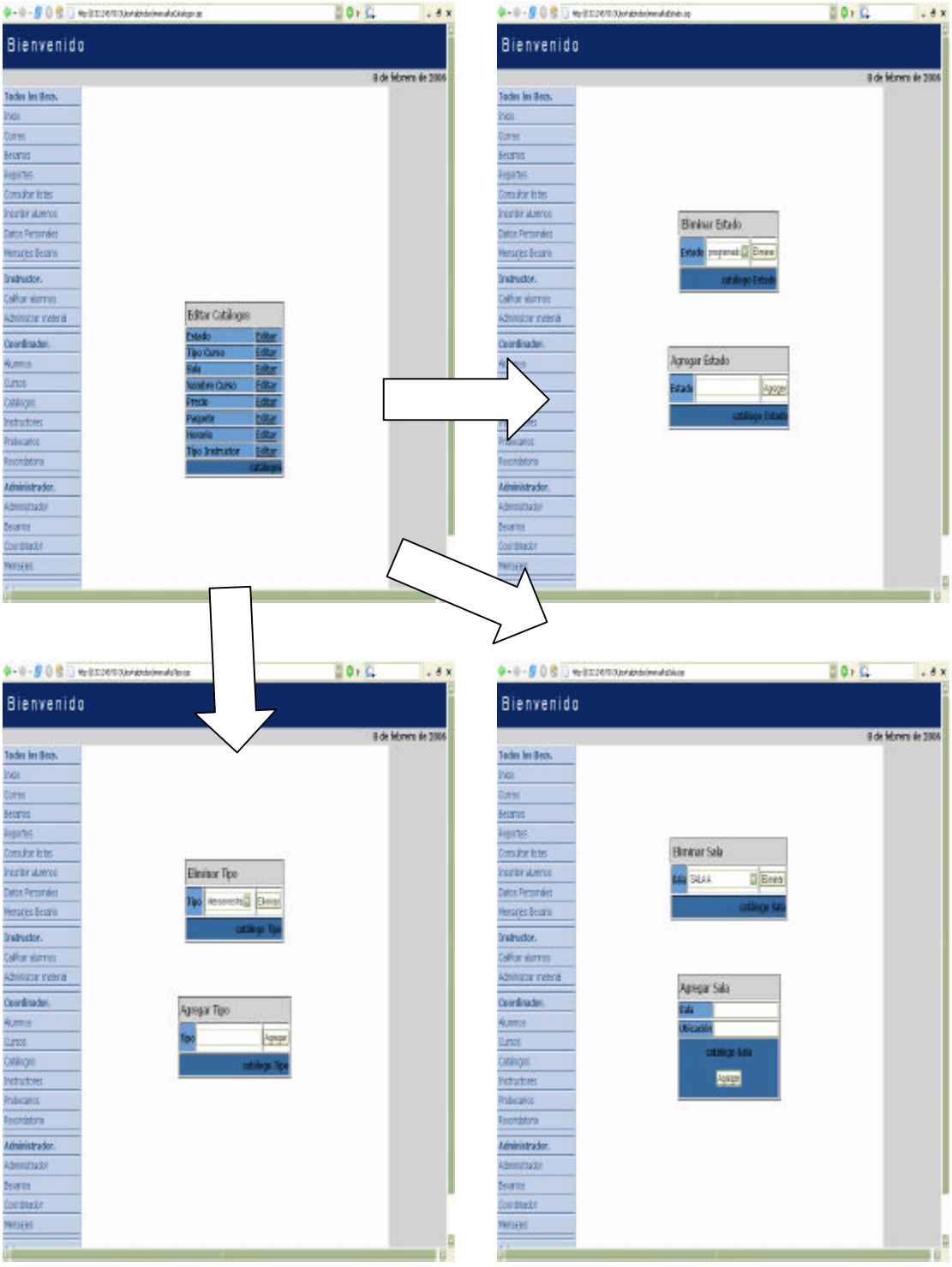

Bienvenido
Alta Curso

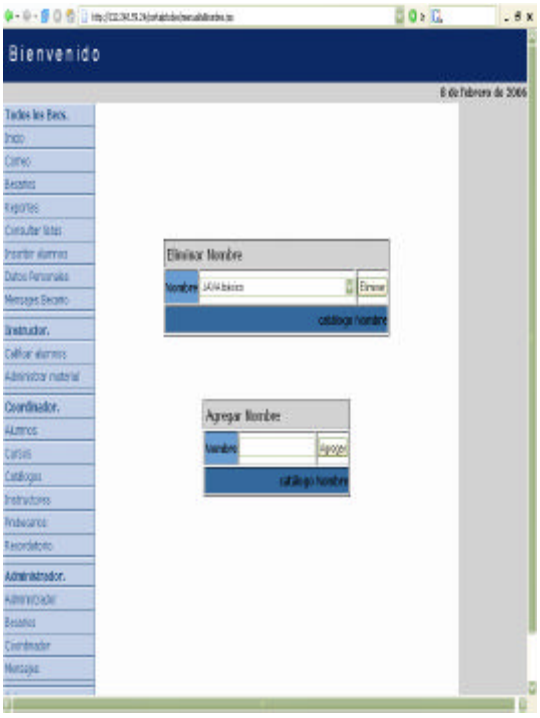

Bienvenido
Edita Curso

Procedimiento

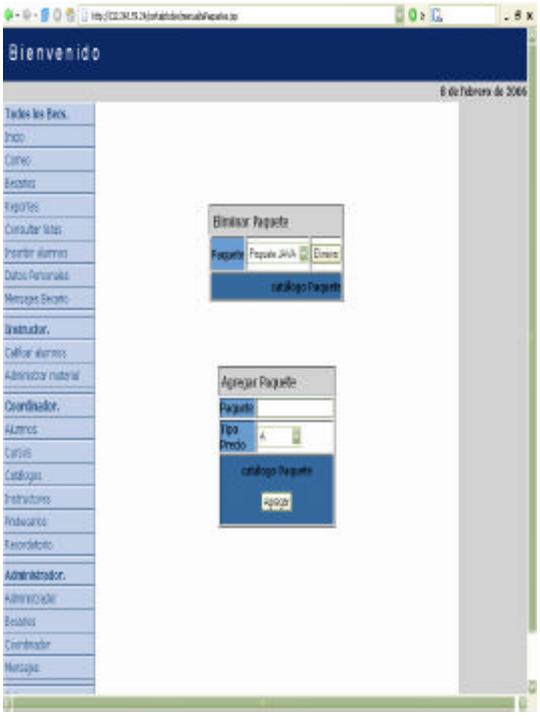
• Seleccionar en el menú ***Cursos***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Alta, Edita***, ***Elimina*** y ***Salir***
• Para dar de alta un curso se da clic en la opción ***Alta***, se llena el formulario correspondiente y se da clic en el botón ***Enviar***.
• Para editar un curso se da clic en la opción ***Edita***. Selecciona el nombre del curso a editar y se da clic en la opción ***Editar***. Se cambian los datos que se requieran y se da clic en ***Enviar***.
• Para eliminar un curso se da clic en la opción ***Elimina***. Selecciona el nombre del curso a eliminar y se da clic en la opción Eliminar.
• Para regresar al menú principal se da clic en ***Salir.***

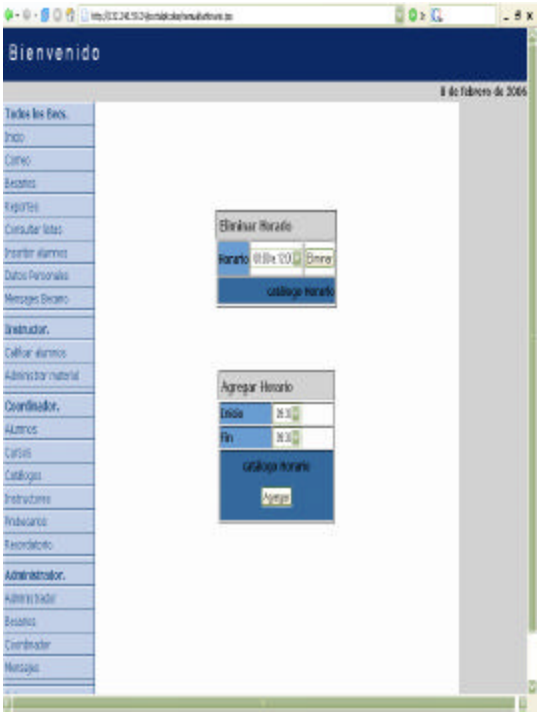
- Catálogos


Bienvenido
Editar Catálogos
Bienvenido
Eliminar Estado
Agregar Estado
Bienvenido
Eliminar Tipo
Agregar Tipo
Bienvenido
Eliminar Sala
Agregar Sala


Bienvenido
Eliminar Nombre
Agregar Nombre


Bienvenido
Eliminar Precios
Agregar Precios

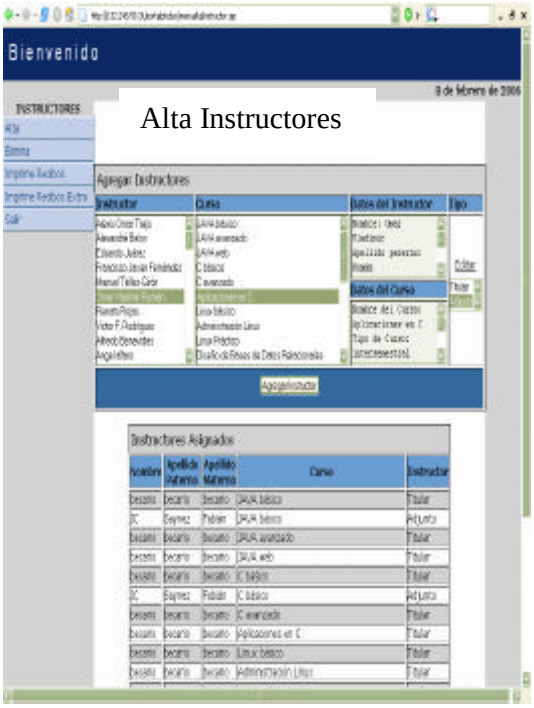

Bienvenido
Eliminar Paquete
Agregar Paquete

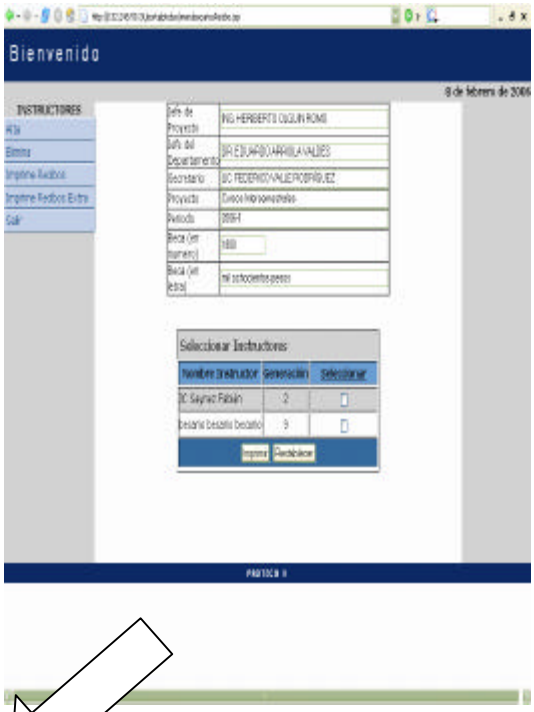

Bienvenido
Eliminar Horario
Agregar Horario

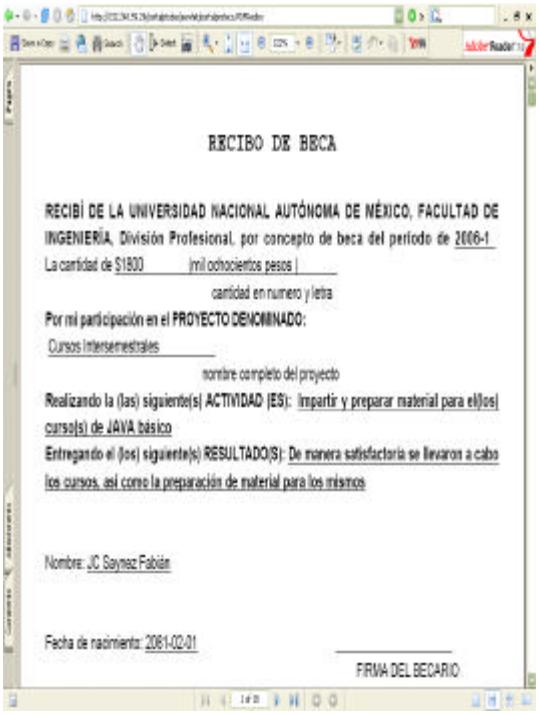
Procedimiento

• Seleccionar en el menú ***Catálogos***. Este botón nos llevará a una pantalla que contiene los catálogos del sistema.
• Seleccionar el catalogo correspondiente a editar. Hacer clic en ***Editar***.
• Para agregar un elemento al catálogo seleccionado se escribe el elemento y se da clic en Aceptar.
• Para eliminar un elemento del catálogo se selecciona y se da clic en ***Eliminar***.

- Instructores







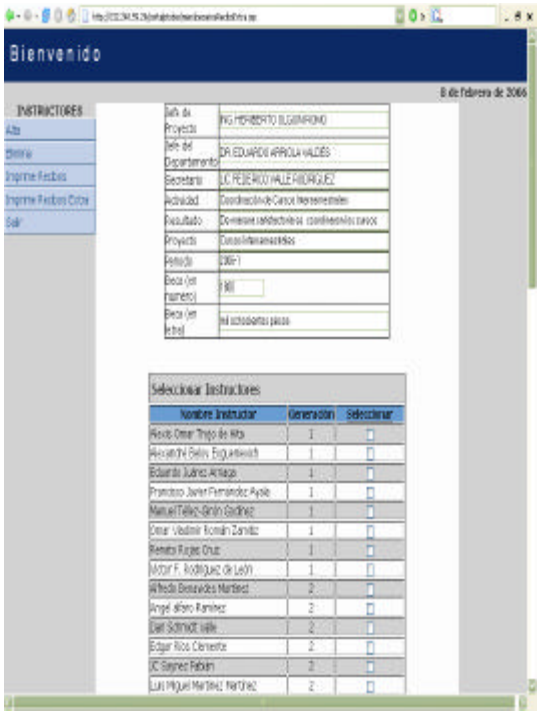
RECIBO DE BECA

RECIBÍ DE LA UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO, FACULTAD DE INGENIERÍA, División Profesional, por concepto de beca del período de 2006-1
La cantidad de $1800 (mil ochocientos pesos)
cantidad en numero y letra
Por mi participación en el PROYECTO DENOMINADO:
Cursos Intersemestrales
nombre completo del proyecto
Realizando la (las) siguiente(s) ACTIVIDAD (ES): Impartir y preparar material para el(los) curso(s) de JAVA básico
Entregando el (los) siguiente(s) RESULTADO(S): De manera satisfactoria se llevaron a cabo los cursos, así como la preparación de material para los mismos

Nombre: JC Saynez Fabián

Fecha de nacimiento: 2081-02-01

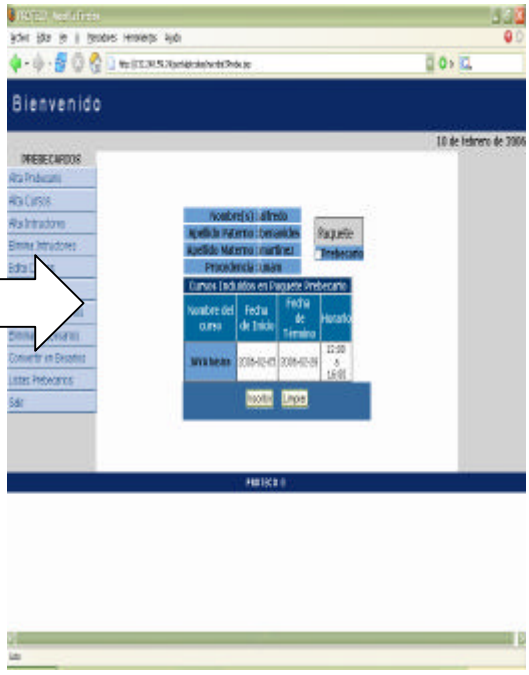
FIRMA DEL BECARIO

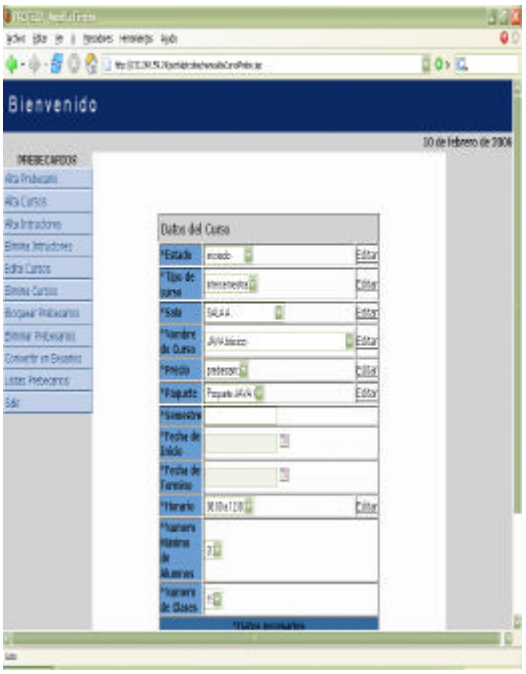


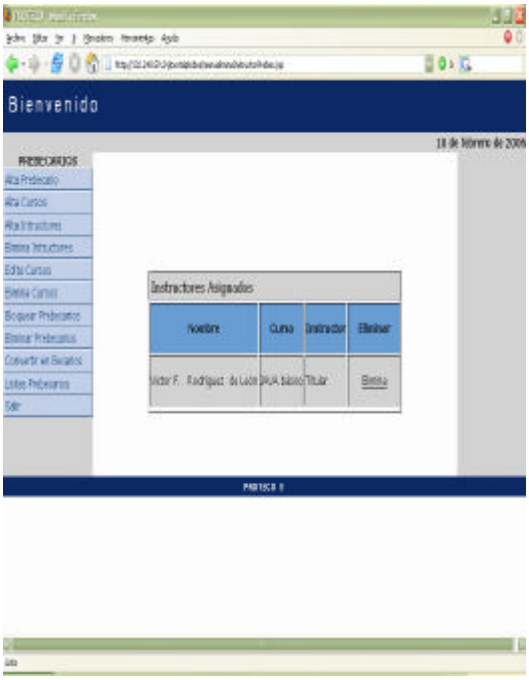
Procedimiento

• Seleccionar en el menú ***Instructor***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Alta, Elimina***, ***Imprime Recibos*** y ***Salir***
• Para dar de alta un instructor se da clic en la opción ***Alta***, se llena el formulario correspondiente y se da clic en el botón ***Agregar***.
• Para eliminar un instructor se da clic en la opción ***Elimina***. Selecciona el nombre del instructor a eliminar y se da clic en la opción ***Eliminar***.
• Para imprimir los recibos de beca de los instructores, se selecciona la opción Imprimir Recibos o Imprimir Recibos extra, se llena el formulario con los datos del recibo y se imprime, la impresión será en documento PDF.
• Para regresar al menú principal se da clic en ***Salir.***

- Prebecarios

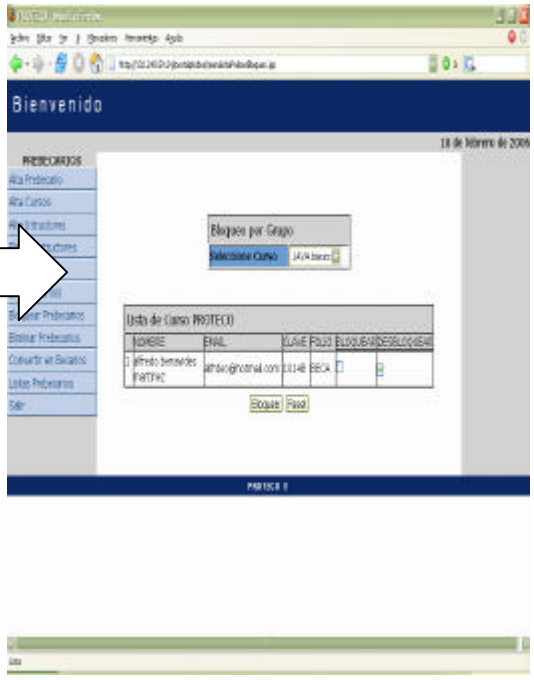


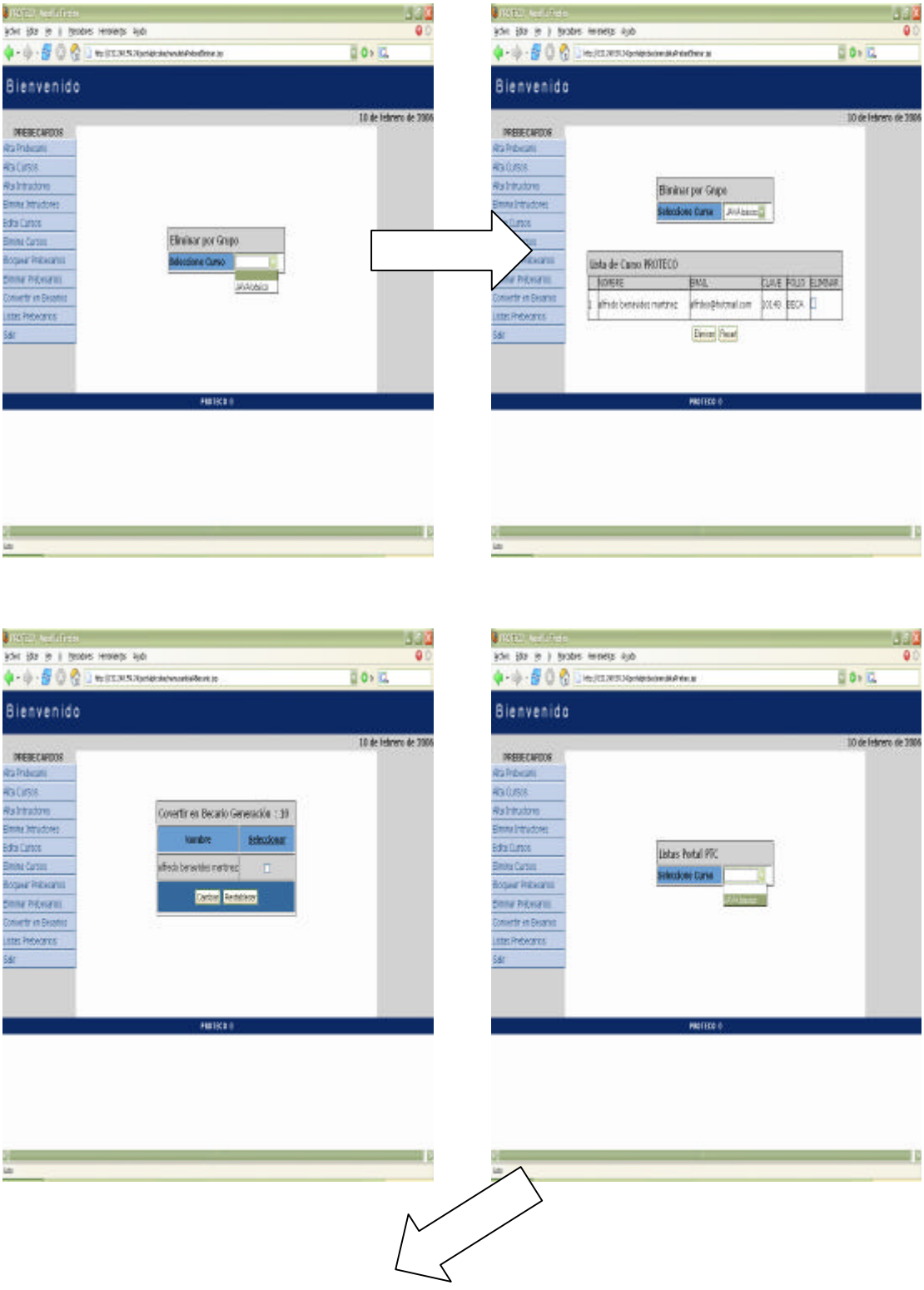



Bienvenido
PREDICARIOS
Instructores Asignados
Nombre
Curso
Instructor
Eliminar


Bienvenido
PREDICARIOS
Eliminar o Editar Cursos
Tipo de Curso
Nombre del curso
Editar
Eliminar


Bienvenido
PREDICARIOS
Bloqueo por Grupo
Seleccione Curso


Bienvenido
PREDICARIOS
Bloqueo por Grupo
Seleccione Curso
Lista de Curso PROTECO
NOMBRE
EMAIL
CLAVE
PAIS
BLOQUEAR
DESBLOQUEAR
Bloquear
Read

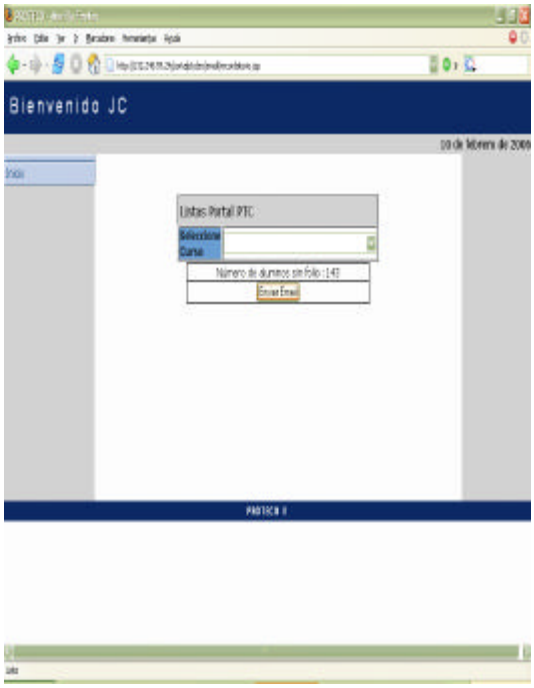

Bienvenido
Eliminar por Grupo
Bienvenido
Eliminar por Grupo
Lista de Curso PROTECO
Bienvenido
Convertir en Becario Generación : 10
Bienvenido
Listas Portal PTC

Procedimiento

• Seleccionar en el menú, *Prebecarios*. Este botón mostrará un submenú que contiene las siguientes opciones: *Alta prebecario, Alta cursos, Alta instructores, Elimina instructores, Edita cursos, Elimina cursos, Bloquea prebecarios, Eliminar prebecarios, Convertir en becarios, Lista becarios, Salir.*
• Para dar de alta un prebecario se da clic en la opción *Alta prebecario*, se llena el formulario correspondiente y se da clic en el botón *Agregar*.
• Para dar de alta un curso de prebecario se da clic en la opción *Alta cursos*, se llena el formulario correspondiente y se da clic en el botón *Enviar*.
• Para dar de alta un instructor se da clic en la opción *Alta*, se llena el formulario correspondiente y se da clic en el botón *Agregar*.
• Para eliminar un instructor se da clic en la opción *Elimina instructores*. Selecciona el nombre del instructor a eliminar y se da clic en la opción *Eliminar*.
• Para editar un curso se da clic en la opción *Edita cursos*. Selecciona el nombre del curso a editar y se da clic en la opción *Editar*. Se cambian los datos que se requieran y se da clic en *Enviar*.
• Para eliminar un curso se da clic en la opción *Elimina cursos*. Selecciona el nombre del curso a eliminar y se da clic en la opción Eliminar.
• Para bloquear prebecarios se da clic en la opción *Bloquea prebecarios*, se selecciona el grupo del prebecario y posteriormente al prebecario a ser bloqueado, se da clic en *Bloquear*.
• Para eliminar prebecario se da clic en la opción Eliminar Prebecarios. Selecciona el curso de los prebecario y posteriormente al prebecario a eliminar, se da clic en *Eliminar*.
• Para convertir en becario a un prebecario, se da clic en *Convertir en becario*, se

seleccionan el o los becarios a convertir y se da clic en ***Enviar***.
• Para ver la lista de prebecario, se da clic en *Ver listas prebecarios*.
• Para regresar al menú principal se da clic en *Salir.*

- Recordatorio



Procedimiento

• Seleccionar en el menú, ***Recordatorio***. Selecciona el grupo y los alumnos a los que se les enviará un mail de recordatorio de inscripción.

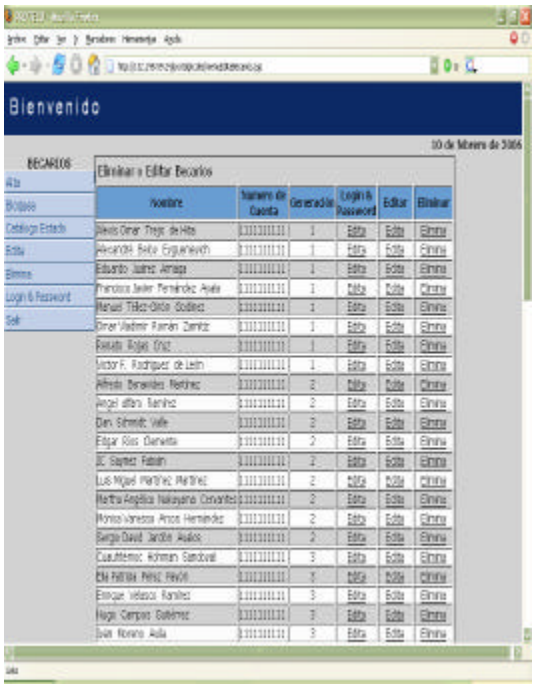
4.3.4. Menú administrador

- Alta administrador.

Procedimiento

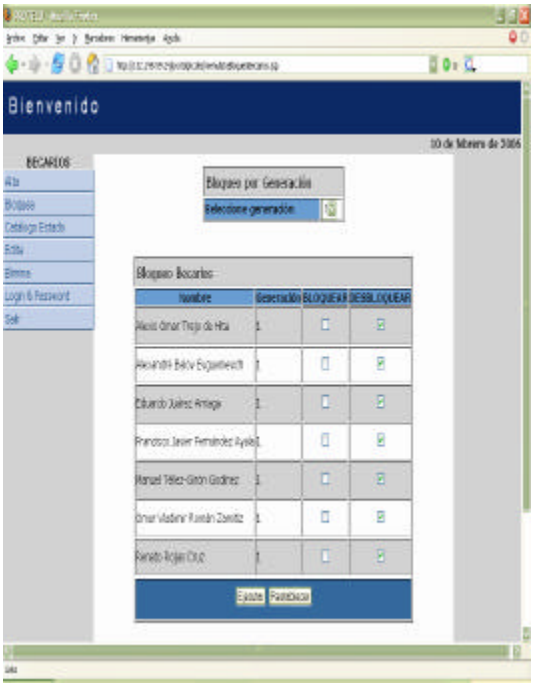
• Seleccionar en el menú, *Administrador*. Selecciona becario con el nuevo de cargo de Administrador y hacer clic en *Enviar*.

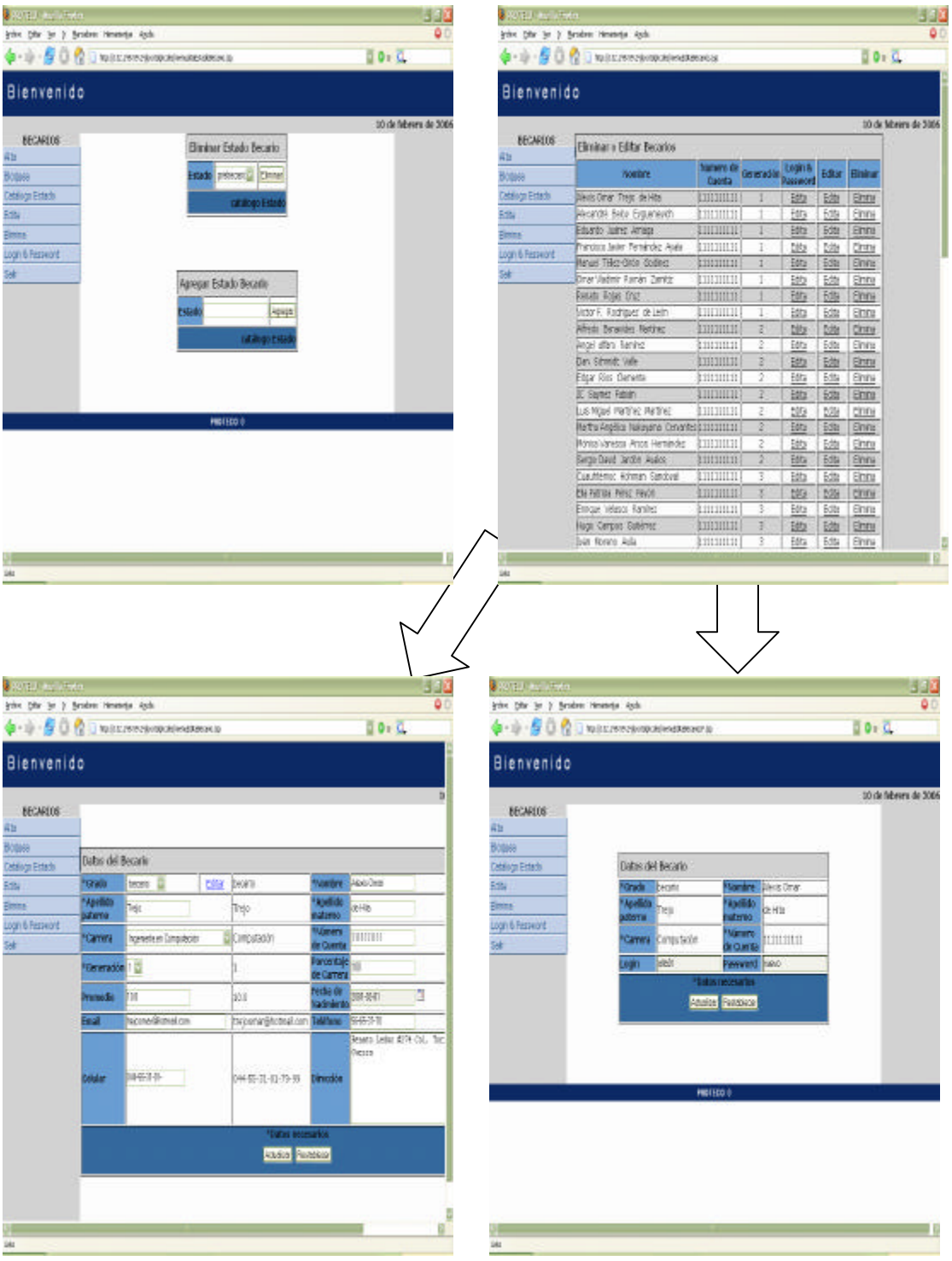
- Becarios.


Bienvenido
BECARIOS
Eliminar o Editar Becarios


Bienvenido
BECARIOS
Datos del Becario


Bienvenido
BECARIOS
Bloqueo por Generación


Bienvenido
BECARIOS
Bloqueo por Generación
Bloqueo Becarios


Bienvenido
Eliminar Estado Becario
Agregar Estado Becario
Eliminar o Editar Becarios
Datos del Becario

Procedimiento

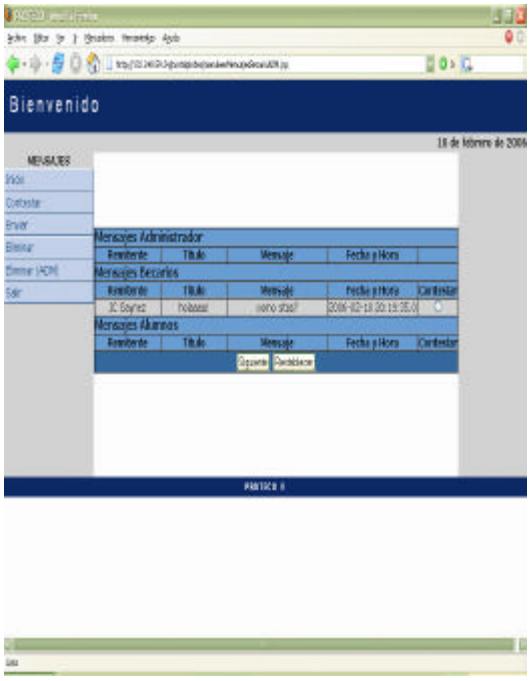
• Seleccionar en el menú, ***Becarios***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Alta, Bloquea, Catálogo estado, Edita, Elimina, Login & Password, Salir.***
• Para dar de alta un becario se da clic en la opción ***Alta***, se llena el formulario correspondiente y se da clic en el botón ***Agregar***.
• Para editar el catalogo de estado del becario, se da clic en Catálogo estado, se agregan elementos o se eliminan del catálogo.
• Para editar los datos de un becario, se da clic en la opción ***Edita***, se selecciona el nombre del becario se da clic en ***Editar***, se guardan los datos y se da clic en ***Enviar***.
• Para eliminar un becario se da clic en ***Elimina***, se selecciona el becario a ser eliminado, y se da clic en ***Eliminar***.
• Para editar el login y password de un becario, se da clic en la opción ***Login & Password***, se selecciona el nombre del becario se da clic en ***Editar***, se guardan los datos y se da clic en ***Enviar***.
• Para regresar al menú principal se da clic en ***Salir.***

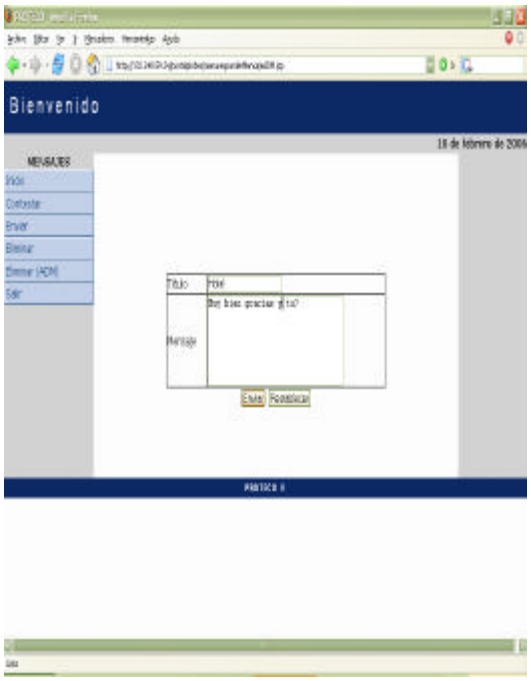
- Alta coordinador de cursos.

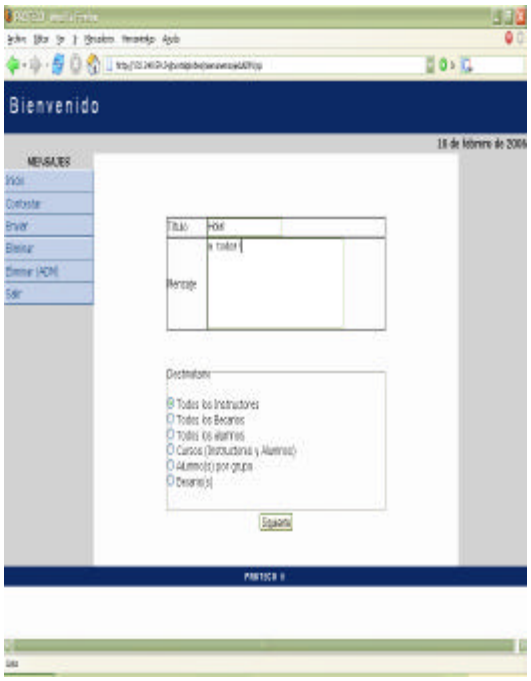
Procedimiento

• Seleccionar en el menú, *Coordinador*. Selecciona el becario con el nuevo cargo de coordinador y hacer clic en *Enviar*.

- Administración del centro de mensajes.








Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Mensaje Alumnos por Grupo
Selecciona Curso


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Mensaje Alumnos por Grupo
Selecciona Curso
Lista de Curso PROTECO
NOMBRE
EMAIL
CLAVE
PG_ID
SELECCIONAR
Enviar
Reset

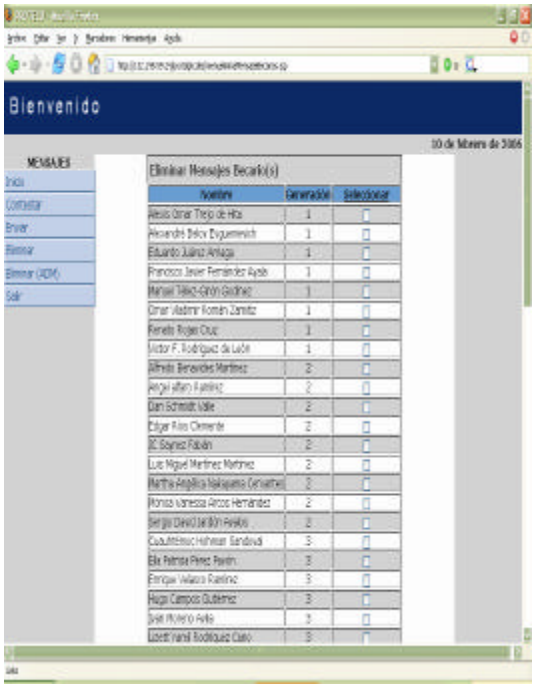

Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Destinatario:
Nombre
Generación
Seleccionar


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Mensajes Administrador
Mensajes Docentes
Mensajes Alumnos
Remitente
Título
Mensaje
Fecha y Hora
Eliminar
Siguiente
Anterior


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Eliminar Mensajes Enviados por:
Eliminar todos los mensajes
Eliminar todos los mensajes para Instructores y Alumnos
Eliminar todos los mensajes para Becario
Eliminar todos los mensajes para Instructor
Eliminar todos los mensajes para Alumno
Eliminar todos los mensajes de Becarios
Eliminar todos los mensajes de Alumnos
Eliminar todos los mensajes de Adm/Coord
Eliminar mensajes de Alumno(s) por grupo
Eliminar mensajes de Becario(s)
Siguiente


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Eliminar Mensajes Alumnos por Grupo
Seleccione Campo


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Eliminar Mensajes Alumnos por Grupo
Seleccione Campo
Administrador Linux
Lista de Curso PROTECO
NOMBRE
EMAIL
Enviar
Reset

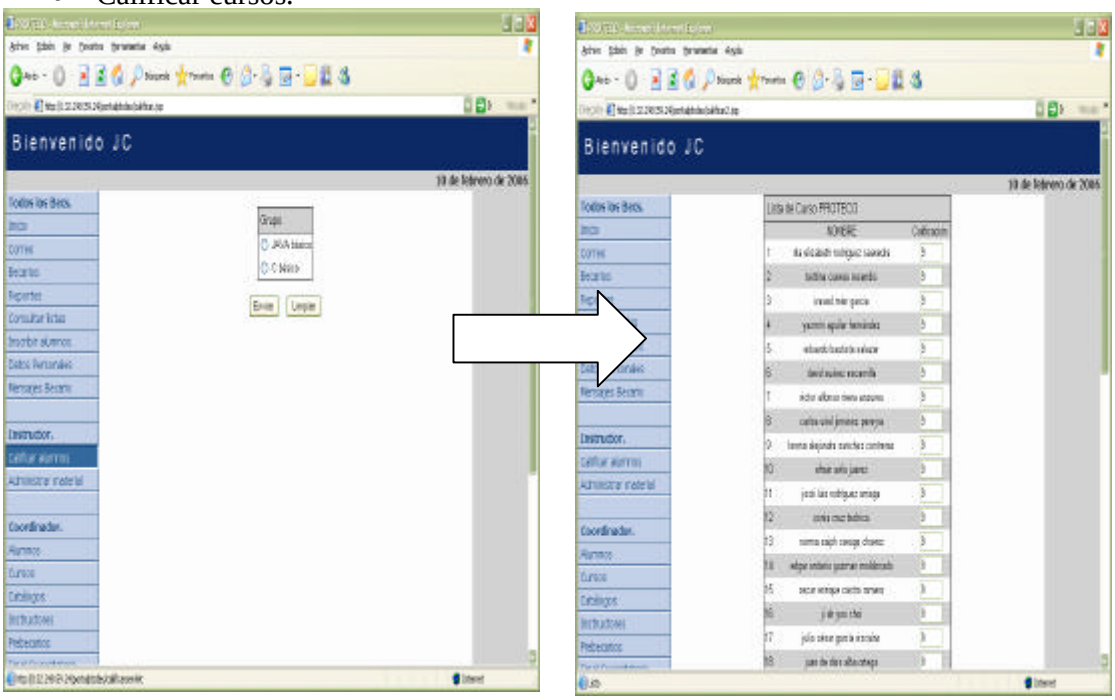

Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Eliminar (ADM)
Salir
Eliminar Mensajes Becario(s)
Nombre
Generación
Seleccionar

Procedimiento

• Seleccionar en el menú, ***Mensajes***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Inicio, Contestar, Eliminar, Enviar, Eliminar (ADM) Salir.***
• Para ver los mensajes recibidos se da clic en ***Inicio*** y todos los mensajes serán desplegados en pantalla.
• Para contestar mensajes, se da clic en Contestar.
• Para eliminar mensajes, se da clic en Eliminar.
• Para enviar un mensaje, se da clic en Enviar.
• Para eliminar los mensajes de otros becarios, se da clic en Eliminar (ADM).
• Para regresar al menú principal se da clic en ***Salir.***

4.3.5. Menú instructor

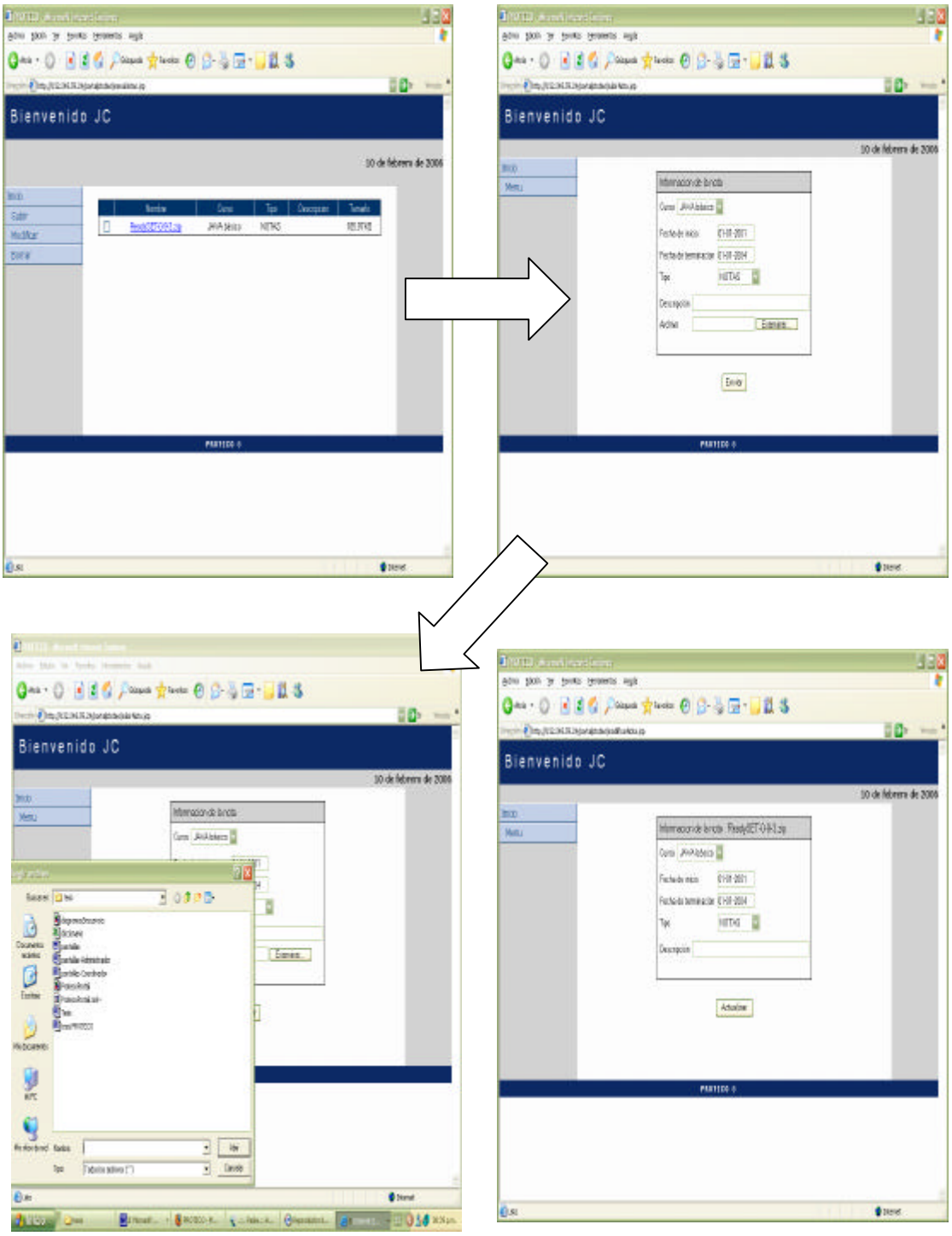
- Calificar cursos.



Procedimiento

• Seleccionar en el menú, *Calificar*. Este botón mostrará un submenú que contiene los grupos en los cuales es instructor el becario, se selecciona un grupo y se califican a los alumnos.

- Material.

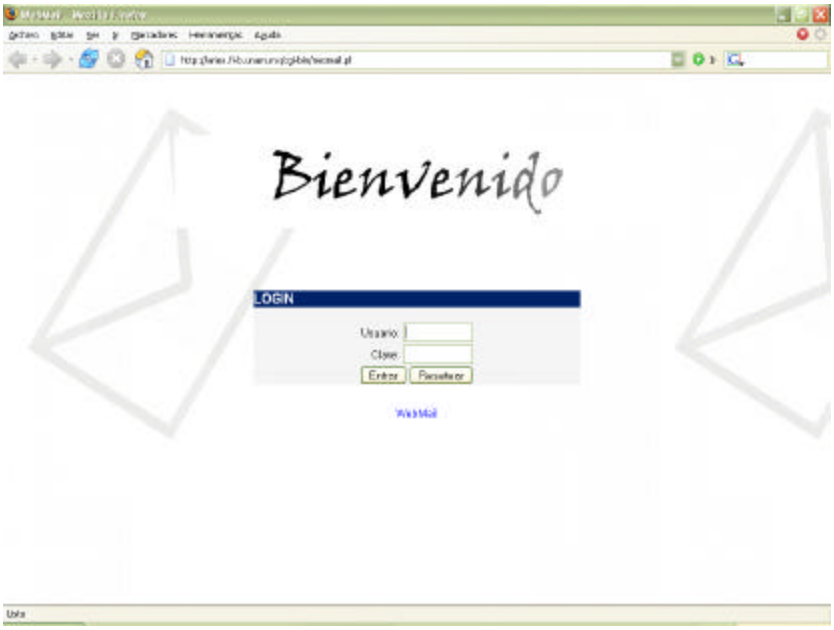


Procedimiento

• Seleccionar en el menú, ***Material***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Inicio, Subir, Modificar, Borrar.***
• Para subir material se da clic en ***Subir***.
• Para modificar los archivos que ya se encuentran en el servidor, se da clic en ***Modificar***.
• Para borrar mensajes, se da clic en Borrar.

4.3.6. Menú becario

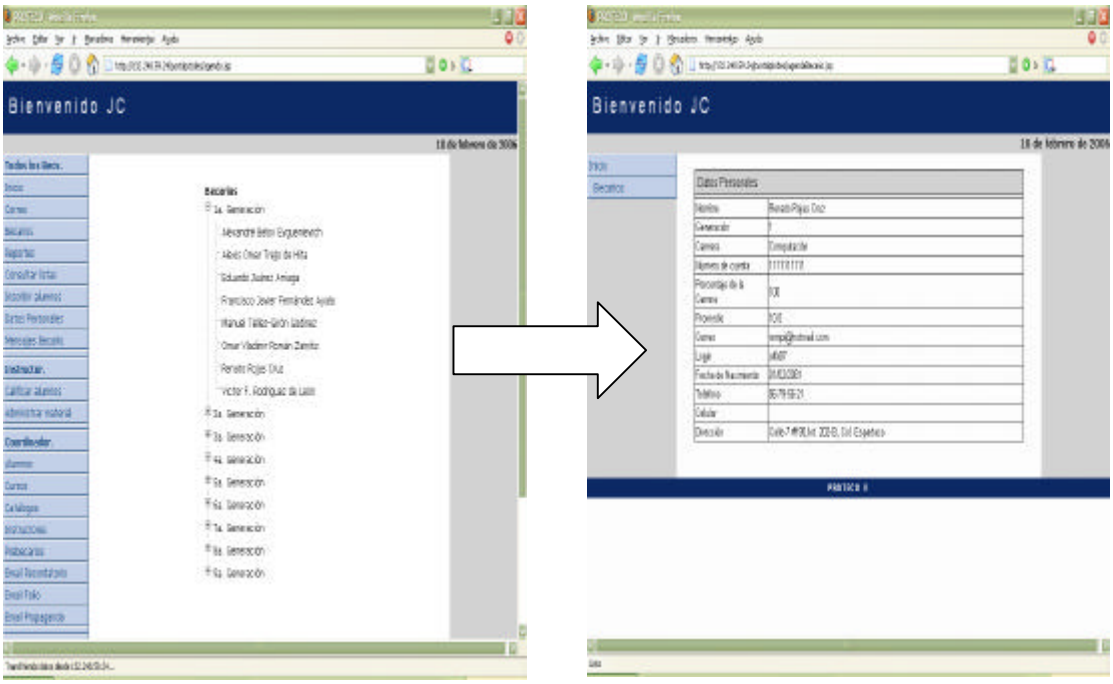
- Correo.



Procedimiento

• Seleccionar en el menú, *Correo*. Este botón mostrará una pantalla que despliega la página de entreda del correo electrónico de la DIE.

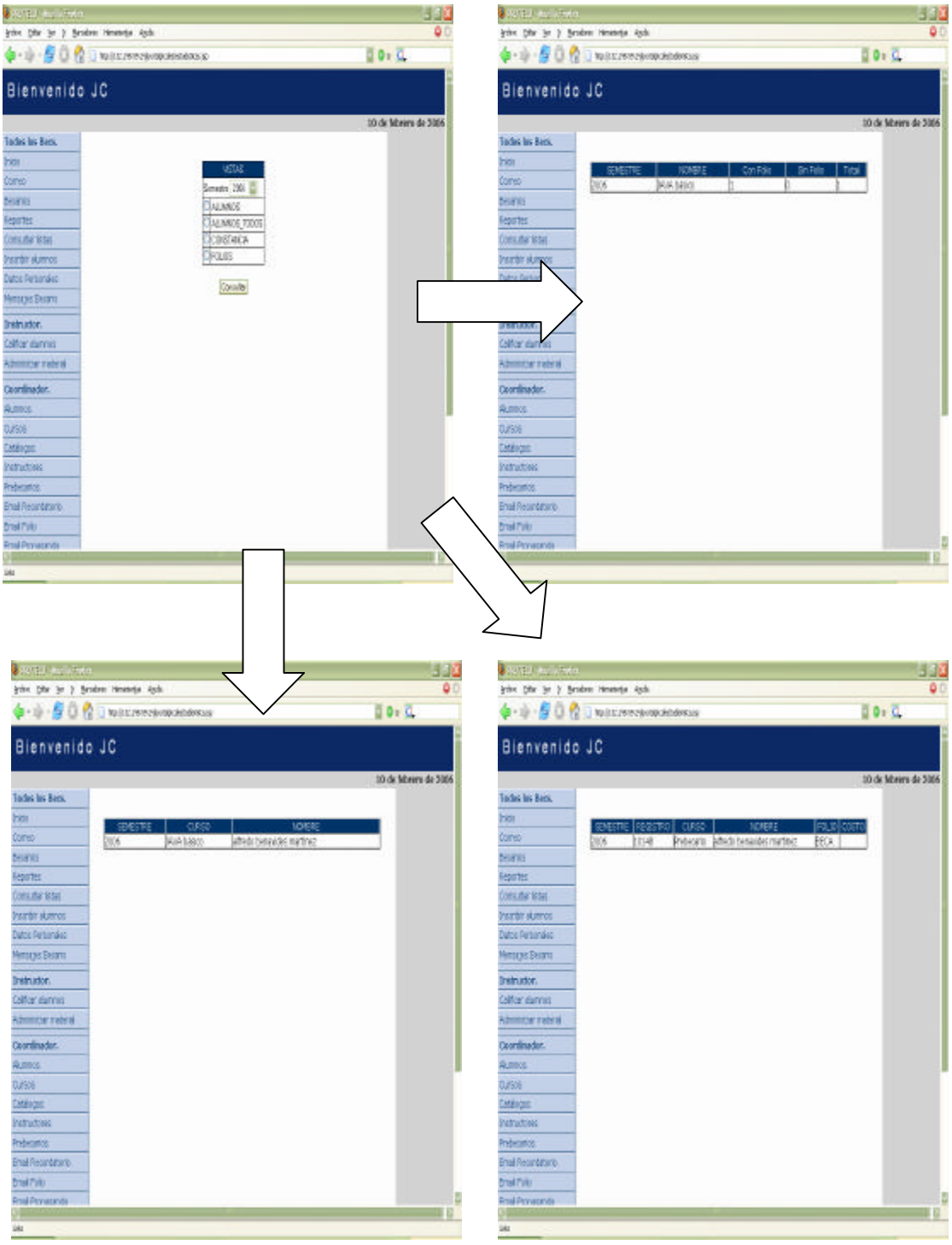
- Becarios.



Procedimiento

• Seleccionar en el menú, ***Becarios***. Este botón nos lleva a una pantalla que despliega las generaciones de becarios que se encuentran inscritas en el sistema, al hacer clic en el nombre de cualquier becario, se despliega la información correspondiente al mismo becario.

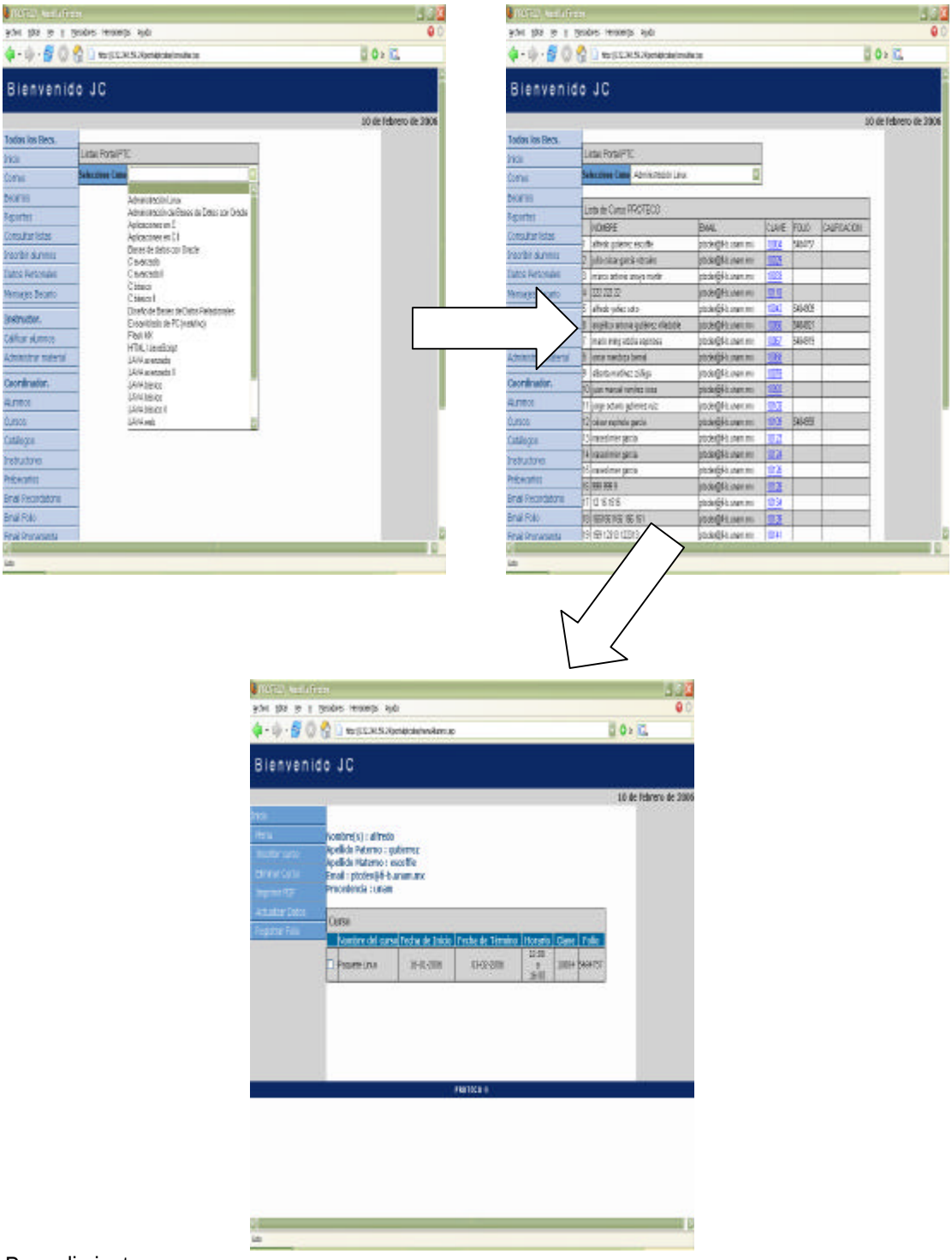
- Reportes



Procedimiento

• Seleccionar en el menú, ***Reportes***. Este botón nos llevará a una pantalla que nos muestra reportes de los semestres de cursos que se han impartido en dos modalidades alumnos con derecho a constancia y folios.

- Listas



Procedimiento

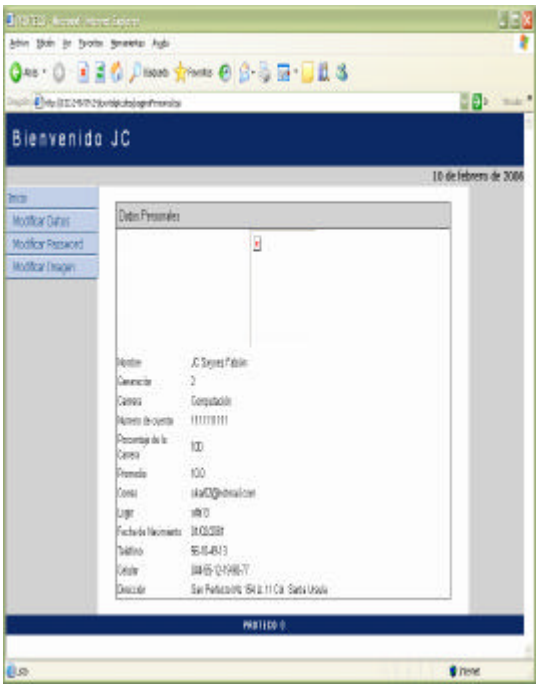
• Seleccionar en el menú, *Consultar listas*. Este botón nos llevará a una pantalla con todos los cursos que se estén desarrollando, se consulta el total de alumnos preinscritos e inscritos. Además se accede a la información de cada alumno inscrito.

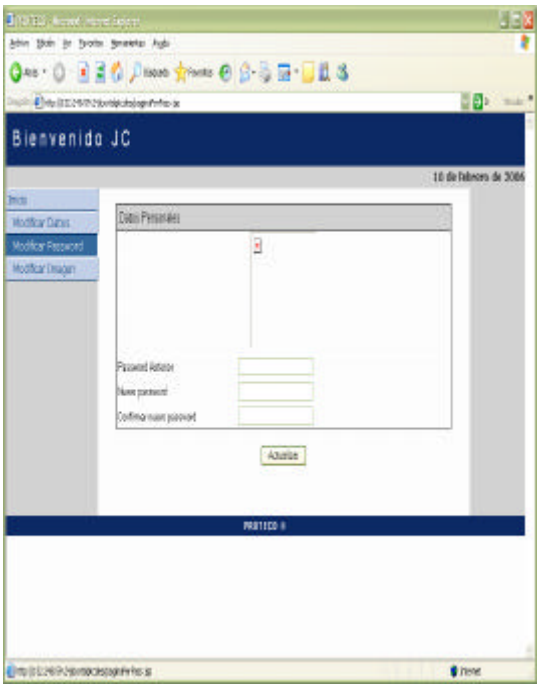
- Inscripción alumnos.

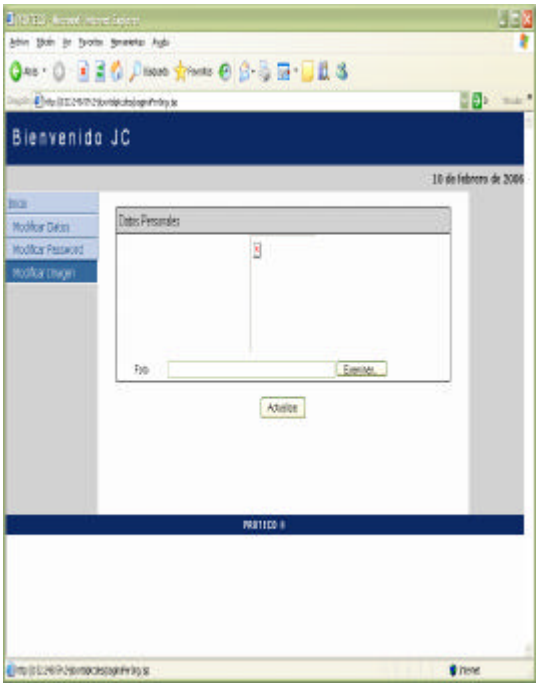
Procedimiento

• Seleccionar en el menú, ***Inscribir alumno***. Este botón nos llevará a una pantalla de formulario que permite inscribir la información básica del alumno a inscribirse.
• Para guardar los datos se da clic en ***Preinscribir***.
• La siguiente pantalla nos muestra los paquetes y grupos disponibles, para la inscripción, en código de colores se muestra la disponibilidad de cada uno de estos. Se selecciona el curso o los cursos deseados y se da clic en ***Preinscribir***
• La siguiente pantalla nos muestra un submenú que cuenta con los siguientes campos: ***Inicio, Menú, Inscribir curso, Borrar curso, Imprimir PDF, Actualizar datos, Registrar folio.***
• Para regresar al menú de Inicio se da clic en el botón de ***Inicio***.
• Para regresar al menú de inscripción se da clic en el botón de ***Menú***.
• Para inscribir al alumno en un nuevo curso se da clic en ***Inscribir curso***.
• Para borrar a un alumno de un curso preinscrito, se selecciona el curso a borrar y se da clic en el botón de ***Borrar curso***.
• Para imprimir él o los comprobantes de preinscripción, se seleccionan los cursos o el curso y se da clic en ***Imprimir PDF.*** El formato de impresión es pdf.
• Para actualizar los datos del alumno, se da clic en ***Actualizar datos***.
• Para registrar el folio de inscripción de los cursos, se selecciona ***Registrar Folio***.

- Datos personales.







Procedimiento

• Seleccionar en el menú, *Datos personales*. Este botón mostrará un submenú el cual tiene las siguientes opciones: *Inicio, Modificar Datos, Modificar password, Modificar Imagen*.
• Para modificar los datos personales, seleccionar *Modificar datos*.
• Para cambiar tu password, seleccionas *Modificar password.*
• Para cambiar la imagen de despliegue, seleccionas *Modificar imagen*.

- Mensajes


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Mensajes Administrador
Mensajes Docentes
Mensajes Alumnos


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Título
Mensaje
Enviar
Restablecer


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Título
Mensaje
Destinatario
Siguiente


Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Seleccionar Curso:
Seleccione Curso
Cálculo
PROTECO ©


Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Seleccionar Curso:
Seleccione Curso
Lista de Curso PROTECO
NOMBRE
EMAIL
CLAVE
FOLIO
SELECCIONAR

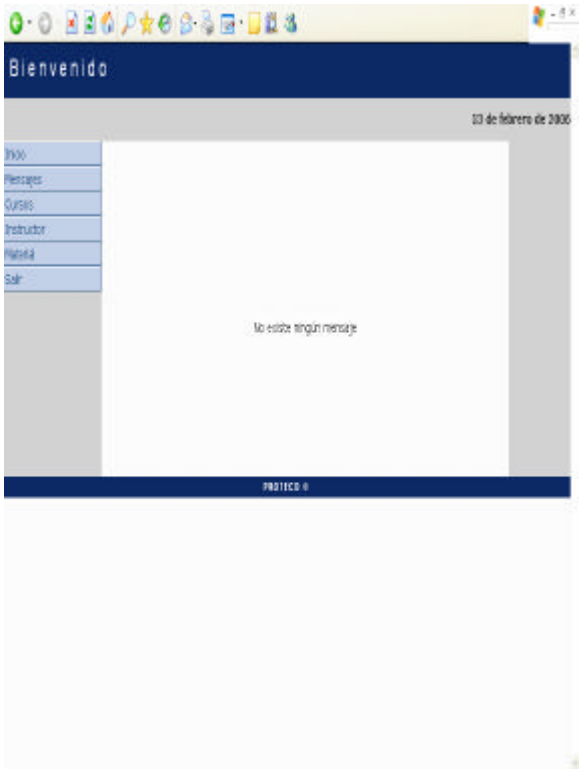

Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Destinatario:
Nombre
Generación
Seleccionar


Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Mensajes Administrador
Remitente
Título
Mensaje
Fecha y Hora
Mensajes Docentes
Eliminar
Mensajes Alumnos
Siguiente
Precedente
PROTECO ©

Procedimiento

• Seleccionar en el menú, ***Mensajes***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Inicio, Contestar, Eliminar, Enviar, Salir.***
• Para ver los mensajes recibidos se da clic en ***Inicio*** y todos los mensajes serán desplegados en pantalla.
• Para contestar mensajes, se da clic en ***Contestar***.
• Para eliminar mensajes, se da clic en ***Eliminar***.
• Para enviar un mensaje, se da clic en ***Enviar***.
• Para regresar al menú principal se da clic en ***Salir.***

4.3.7. Menú alumno



Procedimiento

• Ingresar login y password de alumno en la sección de alumno.

- Mensajes


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Mensajes Administrador
Mensajes Docentes
Mensajes Alumnos


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Titulo
Mensaje
Enviar
Restablecer


Bienvenido
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Titulo
Mensaje
Destinatario
Siguiente


Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Seleccionar Curso:
Seleccione Curso
Cálculo


Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Seleccionar Curso:
Seleccione Curso
Lista de Curso PROTECO
NOMBRE
EMAIL
CLAVE
FOLIO
SELECCIONAR


Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Destinatario:
Nombre
Generación
Seleccionar


Bienvenido
10 de febrero de 2006
MENSAJES
Inicio
Contestar
Enviar
Eliminar
Salir
Mensajes Administrador
Mensajes Docentes
Mensajes Alumnos
Remitente
Título
Mensaje
Fecha y Hora
Eliminar
Siguiente
Precedente

Procedimiento

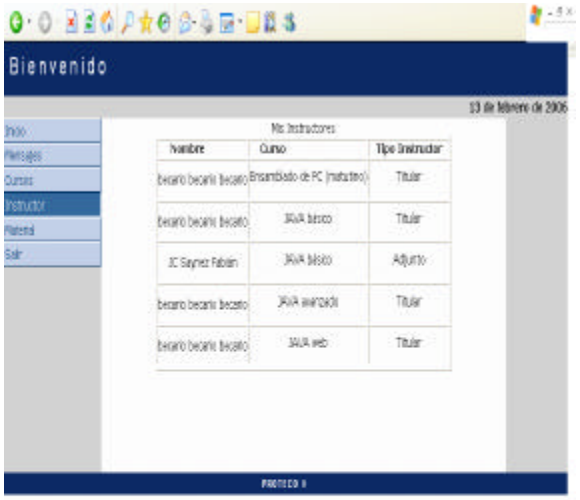
• Seleccionar en el menú, ***Mensajes***. Este botón mostrará un submenú que contiene las siguientes opciones: ***Inicio, Contestar, Eliminar, Enviar, Salir.***
• Para ver los mensajes recibidos se da clic en ***Inicio*** y todos los mensajes serán desplegados en pantalla.
• Para contestar mensajes, se da clic en ***Contestar***.
• Para eliminar mensajes, se da clic en ***Eliminar***.
• Para enviar un mensaje, se da clic en ***Enviar***.
• Para regresar al menú principal se da clic en ***Salir.***

- Cursos

Procedimiento

• Para ver la información de los cursos en los que el alumno se encuentra inscrito, se da clic en *Cursos*.

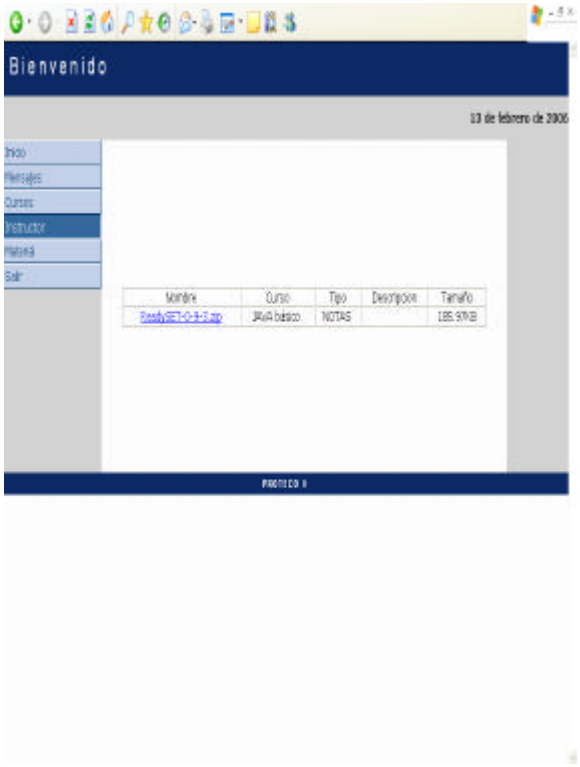
- Instructores



Procedimiento

• Para ver la información de los instructores asignados a los cursos en los que el alumno se encuentra inscrito, se da clic en ***Instructor***.

- Material



Procedimiento

• Para descargar los archivos que los instructores hayan subido, se da clic en *Material*.

4.4. Elección del lenguaje de programación

Para la elección del lenguaje de programación se tomó como punto de partida el diseño del sistema, pensando en particular en un lenguaje que facilitara la transición del diseño a la programación del sistema, el lenguaje seleccionado seria utilizado para implementar de manera sencilla el modelo vista controlador, además, se requería que el leguaje contara con la capacidad para desarrollar aplicaciones para el Web de manera segura y eficiente.

Existen varios lenguajes para el desarrollo de aplicaciones Web. No es posible considerar una solución "definitiva" para el desarrollo. Más que indicar un ganador absoluto, se hizo un repaso de las ventajas y desventajas de cada una de estas soluciones y se estableció cómo se encuentran posicionados en el mundo laboral. A continuación se mencionan algunas tecnologías que se encuentran en el mercado, las cuales se consideraron para el sistema.

4.4.1. Perl.

Perl es el decano de los lenguajes de desarrollo Web. En 1998 casi todo Internet estaba hecho con Perl, poco a poco esto ha cambiado, pero eso no significa que Perl no haya tenido una evolución competitiva desde entonces. El problema fundamental con Perl consistía en la sobrecarga de trabajo que imponía al Webserver, cada vez que se ejecutaba un CGI un nuevo proceso se iniciaba y entre más procesos más lento es un equipo. Para resolver este problema se desarrollo Apache-Perl, también conocido como mod_perl, este es un programa que mezcla Apache con el motor Perl en un mismo demonio de modo que los scripts ya no son CGIs sino que están dentro de Apache, todo como un proceso único.

Perl es una de los lenguajes más poderosos y versátiles que existen en el mundo. Posee una gran cantidad de librerías para hacer de todo y a lo largo de los años se ha reunido una enorme documentación sobre su uso.

4.4.2. ASP

Active Server Pages es una solución de Microsoft basada en Visual Basic con las ventajas y desventajas que ello implica. La principal ventaja de ASP es que hay un flujo constante de trabajo para estos desarrolladores. Sin embargo, se debe tomar está información con cautela pues las tendencias actuales pronostican un decremento de los servidores de Microsoft y un aumento en el sistema Linux y BSD. Además ASP es un sistema con nula portabilidad debido a que requiere necesariamente de un servidor Windows, con todas las implicaciones de alto costo, poca flexibilidad.

4.4.3. JSP

Los JSP (Java Server Pages) fueron la respuesta de SUN a las tecnologías Web después de la aparición de los Applets. JSP es la tecnología que más ha penetrado en el mundo. Los JSP ofrecen una gran versatilidad al momento de pensar en como resolver un problema, además, Java es un lenguaje serio, altamente tipificado y que invita a desarrollar código bien estructurado y orientado a objetos.

4.4.4. PHP

La tecnología PHP Hypertext Processor fue considerada durante mucho tiempo por sus críticos como un juguete. PHP, como muchos otros desarrollos OpenSource, nació como un pasa tiempo en un garaje. Su facilidad de uso, la rapidez de su motor y su alianza con MySQL lo han convertido en casi un estándar de la red. Su presencia, en el impresionante número de 10 millones de servidores lo ha llevado a estar muy por encima de cualquier otro lenguaje script. La declaración definitiva de su fuerza llegó cuando en el 2002 Yahoo anunció que cambiaría todos sus servicios a este lenguaje. Hasta ese momento PHP había sido acusado de inseguro y poco escalable. La realidad es que PHP es sumamente escalable, si consideramos "escalable" como la capacidad de un sistema de aumentar el número de sus usuarios aumentando sus recursos y sin perder ninguna de sus ventajas. PHP sigue el concepto KISS (Keep it simple stupid!) al momento de ofrecer una solución. PHP5 ofrece una API madura para la programación orientada a objetos.

4.4.5. Python

Es rápido, intuitivo, con una sintaxis estructurada y por supuesto, libre. Python está pensado para programar clases desde el inicio, lo que lo hace ideal para la programación orientada a objetos. Las empresas futuristas del ramo de la tecnología (como Google) lo están tomando como su lenguaje base. Uno de los problemas graves de Python es su escasa documentación y el reducido número de aplicaciones existentes.

4.4.6. Elección

Java cuenta con las herramientas necesarias para el desarrollo de nuestra aplicación con tecnologías seguras y confiables como los JSP y los Servlets, además de ser una tecnología libre y contar con demás características ya conocidas de Java que lo han posicionado como líder de tecnología en su ramo.

La tecnología Java, una tecnología madura, extremadamente eficaz y sorprendentemente versátil, se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.
- Crear programas para que funcionen en un navegador Web y en servicios Web.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados.

Java al ser parte de las tecnologías líderes en software libre se convierte en una solución multiplataforma que permite el desarrollo de tecnologías Web, que para este proyecto resulta indispensable, debido a su robustez y confiabilidad.

El lenguaje de programación Java ha sido totalmente mejorado, ampliado y probado por una comunidad activa de unos cuatro millones de desarrolladores de software.

4.5. Elección del manejador de base de datos

Necesitábamos una base de datos con la suficiente capacidad y confiabilidad que nos permitiera múltiples conexiones sin escatimar en velocidad. En el laboratorio de computación de la FI, contamos con Oracle como manejador y gestor de la base de datos. Se eligió esta base de datos por las siguientes razones:

Oracle es un sistema de administración de base de datos (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

Soporte de transacciones.
Estabilidad.
Escalabilidad.
Es multiplataforma.

Su mayor defecto es su enorme precio, que es de varios miles de pesos (según versiones y licencias). Otro aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird.

5. Programación

5.1. Java

5.1.1. Historia

La tecnología Java se creó como una herramienta de programación en una pequeña operación secreta y anónima denominada "the Green Project" en Sun Microsystems en el año 1991.

El equipo secreto ("Green Team"), compuesto por trece personas y dirigido por James Gosling, se encerró en una oficina desconocida de Sand Hill Road en Menlo Park, interrumpió todas las comunicaciones regulares con Sun y trabajó sin descanso durante 18 meses.

Intentaban anticiparse y prepararse para el futuro de la informática. Su conclusión inicial fue que al menos en parte se tendería hacia la convergencia de los dispositivos digitales y los ordenadores.

El resultado fue un lenguaje de programación que no dependía de los dispositivos denominado "Oak".

Para demostrar cómo podía contribuir este nuevo lenguaje al futuro de los dispositivos digitales, el equipo desarrolló un controlador de dispositivos de mano para uso doméstico destinado al sector de la televisión digital por cable. Por desgracia, la idea resultó ser demasiado avanzada para el momento y el sector de la televisión digital por cable no estaba listo para el gran avance que la tecnología Java les ofrecía.

Pero poco tiempo después Internet estaba listo para la tecnología Java y justo a tiempo para su presentación en público en 1995, el equipo pudo anunciar que el navegador “Netscape Navigator” incorporaría la tecnología Java.

Actualmente, ya con más de 10 años de existencia, la plataforma Java ha atraído a cerca de 4 millones de desarrolladores de software, se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, ordenadores y redes de cualquier tecnología de programación.

De hecho, su versatilidad y eficiencia, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación a redes, de manera que hoy en día, más de 2,500 millones de dispositivos utilizan la tecnología Java.

- Más de 700 millones de computadoras.
- 708 millones de teléfonos móviles y otros dispositivos de mano (fuente: Ovum)
- 1000 millones de tarjetas inteligentes

Además de sintonizadores, impresoras, cámaras para el Web, juegos, sistemas de navegación para automóviles, terminales de lotería, dispositivos médicos, cajeros de pago en estacionamientos, etc.

5.1.2. Arquitectura

Modelo Vista Controlador

El objetivo de un patrón de sistema es llevar una aplicación al nivel más abstracto de la arquitectura. Los patrones de sistema pueden aplicarse a los procesos principales de una aplicación, o incluso entre aplicaciones.

El Modelo Vista Controlador (o simplemente MVC), divide un componente o un sistema en tres partes lógicas - modelo, vista y controlador – facilitando la modificación o personalización de cada parte.

MVC es útil cuando hay un componente o subsistema que tiene alguna de las siguientes características:

- Es posible ver el componente o subsistema de distintas formas. La representación interna del sistema podría ser completamente diferente de la representación en la pantalla.
- Hay diferentes tipos de comportamiento, lo cual quiere decir que múltiples fuentes pueden invocar el comportamiento en el mismo componente pero el comportamiento puede ser muy diferente.
- Se utiliza un comportamiento o representación que se modifica conforme se usa el componente.
- A menudo quiere tener la capacidad de adaptar o reutilizar cierto componente bajo distintas circunstancias con un mínimo de trabajo extra.

MVC ofrece una alternativa elegante. Define un elemento complejo en términos de tres subunidades lógicas:

> **Modelo**: el estado del elemento; se ocupa de los cambios de estado.
> **Vista**: la representación del elemento (visual o no visual).
> **Controlador**: controla la funcionalidad del elemento, asignando las acciones a la vista según se impacto en el modelo.

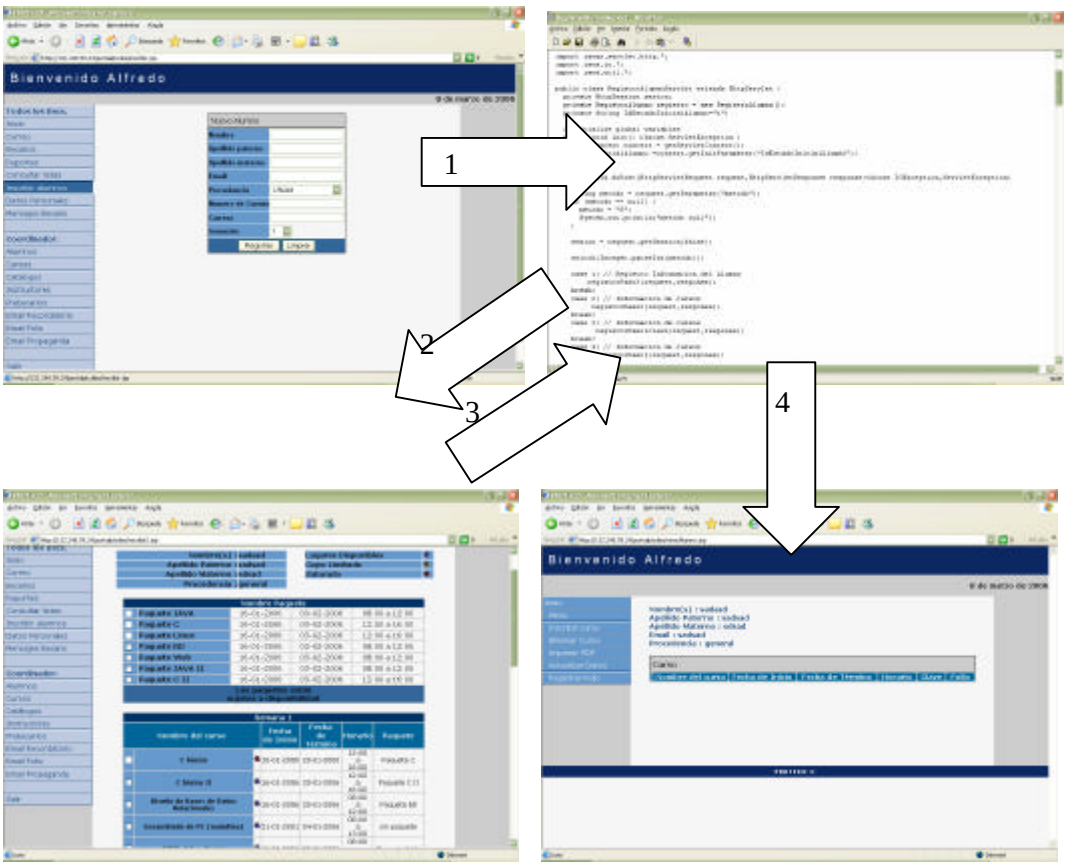
Se desarrollaron las vistas con tecnología Java en este caso JSP (Java Server Page) por su facilidad de embeber código HTML con el cual son creadas las páginas Web dinámicamente y se utilizaron hojas de estilo (o simplemente CSS -Cascade Style Sheet-) para mantener un esquema homogéneo en las páginas.

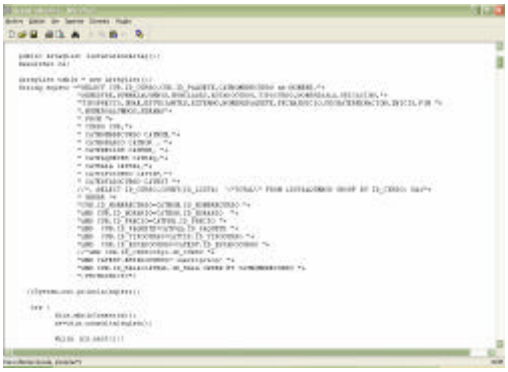
Como controlador se utilizo el API de Servlets que proporciona Java Enterprise Edition (o simplemente J2EE), con los Servlets se realizaron los cambios en el modelo, además de manejar el flujo de la aplicación, es decir, la lógica de negocio. El Servlet es el encargado de procesar las peticiones creando cualquier Java Bean u objeto usado por el JSP, así como decidiendo de las acciones de los usuarios a que JSP reenviar las peticiones. Cabe mencionar que no hay dentro del JSP lógica que procese las peticiones; el JSP simplemente es responsable de recuperar cualquier objeto o Java Bean que fue creado previamente por el Servlet y extrayendo el contenido dinámico del Servlet para insertarlo dentro de la plantilla estática.

Para la parte del modelo se utilizaron los Java Beans, específicamente beans de entidad, que permiten mantener el estado de un objeto en sus atributos y utilizando métodos de acceso se modifica el estado del objeto. Existe por lo tanto un bean para la entidad alumno, para el curso, para el becario, etc.

En general este enfoque da como resultado una separación más limpia de la presentación del contenido, produciendo una clara delimitación entre los roles y responsabilidades de los desarrolladores y los diseñadores de páginas Web dentro del equipo de programación.

A continuación veremos un ejemplo de la programación que se realizo para el sistema de PROTECO.

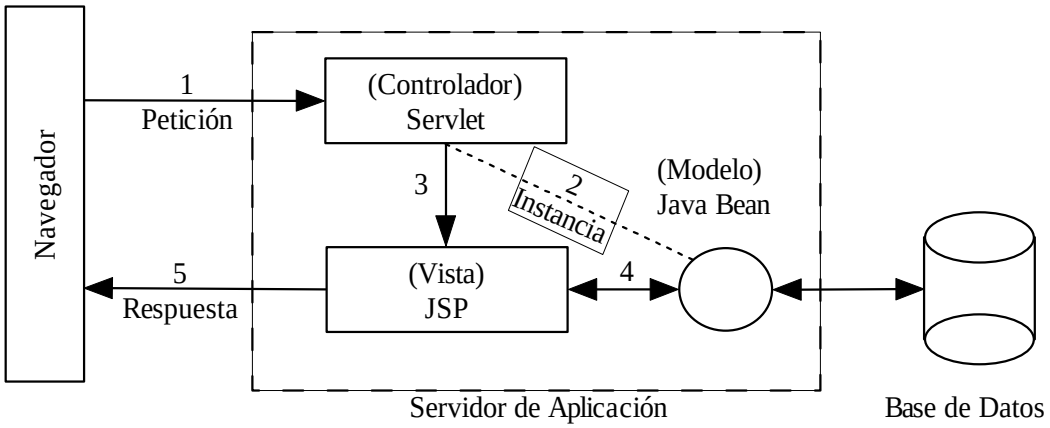




Clase que contiene las conexiones a la base de datos

El proceso de inscripción de alumnos sigue estos pasos:

1. Acceder al JSP inscribir.jsp.
2. Al llenar la información del JSP anterior la información se envía al Servlet RegistroAlumnoServlet, el cuál utiliza métodos de la clase RegistroAlumno. El Servlet dispara el siguiente JSP dependiendo de que la información sea valida o no.
3. Se dispara el JSP inscribir2.jsp con la información de los cursos que el Servlet anterior proporcionó. Al llenar la información correspondiente, esta es enviada al Servlet RegistroAlumnoServlet.
4. El servlet RegistroAlumnoServlet dispara al JSP al menuAlumno.jsp, el cual despliega la información del manejo de la inscripción.



5.1.3. Codificación

a) nuevoAlumno.jsp

El siguiente código es un ejemplo de JSP como los que se utilizaron en todas las páginas Web del sistema, en este caso en particular se tiene un formulario para capturar los datos del alumno como nombre, apellidos, número de cuenta, etc.

```
<%@ page contentType="text/html; language="java" import="java.util.*,portalproteco.*" %>

...

<form action="registroalumnoservlet" method="post" >
  <%
  BeanAlumno alumno = null;
  alumno=(BeanAlumno)session.getAttribute("alumno");
  Boolean  correccion = (Boolean)session.getAttribute("correccion");
  %>
 <center>
  <table border="1" align="center" class="formulario">
      <caption>
      Nuevo Alumno
```

```
      </caption>
      <tfoot>
        <tr>
          <td colspan="2"><div align="center">
              <input name="Envia" type="submit" class="boton" value="Registrar">
              <input name="Borra" type="reset" class="boton" onclick="ver()"
value="Limpiar" >
            </div></td>
        </tr>
      </tfoot>
      <tbody>
        <tr>
          <th scope="row">Nombre</th>
          <td><input type="text" name="nombre" value="<%=alumno.getNombre()%>"
maxlength="30"></td>
        </tr>
        <tr>
          <th scope="row"> Apellido paterno</th>
          <td><input type="text" name="apaterno" value="<%=alumno.getApaterno()%>"
maxlength="30" ></td>
        </tr>
        <tr>
          <th scope="row"> Apellido materno</th>
          <td><input type="text" name="amaterno" value="<%=alumno.getAmaterno()%>"
maxlength="30"></td>
        </tr>
        <tr>
          <th scope="row"> Email</th>
          <td><input type="text" name="email" value="<%=alumno.getEmail()%>"
maxlength="40"></td>
        </tr>
        <tr>
          <th scope="row"> Procedencia</th>
          <td><select name="procedencia" id="procedencia" onblur="escoger()" >
              <option value="unam">UNAM</option>
              <option value="externos">Estudiante Externo</option>
              <option value="general">P&uacute;blico en general</option>
            </select></td>
        </tr>
        <tr>
          <th scope="row"> Numero de Cuenta</th>
          <td><input type="text" name="numcuenta" id="numcuenta2"
value="<%=alumno.getNumeroCuenta()%>" maxlength="10"></td>
        </tr>
        <tr>
          <th scope="row"> Carrera</th>
          <td><input type="text" name="carrera" value="<%=alumno.getCarrera()%>"
maxlength="40"></td>
        </tr>
        <tr>
          <th scope="row">Semestre</th>
          <td align="left"><select name="semestre">
              <option value="1">1</option>
              <option value="2">2</option>
              <option value="3">3</option>
              <option value="4">4</option>
              <option value="5">5</option>
              <option value="6">6</option>
              <option value="7">7</option>
              <option value="8">8</option>
              <option value="9">9</option>
              <option value="10">10</option>
            </select></td>
        </tr>
        <tr>
          <th scope="row">Nombre de Usuario</th>
          <td><input name="login" type="text"  value="<%=alumno.getLogin()%>"
id="login"></td>
        </tr>
        <tr>
          <th scope="row">Contrase&ntilde;a</th>
          <td><input name="password1" type="passsword" id="password1"></td>
        </tr>
        <tr>
          <th scope="row">Confirmar contrase&ntilde;a</th>
          <td><input name="password2" type="passsword" id="password2"></td>
```

```
        </tr>
      </tbody>
    </table>
</center>

  <p>
    <input name="metodo" type="hidden" id="metodo" value="4">
  </p>
  </form>
```

b) RegistroAlumnoServlet.java

```
package portalproteco;

/**
 * <p>Title: Portal Proteco</p>
 * <p>Description: Sitio WEB para becarios de PROTECO</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: UNAM</p>
 * @author Julio Cesar Saynez Fabian, Alfredo Benavides Martinez
 * @version 1.0
 */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class RegistroAlumnoServlet extends HttpServlet {
  private HttpSession sesion;
  private RegistroAlumno registro = new RegistroAlumno();
  private String IdEstadoInicialAlumno="1";
  private static ArrayList cursosArray = null;
  private static ArrayList paquetesArray = null;
  private Boolean insAbierta;
  private int estadoInscripcion=0;

  private String urlsBecario[] = {"./inscribir.jsp",
                                  "./inscribir2.jsp","./menuAlumno.jsp"};

  private String urlsGeneral[] = {"./inscripcion/inscribir.jsp",
                       "./inscripcion/inscribir2.jsp","./terminarregistroservlet"};

  private String urls[];

  //Initialize global variables
  public void init() throws ServletException {
  ServletContext context = getServletContext();
  IdEstadoInicialAlumno =context.getInitParameter("IdEstadoInicialAlumno");
  paquetesArray = registro.listaPaquetesArray();
  }

  public  void  doPost(HttpServletRequest  request,HttpServletResponse  response)throws
IOException,ServletException{
    try{
    String metodo = request.getParameter("metodo");
    if (metodo == null) {
      metodo = "0";
      System.out.println("metodo null");
    }

    sesion = request.getSession(false);

    insAbierta = (Boolean)sesion.getAttribute("insAbierta");
    if(insAbierta.booleanValue()){
    estadoInscripcion=2;
    urls=urlsGeneral;
    }else{
    estadoInscripcion=1;
     urls=urlsBecario;
    }

    switch(Integer.parseInt(metodo)){

    case 1: // Registro Informacion del Alumno
```

```
        registroPaso3(request,response);
    break;
    case 2: // Informacion de cursos
         registroPaso2(request,response);
    break;
    case 3: // Informacion de cursos
          registroPaso1Clave(request,response);
    break;
    case 4: // Informacion de cursos
          registroPaso1(request,response);
    break;
    default:
      System.out.println("Default RegAluSer");
      response.sendRedirect("./salirservlet");
    break;
 }
   }
  catch (NullPointerException ex)
  {
  ex.printStackTrace();
  response.sendRedirect("./salirservlet");
  }
 }


 void registroPaso1(HttpServletRequest request,HttpServletResponse response)
                  throws IOException,ServletException,NullPointerException{

      System.out.println("Método 1 RegistroAlumnoServlet");

      if(leerDatosAlumno(request,response)){
      BeanAlumno alumno = (BeanAlumno)sesion.getAttribute("alumno");
       if(!(registro.buscarLogin(alumno))){
        cargarDatos(request,response);
        sesion.setAttribute("generarPDF",new Boolean(false));
        response.sendRedirect(urls[1]);
       }else{
         sesion.setAttribute("msg","Cambiar el Nombre de Usuario");
         alumno.setLogin("");
         sesion.setAttribute("alumno",alumno);
         sesion.setAttribute("correccion",new Boolean(true));
         response.sendRedirect(urls[0]);

       }

      }else{
      sesion.setAttribute("correccion",new Boolean(true));
      response.sendRedirect(urls[0]);
      }

  }


   void registroPaso2(HttpServletRequest request,HttpServletResponse response)
        throws IOException,ServletException,NullPointerException {

      System.out.println("Método 2 RegistroAlumnoServlet");

      Boolean generarPDF=(Boolean)sesion.getAttribute("generarPDF");

      if(generarPDF.booleanValue()==false && leerDatosCurso(request,response)){
          sesion.setAttribute("generarPDF",new Boolean(true));
          ArrayList cursosIns=null;
          BeanAlumno alumno = (BeanAlumno)sesion.getAttribute("alumno");

          int cursosSelecionados[] = (int[])sesion.getAttribute("cursosSeleccionados");
          int                          paquetesSeleccionados[]                          =
(int[])sesion.getAttribute("paquetesSeleccionados");

          // Generar nuevo Registro


          registro.insertarAlumno(alumno,cursosArray,cursosSelecionados,
                                  paquetesArray,paquetesSeleccionados,
                                  estadoInscripcion);
```

```
          if(alumno.getId_alumno()==-1){
          alumno=registro.buscarDatosAlumnoBD(alumno);
          sesion.setAttribute("alumno",alumno);
          }

          cursosIns = registro.cursosIncritosAlumno(alumno);
          sesion.setAttribute("cursosIns",cursosIns);


          sesion.setAttribute("generarPDF",new Boolean(true));
          sesion.removeAttribute("correccion");
          response.sendRedirect(urls[2]);
      }else{
            cargarDatos(request,response);
            sesion.setAttribute("generarPDF",new Boolean(false));
            response.sendRedirect(urls[1]);

      }

    }

void registroPaso3(HttpServletRequest request,HttpServletResponse response)
 throws IOException,ServletException,NullPointerException{
      System.out.println("Método 3 RegistroAlumnoServlet");
      cargarDatos(request,response);
      sesion.setAttribute("generarPDF",new Boolean(false));
      response.sendRedirect(urls[1]);
  }


  boolean leerDatosAlumno(HttpServletRequest request,HttpServletResponse response)throws
NullPointerException{
  boolean resultado=true;

  String nombre = request.getParameter("nombre");
    if (nombre == null) {
      nombre = "";

    }
    String apaterno = request.getParameter("apaterno");
    if (apaterno == null) {
      apaterno = "";

    }
    String amaterno = request.getParameter("amaterno");
    if (amaterno == null) {
      amaterno = "";

    }
    String email = request.getParameter("email");
    if (email == null) {
      email = "";

    }
    String procedencia = request.getParameter("procedencia");
    if (procedencia == null) {
      procedencia = "";

    }

    String numCuenta = request.getParameter("numcuenta");
    if (numCuenta == null) {
      numCuenta = "";
    }
    String carrera = request.getParameter("carrera");
    if (carrera == null) {
      carrera = "";
    }
    String semestre = request.getParameter("semestre");
    if (semestre == null) {
      semestre = "";
    }
    String telefono = request.getParameter("telefono");
    if (telefono == null) {
      telefono = "";
```

```
    }
    String login = request.getParameter("login");
    if (login == null) {
      login = "";
    }
   String password1 = request.getParameter("password1");
    if (password1 == null) {
      password1 = "";
    }
       String password2 = request.getParameter("password2");
    if (password2 == null) {
      password2 = "";
    }


    if(password1.length()<6){
      sesion.setAttribute("msg","La contraseña tiene que tener <br/> por lo menos 6
digitos");
      resultado=false;
    }
    if(password1.compareTo(password2)!=0){
      sesion.setAttribute("msg","La contraseña no coincide");
      resultado=false;
    }
    if(nombre.compareTo("")==0 || apaterno.compareTo("")==0 ||
    amaterno.compareTo("")==0 || email.compareTo("")==0 ||
    login.compareTo("")==0 || password1.compareTo("")==0 ||
    password2.compareTo("")==0 ){
      sesion.setAttribute("msg","No dejar campos vacios");
      resultado=false;
    }


    BeanAlumno alumno = new BeanAlumno();
    alumno.setNombre(nombre);
    alumno.setApaterno(apaterno);
    alumno.setAmaterno(amaterno);
    alumno.setId_estado(IdEstadoInicialAlumno);
    alumno.setEmail(email);
    alumno.setTelefono(telefono);
    alumno.setProcedencia(procedencia);
    alumno.setNumeroCuenta(numCuenta);
    alumno.setCarrera(carrera);
    alumno.setSemestre(semestre);
    alumno.setLogin(login);
    alumno.setPassword(password1);
    sesion.setAttribute("alumno",alumno);

  return resultado;
  }

  boolean leerDatosCurso(HttpServletRequest request,HttpServletResponse response) throws
NullPointerException{
  String[] cursosStr = request.getParameterValues("curso");
  String[] paquetesStr = request.getParameterValues("paquete");
  int nCursos;
  int nPaquetes;
  boolean resultado=false;


  if(cursosStr==null){
  nCursos=0;
  }
  else{ nCursos=cursosStr.length; }

  if(paquetesStr==null){
  nPaquetes=0;
  }else{
  nPaquetes=paquetesStr.length;
  }

  if(nCursos==0&&nPaquetes==0){
  return resultado;
  }
  else{
  resultado=true;
```

```
  }

  int[] cursos = new int[nCursos];
  int[] paquetes = new int[nPaquetes];

        for(int i=0; i<nCursos; i++) {
        cursos[i]=Integer.parseInt(cursosStr[i]);
        System.out.println("c-"+cursos[i]);
        }

        for(int i=0; i<nPaquetes; i++) {
          paquetes[i]=Integer.parseInt(paquetesStr[i]);
          System.out.println("p-"+paquetes[i]);
        }
    sesion.setAttribute("cursosSeleccionados",cursos);
    sesion.setAttribute("paquetesSeleccionados",paquetes);

    return resultado;
  }


  void   cargarDatos(HttpServletRequest   request,HttpServletResponse   response)   throws
NullPointerException{
      cursosArray = registro.listaCursosArray();

      sesion.setAttribute("cursosArray",cursosArray);
      sesion.setAttribute("paquetesArray",paquetesArray);


      BeanAlumno alumno = (BeanAlumno)sesion.getAttribute("alumno");

      if(registro.buscarIdAlumnoBD(alumno)!=-1){
          sesion.setAttribute("cursosIncritos",registro.listaCursosIncrito(alumno));
          sesion.setAttribute("paquetesIncritos",registro.listaPaquetesIncrito(alumno));
      }else{
          int vacio[]={-1};
          sesion.setAttribute("cursosIncritos",vacio);
          sesion.setAttribute("paquetesIncritos",vacio);
       }
  }

    void descargarDatos(HttpServletRequest request,HttpServletResponse response) throws
NullPointerException{
      sesion.removeAttribute("cursosArray");
      sesion.removeAttribute("paquetesArray");
      sesion.removeAttribute("cursosIncritos");
      sesion.removeAttribute("paquetesIncritos");
      sesion.removeAttribute("cursosSeleccionados");
      sesion.removeAttribute("paquetesSeleccionados");

  }

    //Clean up resources
  public void destroy() {
  }
}
```

c) RegistroAlumno.java

```
package portalproteco;

/**
 * <p>Title: Portal Proteco</p>
 * <p>Description: Sitio WEB para becarios de PROTECO</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: UNAM</p>
 * @author Julio Cesar Saynez Fabian, Alfredo Benavides Martinez
 * @version 1.0
 */

import java.sql.*;
import java.util.*;

public class RegistroAlumno  extends ConexionBD {
```

```
  public ArrayList listaCursosArray(){
  ResultSet rs;

  ArrayList tabla = new ArrayList();
  String sqlstr ="SELECT CUR.ID_CURSO,CUR.ID_PAQUETE,CATNOMBRECURSO as NOMBRE,"+

"SEMESTRE,NUMMAXALUMNOS,NUMCLASES,ESTADOCURSO,TIPOCURSO,NOMBRESALA,UBICACION,"+

"TIPOPRECIO,UNAM,ESTUDIANTES,EXTERNO,NOMBREPAQUETE,FECHAINICIO,FECHATERMINACION,INICIO,F
IN "+
                 ",NUMEROALUMNOS,SEMANA"+
                 " FROM "+
                 " CURSO CUR,"+
                 " CATNOMBRECURSO CATNOM,"+
                 " CATHORARIO CATHOR , "+
                 " CATPRECIOS CATPRE, "+
                 " CATPAQUETES CATPAQ,"+
                 " CATSALA CATSAL,"+
                 " CATTIPOCURSO CATTIP,"+
                 " CATESTADOCURSO CATEST "+
                 //",(SELECT ID_CURSO,COUNT(ID_LISTA)  \"TOTAL\" FROM LISTAALUMNOS GROUP
BY ID_CURSO) SQ1"+
                 " WHERE "+
                 "CUR.ID_NOMBRECURSO=CATNOM.ID_NOMBRECURSO "+
                 "AND CUR.ID_HORARIO=CATHOR.ID_HORARIO  "+
                 "AND CUR.ID_PRECIO=CATPRE.ID_PRECIO "+
                 "AND  CUR.ID_PAQUETE=CATPAQ.ID_PAQUETE "+
                 "AND  CUR.ID_TIPOCURSO=CATTIP.ID_TIPOCURSO "+
                 "AND  CUR.ID_ESTADOCURSO=CATEST.ID_ESTADOCURSO "+
                 "AND CATEST.ESTADOCURSO='inscripcion' "+
                 "AND CUR.ID_SALA=CATSAL.ID_SALA ORDER BY CATNOMBRECURSO "+
                 ",FECHAINICIO";


     try {
           this.abrirConexion();
           rs=this.consulta(sqlstr);

           while (rs.next()){

            BeanCursoPaquete miBean = new BeanCursoPaquete();

            miBean.setId_curso(rs.getInt(1));
            miBean.setId_paquete(rs.getInt(2));
            miBean.setNombre(rs.getString(3));
            miBean.setSemestre(rs.getString(4));
            miBean.setNumeroAlumnosMax(rs.getInt(5));
            miBean.setNumeroClases(rs.getInt(6));
            miBean.setEstado(rs.getString(7));
            miBean.setTipoCurso(rs.getString(8));
            miBean.setNombreSala(rs.getString(9));
            miBean.setUbicacion(rs.getString(10));
            miBean.setTipoPrecio(rs.getString(11));
            miBean.setPreUnam(rs.getInt(12));
            miBean.setPreEstudiante(rs.getInt(13));
            miBean.setPreExterno(rs.getInt(14));
            miBean.setPaquete(rs.getString(15));
            miBean.setFechaInicio(rs.getDate(16));
            miBean.setFechaTerminacion(rs.getDate(17));
            miBean.setHoraInicio(rs.getString(18));
            miBean.setHoraFin(rs.getString(19));
            miBean.setNumAlumnosRegistrados(rs.getInt(20));
            miBean.setSemana(rs.getInt(21));

          tabla.add(miBean);
        }
        this.cerrarConexion();
     }
     catch (Exception ex) {
       ex.printStackTrace();
     }

  return tabla;
  }
```

```
  public ArrayList listaPaquetesArray(){
    ResultSet rs;
    ArrayList tabla = new ArrayList();
    ArrayList cursos;

   String sqlstr ="SELECT ID_CURSO,CUR.ID_PAQUETE,NOMBREPAQUETE,SEMESTRE,FECHAINICIO,"+
                  "INICIO,FIN,UNAM,ESTUDIANTES,EXTERNO "+
                  "FROM "+
                  "CURSO CUR, CATHORARIO CATHOR, CATPRECIOS CATPRE, "+
                  "CATPAQUETES CATPAQ, CATESTADOCURSO CATEST "+
                  "WHERE "+
                  "CUR.ID_PAQUETE = CATPAQ.ID_PAQUETE "+
                  "AND  CUR.ID_HORARIO = CATHOR.ID_HORARIO "+
                  "AND  CATPAQ.ID_PRECIO = CATPRE.ID_PRECIO "+
                  "AND  CUR.SEMANA = 1 "+
                  "AND  CATPAQ.NOMBREPAQUETE <> 'sin paquete' "+
                  "AND  CUR.ID_ESTADOCURSO = CATEST.ID_ESTADOCURSO "+
                  "AND  CATEST.ESTADOCURSO = 'inscripcion' ";


 String sqlstr2="SELECT ID_CURSO,FECHATERMINACION FROM CURSO CUR, "+
                "CATESTADOCURSO CATEST  WHERE  "+
                " CATEST.ESTADOCURSO = 'inscripcion'  "+
                " AND  CUR.ID_ESTADOCURSO = CATEST.ID_ESTADOCURSO "+
                " AND ID_PAQUETE= ";

  try {
   this.abrirConexion();

   rs=this.consulta(sqlstr);
   while (rs.next()){

     BeanCursoPaquete miBean = new BeanCursoPaquete();

      miBean.setId_paquete(rs.getInt(2));
      miBean.setPaquete(rs.getString(3));
      miBean.setNombre(rs.getString(3));
      miBean.setSemestre(rs.getString(4));
      miBean.setFechaInicio(rs.getDate(5));
      miBean.setHoraInicio(rs.getString(6));
      miBean.setHoraFin(rs.getString(7));
      miBean.setPreUnam(rs.getInt(8));
      miBean.setPreEstudiante(rs.getInt(9));
      miBean.setPreExterno(rs.getInt(10));

   tabla.add(miBean);
   }

    BeanCursoPaquete miBean2;
   for(int k=0;k<tabla.size();k++){
       miBean2=(BeanCursoPaquete)tabla.get(k);
       rs=this.consulta(sqlstr2+"'"+miBean2.getId_paquete()+"'          ORDER          BY
FECHATERMINACION");
       cursos = new ArrayList();
           while (rs.next()){
           cursos.add(new Integer(rs.getInt(1)));
            miBean2.setFechaTerminacion(rs.getDate(2));
           }
          miBean2.setCursos(cursos);
   }
    this.cerrarConexion();

  return tabla;
  }
  catch (Exception ex) {
   ex.printStackTrace();
  return null;
  }

  }

   int insertarDatosAlumnoBD(BeanAlumno datos) {
    try {
       CallableStatement cs;
       cs = conexion.prepareCall("{call addalumno(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}");
```

```
        // Set the value for the IN parameter
        cs.setString(1,datos.getNombre());
        cs.setString(2,datos.getApaterno());
        cs.setString(3,datos.getAmaterno());
        cs.setString(4,datos.getId_estado());
        cs.setString(5,datos.getEmail());
        cs.setString(6,datos.getTelefono());
        cs.setString(7,datos.getLogin());
        cs.setString(8,datos.getPassword());
        cs.setString(9,datos.getProcedencia());
        cs.setString(10,datos.getNumeroCuenta());
        cs.setString(11,datos.getCarrera());
        cs.setString(12,datos.getSemestre());
        cs.setInt(13,0);
        // Register the type of the OUT parameter
        cs.registerOutParameter(14, Types.INTEGER);
        // Execute the stored procedure and retrieve the OUT value
        cs.execute();

        int idAlumno = cs.getInt(14);     // OUT parameter
        cs.close();

        System.out.println("idAlumno : "+idAlumno);
        return idAlumno;

        }
        catch (SQLException ex) {
        System.out.println("Error al insertar Alumno");
        ex.printStackTrace();
        return -1;
        }
      }

   int  insertarAlumnoCursoBD(int  idalumno,int  idcurso,int  idclave,int  idpaquete,int
estadoInscripcion) {
   try {

      System.out.println("insertarAlumnoCursoBD");
      CallableStatement cs;
      // Call a procedure with one OUT parameter

      cs = conexion.prepareCall("{ call addalumnocurso(?,?,?,?,?,?) }");

        // Set the value for the IN parameter
        cs.setInt(1,idalumno);
        cs.setInt(2,idcurso);
        cs.setInt(3,idclave);
        cs.setInt(4,idpaquete);
        cs.setInt(5,estadoInscripcion);

        // Register the type of the OUT parameter
        cs.registerOutParameter(6, Types.INTEGER);
        // Execute the stored procedure and retrieve the OUT value

        cs.execute();

        int idLista = cs.getInt(6);     // OUT parameter
        cs.close();

        System.out.println("idLista : "+idLista);
        return idLista;
        }
        catch (Exception ex) {
          //ex.printStackTrace();
          System.out.println(ex.getMessage());

          return -1;
        }

  }

  int insertarClaveRegistro(int idalumno, int idcurso){
   try {
   String str = "";
    CallableStatement cs;
       // Call a procedure with one OUT parameter
```

```
        str = this.crearCadena(12);

        cs = conexion.prepareCall("{call ADDCLAVEREGISTRO(?,?,?,?)}");
        cs.setInt(1,idalumno);
        cs.setInt(2,idcurso);
        cs.setString(3,str);
        cs.registerOutParameter(4, Types.INTEGER);
        // Execute the stored procedure and retrieve the OUT value
        cs.execute();
        int idClaveRegistro = cs.getInt(4);     // OUT parameter
        cs.close();


        System.out.println("idClaveRegistro : "+idClaveRegistro);
        return idClaveRegistro;
  }
  catch (Exception ex) {
 ex.printStackTrace();
  return 1;
  }
  }


ArrayList insertarAlumno(BeanAlumno datosAlumno,ArrayList cursosArray,
                        int cursos[],ArrayList paquetesArray,int paquetes[],
                        int estadoInscripcion) {

  try {

   Arrays.sort(cursos);
   Arrays.sort(paquetes);


   int idcurso=0;
   int idpaquete=0;
   int idalumno = 0;
   int cursosInscrito=0;
   int idclavereg=0;
   BeanCursoPaquete miBean = new BeanCursoPaquete();
   ArrayList cursosInscritos = new ArrayList();

   abrirConexion();
    // Insertar Alumno en la BD.
   idalumno = this.insertarDatosAlumnoBD(datosAlumno);

   //if(idalumno==-1){ return null; }// if

   for(int icurso=0;icurso<cursosArray.size();icurso++){
    miBean = (BeanCursoPaquete)cursosArray.get(icurso);
    idcurso = miBean.getId_curso();
    idpaquete = miBean.getId_paquete();

   if(                     Arrays.binarySearch(cursos,idcurso)>-1                     &&
Arrays.binarySearch(paquetes,idpaquete)<0){
        idclavereg = this.insertarClaveRegistro(idalumno,idcurso);
        this.insertarAlumnoCursoBD(idalumno,idcurso,idclavereg,-1,estadoInscripcion);
        miBean.setClaveRegistro(idclavereg);
        cursosInscritos.add(miBean);
     } // if
   } // for

   for(int ipaq=0;ipaq<paquetesArray.size();ipaq++){
    miBean = (BeanCursoPaquete)paquetesArray.get(ipaq);
    idpaquete = miBean.getId_paquete();

    if(Arrays.binarySearch(paquetes,idpaquete)>-1){



       for(int k=0;k<miBean.getCursos().size();k++){
      idcurso = ((Integer)miBean.getCursos().get(k)).intValue();
      if(k==0){
      idclavereg=insertarClaveRegistro(idalumno,idcurso);
      }
```

```
        //  System.out.println("  idalumno  "+idalumno+"  idcurso  "+idcurso+"  idclave
"+idclavereg);

this.insertarAlumnoCursoBD(idalumno,idcurso,idclavereg,idpaquete,estadoInscripcion);
       miBean.setClaveRegistro(idclavereg);
       }

     cursosInscritos.add(miBean);
    }
   }

cerrarConexion();

return cursosInscritos;
}
catch (Exception ex) {
ex.printStackTrace();
return null;
}

}

public ArrayList cursosIncritosAlumno(BeanAlumno datosAlumno) {
  ArrayList cursosInscritos = new ArrayList();

  int idalumno = datosAlumno.getId_alumno();

  if(idalumno==-1){
  return cursosInscritos;
  }

  String sqlstr ="SELECT CUR.ID_CURSO,CUR.ID_PAQUETE,CATNOMBRECURSO as NOMBRE,"+

"SEMESTRE,NUMMAXALUMNOS,NUMCLASES,ESTADOCURSO,TIPOCURSO,NOMBRESALA,UBICACION,"+

"TIPOPRECIO,UNAM,ESTUDIANTES,EXTERNO,NOMBREPAQUETE,FECHAINICIO,FECHATERMINACION,INICIO,F
IN "+
                ",NUMEROALUMNOS,SEMANA,CLA.ID_CLAVEREGISTRO,CLA.FOLIO,VALIDACION"+
                " FROM "+
                " CURSO CUR,"+
                " CATNOMBRECURSO CATNOM,"+
                " CATHORARIO CATHOR , "+
                " CATPRECIOS CATPRE, "+
                " CATPAQUETES CATPAQ,"+
                " CATSALA CATSAL,"+
                " CATTIPOCURSO CATTIP,"+
                " CATESTADOCURSO CATEST, "+
                " LISTAALUMNOS LIS, "+
                " CLAVEREGISTRO CLA"+
                " WHERE "+
                "CUR.ID_NOMBRECURSO=CATNOM.ID_NOMBRECURSO "+
                "AND CUR.ID_HORARIO=CATHOR.ID_HORARIO  "+
                "AND CUR.ID_PRECIO=CATPRE.ID_PRECIO "+
                "AND CUR.ID_PAQUETE=CATPAQ.ID_PAQUETE "+
                "AND CUR.ID_TIPOCURSO=CATTIP.ID_TIPOCURSO "+
                "AND CUR.ID_ESTADOCURSO=CATEST.ID_ESTADOCURSO "+
                "AND LIS.ID_CURSO = CUR.ID_CURSO "+
                "AND LIS.ID_CLAVEREGISTRO = CLA.ID_CLAVEREGISTRO "+
                "AND LIS.PAQUETE ='-1' "+
                "AND LIS.ID_ALUMNO='"+idalumno+"' "+
                "AND CUR.ID_SALA=CATSAL.ID_SALA ORDER BY FECHAINICIO";


     try {
           ResultSet rs;
           BeanCursoPaquete miBean;
           ArrayList cursos;

           this.abrirConexion();
           rs=this.consulta(sqlstr);

           while (rs.next()){

            miBean = new BeanCursoPaquete();

            miBean.setId_curso(rs.getInt(1));
```

```
          miBean.setId_paquete(rs.getInt(2));
          miBean.setNombre(rs.getString(3));
          miBean.setSemestre(rs.getString(4));
          miBean.setNumeroAlumnosMax(rs.getInt(5));
          miBean.setNumeroClases(rs.getInt(6));
          miBean.setEstado(rs.getString(7));
          miBean.setTipoCurso(rs.getString(8));
          miBean.setNombreSala(rs.getString(9));
          miBean.setUbicacion(rs.getString(10));
          miBean.setTipoPrecio(rs.getString(11));
          miBean.setPreUnam(rs.getInt(12));
          miBean.setPreEstudiante(rs.getInt(13));
          miBean.setPreExterno(rs.getInt(14));
          miBean.setPaquete(rs.getString(15));
          miBean.setFechaInicio(rs.getDate(16));
          miBean.setFechaTerminacion(rs.getDate(17));
          miBean.setHoraInicio(rs.getString(18));
          miBean.setHoraFin(rs.getString(19));
          miBean.setNumAlumnosRegistrados(rs.getInt(20));
          miBean.setSemana(rs.getInt(21));
          miBean.setClaveRegistro(rs.getInt(22));
          miBean.setFolio(rs.getString(23)==null? "":rs.getString(23));
         miBean.setValidacion(rs.getString(24)==null? "":rs.getString(24));
        cursosInscritos.add(miBean);
      }


    sqlstr = "SELECT CUR.ID_CURSO,CUR.ID_PAQUETE,NOMBREPAQUETE,"+
                    "SEMESTRE,FECHAINICIO,"+
                    "INICIO,FIN,UNAM,ESTUDIANTES,EXTERNO,"+
                    "CLA.ID_CLAVEREGISTRO,CLA.FOLIO,CLA.VALIDACION"+
                    " FROM "+
                    " CURSO CUR,"+
                    " CATHORARIO CATHOR , "+
                    " CATPRECIOS CATPRE, "+
                    " CATPAQUETES CATPAQ,"+
                    " LISTAALUMNOS LIS, "+
                    " CLAVEREGISTRO CLA"+
                    " WHERE "+
                    " CUR.ID_PAQUETE=CATPAQ.ID_PAQUETE "+
                    " AND CUR.ID_HORARIO=CATHOR.ID_HORARIO "+
                    " AND CATPAQ.ID_PRECIO=CATPRE.ID_PRECIO "+
                    " AND LIS.ID_CURSO = CUR.ID_CURSO "+
                    " AND LIS.ID_CLAVEREGISTRO = CLA.ID_CLAVEREGISTRO "+
                    " AND CUR.SEMANA='1' "+
                    " AND LIS.PAQUETE<>'-1' "+
                    " AND LIS.ID_ALUMNO='"+idalumno+"' "+
                    " ORDER BY FECHAINICIO";


rs=this.consulta(sqlstr);
 while (rs.next()){

   miBean = new BeanCursoPaquete();

    miBean.setId_paquete(rs.getInt(2));
    miBean.setPaquete(rs.getString(3));
    miBean.setNombre(rs.getString(3));
    miBean.setSemestre(rs.getString(4));
    miBean.setFechaInicio(rs.getDate(5));
    miBean.setHoraInicio(rs.getString(6));
    miBean.setHoraFin(rs.getString(7));
    miBean.setPreUnam(rs.getInt(8));
    miBean.setPreEstudiante(rs.getInt(9));
    miBean.setPreExterno(rs.getInt(10));
    miBean.setClaveRegistro(rs.getInt(11));
    miBean.setFolio(rs.getString(12)==null? "":rs.getString(12));
    miBean.setValidacion(rs.getString(13)==null? "":rs.getString(13));


 cursosInscritos.add(miBean);
 }

  sqlstr="SELECT ID_CURSO,FECHATERMINACION FROM CURSO WHERE ID_PAQUETE=";
  BeanCursoPaquete miBean2;
```

```
    for(int k=0;k<cursosInscritos.size();k++){
        miBean2=(BeanCursoPaquete)cursosInscritos.get(k);
       rs=this.consulta(sqlstr+"'"+miBean2.getId_paquete()+"'          ORDER          BY
FECHATERMINACION");
       cursos = new ArrayList();
           while (rs.next()){
           cursos.add(new Integer(rs.getInt(1)));
            miBean2.setFechaTerminacion(rs.getDate(2));
           }
    miBean2.setCursos(cursos);
   }


this.cerrarConexion();
return cursosInscritos;
}
catch (Exception ex) {
ex.printStackTrace();
return null;
}
}


public int[] listaCursosIncrito(BeanAlumno alumno){
    ResultSet rs;
    int tabla[]={-1};


   String sqlstr = "SELECT LIS.ID_CURSO FROM ALUMNO ALU,LISTAALUMNOS LIS WHERE "+
                " NOMBREALUMNO='"+alumno.getNombre()+"'"+
                " AND APATERNO='"+alumno.getApaterno()+"'"+
                " AND AMATERNO='"+alumno.getAmaterno()+"'"+
                " AND EMAIL='"+alumno.getEmail()+"'"+
                " AND PROCEDENCIA='"+alumno.getProcedencia()+"'"+
                " AND ALU.ID_ALUMNO=LIS.ID_ALUMNO";

  try {
  this.abrirConexion();

  rs = consulta("SELECT COUNT(LIS.ID_CURSO) FROM ALUMNO ALU,LISTAALUMNOS LIS WHERE "+
             " NOMBREALUMNO='"+alumno.getNombre()+"'"+
                " AND APATERNO='"+alumno.getApaterno()+"'"+
                " AND AMATERNO='"+alumno.getAmaterno()+"'"+
                " AND EMAIL='"+alumno.getEmail()+"'"+
                " AND PROCEDENCIA='"+alumno.getProcedencia()+"'"+
                " AND ALU.ID_ALUMNO=LIS.ID_ALUMNO");



  rs.next();

  int lineas = rs.getInt(1);

  if(lineas==0){
   this.cerrarConexion();
    return tabla;
  }

  tabla= new int[lineas];
  int i=0;

   rs=consulta(sqlstr);
   while (rs.next()){
   tabla[i++]=rs.getInt(1);
   }
    this.cerrarConexion();
    Arrays.sort(tabla);
  }
  catch (Exception ex) {
   ex.printStackTrace();

  }
return tabla;
  }
```

```
   public int[] listaPaquetesIncrito(BeanAlumno alumno){
    ResultSet rs;
    int tabla[]={-1};


   String sqlstr = "SELECT DISTINCT PAQUETE FROM ALUMNO ALU,LISTAALUMNOS LIS WHERE "+
                     " NOMBREALUMNO='"+alumno.getNombre()+"'"+
                  " AND APATERNO='"+alumno.getApaterno()+"'"+
                  " AND AMATERNO='"+alumno.getAmaterno()+"'"+
                  " AND EMAIL='"+alumno.getEmail()+"'"+
                  " AND PROCEDENCIA='"+alumno.getProcedencia()+"'"+
                  " AND ALU.ID_ALUMNO=LIS.ID_ALUMNO"+
                  " AND LIS.PAQUETE<>'-1'";
  try {
  this.abrirConexion();

  rs = consulta("SELECT  DISTINCT COUNT(PAQUETE)  FROM ALUMNO ALU,LISTAALUMNOS LIS WHERE
"+
                  " NOMBREALUMNO='"+alumno.getNombre()+"'"+
                  " AND APATERNO='"+alumno.getApaterno()+"'"+
                  " AND AMATERNO='"+alumno.getAmaterno()+"'"+
                  " AND EMAIL='"+alumno.getEmail()+"'"+
                  " AND PROCEDENCIA='"+alumno.getProcedencia()+"'"+
                  " AND ALU.ID_ALUMNO=LIS.ID_ALUMNO"+
                  " AND LIS.PAQUETE<>'-1'");


  rs.next();

  int lineas = rs.getInt(1);

  if(lineas==0){
  this.cerrarConexion();
    return tabla;
  }

  tabla= new int[lineas];
  int i=0;

   rs=consulta(sqlstr);
   while (rs.next()){
   tabla[i++]=rs.getInt(1);
   }


    this.cerrarConexion();
  Arrays.sort(tabla);
  }
  catch (Exception ex) {
   ex.printStackTrace();

  }
  return tabla;
  }

…

}
```

d) BeanAlumno.java

```
package portalproteco;

/**
 * <p>Title: Portal Proteco</p>
 * <p>Description: Sitio WEB para becarios de PROTECO</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: UNAM</p>
 * @author Julio Cesar Saynez Fabian, Alfredo Benavides Martínez
 * @versión 1.0
 */


public class BeanAlumno {
```

```
public BeanAlumno() {
}
private int id_alumno=-1;
private String email="";
private String telefono="";
private String login="";
private String password="";
private String procedencia="";
private String numeroCuenta="";
private String carrera="";
private String estado="";
private String nombre="";
private String apaterno="";
private String amaterno="";
private String semestre="";
private String id_estado="";
private int cuentaActiva;

public int getId_alumno() {
  return id_alumno;
}
public void setId_alumno(int id_alumno) {
  this.id_alumno = id_alumno;
}
public void setEmail(String email) {
  this.email = email.toLowerCase().trim();
}
public String getEmail() {
  return email;
}
public void setTelefono(String telefono) {
  this.telefono = telefono.trim();
}
public String getTelefono() {
  return telefono;
}
public void setLogin(String login) {
  this.login = login.trim();
}
public String getLogin() {
  return login;
}
public void setPassword(String password) {
  this.password = password;
}
public String getPassword() {
  return password;
}
public void setProcedencia(String procedencia) {
  this.procedencia = procedencia.toLowerCase().trim();
}
public String getProcedencia() {
  return procedencia;
}
public void setNumeroCuenta(String numeroCuenta) {
  this.numeroCuenta = numeroCuenta.trim();
}
public String getNumeroCuenta() {
  return numeroCuenta;
}
public void setCarrera(String carrera) {
  this.carrera = carrera.toLowerCase().trim();
}
public String getCarrera() {
  return carrera;
}
public void setEstado(String estado) {
  this.estado = estado;
}
public String getEstado() {
  return estado;
}
public void setNombre(String nombre) {
  this.nombre = nombre.toLowerCase().trim();
}
```

```
  public String getNombre() {
    return nombre;
  }
  public void setApaterno(String apaterno) {
    this.apaterno = apaterno.toLowerCase().trim();
  }
  public String getApaterno() {
    return apaterno;
  }
  public void setAmaterno(String amaterno) {
    this.amaterno = amaterno.toLowerCase().trim();
  }
  public String getAmaterno() {
    return amaterno;
  }
  public void setSemestre(String semestre) {
    this.semestre = semestre.trim();
  }
  public String getSemestre() {
    return semestre;
  }
  public void setId_estado(String id_estado) {
    this.id_estado = id_estado;
  }
  public String getId_estado() {
    return id_estado;
  }
  public void setCuentaActiva(int cuentaActiva) {
    this.cuentaActiva = cuentaActiva;
  }
  public int getCuentaActiva() {
    return cuentaActiva;
  }
}
```

5.2. SQL

El Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Aún a características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

5.2.1. Orígenes y evolución

Los orígenes del SQL están ligados a los de las bases de datos relaciónales. En 1970 Codd propone el modelo relacional y asociado a este un sublenguaje de acceso a los datos basado en el cálculo de predicados. Basándose en estas ideas los laboratorios de IBM definen el lenguaje SEQUEL (Structured English QUEry Language) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo este una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO.

Sin embargo este primer estándar no cubre todas las necesidades de los desarrolladores e incluye funcionalidades de definición de almacenamiento que se consideraron suprimir. Así que en 1992 se lanza un nuevo estándar ampliado y revisado del SQL llamado SQL-92 o SQL2.

En la actualidad el SQL es el estándar de facto de la inmensa mayoría de los SGBD comerciales y aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es extensa, el soporte al estándar SQL-92 es general y muy amplio.

No obstante el éxito del SQL2 en estos momentos se trabaja en un nuevo estándar, el SQL3, que pretende dar soporte a las nuevas características derivadas de la generalización de los conceptos objeto-relacionales en las bases de datos.

5.2.2. Características generales

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos.

Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro.

5.2.3. Funcionalidad

El SQL proporciona una rica funcionalidad más allá de la simple consulta (o recuperación) de datos. Asume el papel de lenguaje de definición de datos (LDD), lenguaje de definición de vistas (LDV) y lenguaje de manipulación de datos (LMD). Además permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas.

Las primeras versiones del SQL incluían funciones propias de lenguaje de definición de almacenamiento (LDA) pero fueron suprimidas en los estándares más recientes con el fin de mantener el lenguaje sólo a nivel conceptual y externo.

5.2.4. Modos de uso

El SQL permite fundamentalmente dos modos de uso:

Un uso interactivo, destinado principalmente a los usuarios finales avanzados u ocasionales, en el que las diversas sentencias SQL se escriben y ejecutan en línea de comandos, o un entorno semejante.

Un uso integrado, destinado al uso por parte de los programadores dentro de programas escritos en cualquier lenguaje de programación anfitrión. En este caso el SQL asume el papel de sublenguaje de datos.

En el caso de hacer un uso embebido del lenguaje podemos utilizar dos técnicas alternativas de programación. En una de ellas, en la que el lenguaje se denomina SQL estático, las sentencias utilizadas no cambian durante la ejecución del programa. En la otra, donde el lenguaje recibe el nombre de SQL dinámico, se produce una modificación total o parcial de las sentencias en el transcurso de la ejecución del programa.

La utilización de SQL dinámico permite mayor flexibilidad y mayor complejidad en las sentencias, pero como contra punto obtenemos una eficiencia menor y el uso de técnicas de programación más complejas en el manejo de memoria y variables.

5.2.5. Optimización

Como ya se dijo arriba, y como suele ser común en los lenguajes de acceso a bases de datos de alto nivel, el SQL es un lenguaje declarativo. O sea, que especifica que es lo que se quiere y no como conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.

El orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que éste lleve a cabo una optimización antes de la ejecución de la misma.

Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa. Este lenguaje ha sido implementado a nivel experimental y está evolucionando rápidamente.

5.3. JDBC

El API JDBC (Java Database Connectivity) es una especificación de un conjunto de clases y métodos de operación que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea. La aplicación de Java debe tener acceso a un "driver" JDBC adecuado. Este "driver" es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real.

La necesidad de JDBC, a pesar de la existencia del API ODBC (Open DataBase Connectivity), viene dada porque ODBC es un interfaz escrito en lenguaje C, que al no ser un lenguaje portable, haría que las aplicaciones Java también perdiesen la portabilidad, además, ODBC tiene el inconveniente de que se ha de instalar manualmente en cada máquina; al contrario que los "drivers" JDBC, que al estar escritos en Java son automáticamente instalables, portables y seguros.

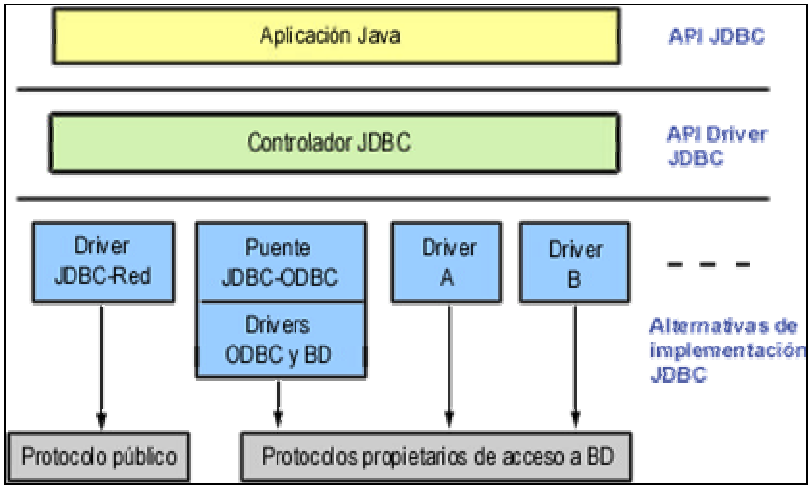


Figura 5-1 API JDBC

La conectividad de bases de datos de Java se basa en sentencias SQL, por lo que se hace imprescindible un conocimiento adecuado de SQL para realizar cualquier clase de operación de bases de datos. Aunque, afortunadamente, casi todos los entornos de desarrollo Java ofrecen componentes visuales que proporcionan una funcionalidad suficientemente potente sin necesidad de que sea necesario utilizar SQL, aunque para usar directamente el JDK se haga imprescindible. La especificación JDBC requiere que cualquier "driver" JDBC sea compatible con al menos el nivel «de entrada» de ANSI SQL 92 (*ANSI SQL 92 Entry Level*).

5.3.1. Tipos de drivers

Un "driver" JDBC puede pertenecer a una de cuatro categorías diferentes en cuanto a la forma de operar.

- Puente JDBC-ODBC

Sun inicialmente para popularizar JDBC y consistente en aprovechar todo lo existente, establecio un puente entre JDBC y ODBC. Este "driver" convierte todas las llamadas JDBC a llamadas ODBC y realiza la conversión correspondiente de los resultados.

La ventaja de este "driver", que se proporciona con el JDK, es que Java dispone de acceso inmediato a todas las fuentes posibles de bases de datos y no hay que hacer ninguna configuración adicional aparte de la ya existente. No obstante, tiene dos desventajas muy importantes; por un lado, la mayoría de los "drivers" ODBC a su vez convierten sus llamadas a llamadas a una librería nativa del fabricante DBMS, con lo cual la lentitud del "driver" JDBC-ODBC puede ser exasperante, al llevar dos capas adicionales que no añaden funcionalidad alguna; y por otra parte, el puente JDBC-ODBC requiere una instalación ODBC ya existente y configurada.

- Java/Binario

Este driver se salta la capa ODBC y habla directamente con la librería nativa del fabricante del sistema DBMS (como pudiera ser DB-Library para Microsoft SQL Server o CT-Lib para Sybase SQL Server). Este "driver" es 100% Java pero aún así necesita la existencia de un código binario (la librería DBMS) en la máquina del cliente, con las limitaciones y problemas que esto implica.

- 100% Java/Protocolo nativo

Es un "driver" realizado completamente en Java que se comunica con el servidor DBMS utilizando el protocolo de red nativo del servidor. De esta forma, el "driver" no necesita intermediarios para hablar con el servidor y convierte todas las peticiones JDBC en peticiones de red contra el servidor. La ventaja de este tipo de "driver" es que es una solución 100% Java y, por lo tanto, independiente de la máquina en la que se va a ejecutar el programa.

Igualmente, dependiendo de la forma en que esté programado el "driver", puede no necesitar ninguna clase de configuración por parte del usuario. La única desventaja de este tipo de drivers es que el cliente está ligado a un servidor DBMS concreto, ya que el protocolo de red que utiliza MS SQL Server por ejemplo no tiene nada que ver con el utilizado por DB2, PostGres u Oracle. La mayoría de los fabricantes de bases de datos han incorporado a sus propios "drivers" JDBC del segundo o tercer tipo, con la ventaja de que no suponen un coste adicional.

- 100% Java/Protocolo independiente

Esta es la opción más flexible, se trata de un "driver" 100% Java/Protocolo independiente, que requiere la presencia de un intermediario en el servidor. En este caso, el "driver" JDBC hace las peticiones de datos al intermediario en un protocolo de red independiente del servidor DBMS. El intermediario a su vez, que está ubicado en el lado del servidor, convierte las peticiones JDBC en peticiones nativas del sistema DBMS. La ventaja de este método es inmediata: el programa que se ejecuta en el cliente, y aparte de las ventajas de los "drivers" 100% Java, también presenta la independencia respecto al sistema de bases de datos que se encuentra en el servidor.

De esta forma, si una empresa distribuye una aplicación Java para que sus usuarios puedan acceder a su servidor MS SQL y posteriormente decide cambiar el servidor por Oracle, Postgres o DB2, no necesita volver a distribuir la aplicación, sino que únicamente debe reconfigurar la aplicación residente en el servidor que se encarga de transformar las peticiones de red en peticiones nativas. La única desventaja de este tipo de "drivers" es que la aplicación intermediaria es una aplicación independiente que suele tener un costo adicional por servidor físico, que hay que añadir al costo del servidor de bases de datos.

5.4. Formatos y estilo

5.4.1. JavaScript

Se utilizo este lenguaje para realizar algunas operaciones dinámicas inmersas en algunos JSP, así como para realizar validaciones de datos en un menor lapso de tiempo e implementar un diseño más agradable al usuario.

Historia

JavaScript es un lenguaje interpretado orientado a las páginas Web, con una sintaxis semejante a la del lenguaje Java.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que fabricó los primeros navegadores de Internet comerciales.

Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se venía utilizando en páginas Web HTML, para realizar tareas y operaciones en el marco de la aplicación cliente servidor. Con la irrupción de Web 2.0, JavaScript se ha convertido en un verdadero lenguaje de programación que aporta la potencia de cálculo del navegador para aumentar la usabilidad de aplicaciones Web con técnicas avanzadas como AJAX o JCC.

Los autores inicialmente lo llamaron Mocha y más tarde LiveScript pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995.

En 1997 los autores propusieron JavaScript para que fuera adoptado como estándar de "The European Computer Manufacturers Association ECMA", que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también lo fué como un estándar ISO.

JScript es la implementación de ECMAScript de Microsoft, muy similar al JavaScript de Netscape, pero con ciertas diferencias en el modelo de objetos del navegador que hacen a ambas versiones con frecuencia incompatibles.

Para evitar estas incompatibilidades, el World Wide Web Consortium diseñó el estándar Document Object Model (DOM, ó Modelo de Objetos del Documento), que incorporan

Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera versión 7, y Mozilla desde su primera versión.

Ejemplo:

```
// JavaScript Document
// vaCurso.js
selNombre= new Array();
selFechaDeInicio= new Array();
selHoraInicio= new Array();
selHoraTermino= new Array();
bandera=0;

function busca(nombre,fechaDeInicio,horaInicio,horaTermino,campo)
{

sizeSel=selNombre.length;
// Revisar el checkbox del formurario
if(campo.checked==true)
{
	// Revisar el numero de elementos del insertatos
	// si es cero se inserta como el primer elemento
	if(sizeSel==0)
	{
	selNombre[sizeSel]=nombre;
	selFechaDeInicio[sizeSel]=fechaDeInicio;
	selHoraInicio[sizeSel]=horaInicio;
	selHoraTermino[sizeSel]=horaTermino;
	}
	// sino es cero recorrer el arreglo en busca del elemento
	else
	{
	for(i=0;i<sizeSel;i++)
		{
	if((fechaDeInicio==selFechaDeInicio[i])&&((horaInicio>=selHoraInicio[i])
	&&(horaInicio<selHoraTermino[i])))
			{
			campo.checked=false;
			bandera=1;
			break;
			}
		}
		if(bandera==0)
		{
		selNombre[sizeSel]=nombre;
		selFechaDeInicio[sizeSel]=fechaDeInicio;
		selHoraInicio[sizeSel]=horaInicio;
		selHoraTermino[sizeSel]=horaTermino;
		}
		bandera=0;
	}
}
// sino esta seleccionado  cambio el valor dentro del arreglo
else
	{
	for(i=0;i<sizeSel;i++)
		{
		if(nombre==selNombre[i])
			{
			selNombre[i]="vacio";
			selFechaDeInicio[i]="vacio";
			selHoraInicio[i]=0;
			selHoraTermino[i]=0;
```

```
                    }
                }
        }
}
```

5.4.2. Hojas de estilo

Las Hojas de estilo se utilizaron para definir un diseño general del sistema, que fuese más amigable con el usuario en general.

Historia

Las hojas de estilo (style sheets) son conjuntos de instrucciones, a son comunes de localizar en forma de archivo anexo, que se asocian a los archivos de texto y se ocupan de los aspectos de formato y de presentación de los contenidos: tipo, fuente y tamaño de letras, justificación del texto, colores y fondos, etc. Las hojas de estilo permiten liberar la composición del texto de los aspectos visuales y favorecen que se estructure y anote mediante códigos que permiten un tratamiento más eficaz de los contenidos. El uso adecuado de las hojas de estilo es uno de los aspectos clave de la edición digital.

Las hojas de estilo son una herramienta de gran utilidad de los programas de tratamiento de textos, como OpenOffice o Microsoft Word. Asimismo, constituyen una parte esencial de los lenguajes de marcas para edición digital: LaTeX, XML y XHTML. Dos lenguajes de hojas de estilo son CSS y XSL.

Ejemplo:

```
/* CSS Document
   estilo.css
*/

.boton {
}
/*
      Valores de color y tipo de letra para las tablas del formulario
*/
.formulario{
  border:1px solid #000;
  border-collapse:collapse;
  font-family:arial,sans-serif;
  font-size:90%;

}
.formulario td{
  padding: 2px;
}
/*
      Semanas
*/
#semana{
background: #003366;
color: #FFF;
text-align:center;
}
```

```
/*
      encabezado de semana
*/
#encsem{
background: #006699;
color: #FFF;
text-align:center;
}

/*
      Celdas de los cursos
*/
#cursos{
font-size:12px;
}
```

6. Pruebas

6.1. Medidor de estrés del Servidor Web (Concurrencia)

La medición del rendimiento de un servidor Web es una característica que sólo se puede mejorar mediante la experiencia y la experimentación continua. Hay muchas variables en juego, como el número de clientes, la velocidad de las conexiones de clientes, los recursos de servidor, el código de la aplicación, etc.

Para realizar las pruebas de estrés (concurrencia) del sistema, se utilizo el software "Webserver Stress Tool" versión 7 de la empresa "Paessler" el cual simula un número de usuarios simultáneos solicitando información del servidor con los siguientes datos de configuración:

1. Tipo de prueba: correr la prueba con carga constante hasta que cada uno de los usuarios haya generado un número específico de pulsaciones.
2. Número de usuarios simultáneos: 10
3. Número del clics del usuario: 10
4. Pausa entre cada clic: 5 segundos

Nota: Estimando una carga de 10 usuarios pulsando una liga cada 5 segundos es aproximadamente 120 páginas vistas por minuto (aprox. 7,200 páginas por hora).

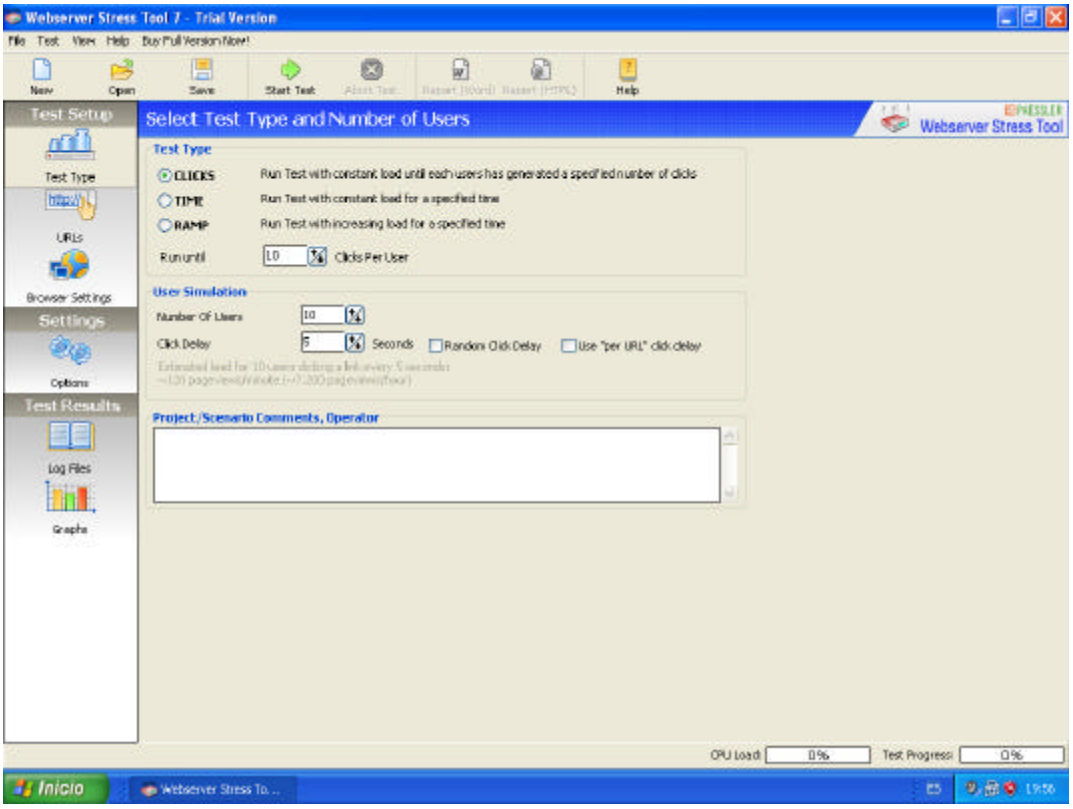


Figura 6-1 Configuración de la prueba de estrés.

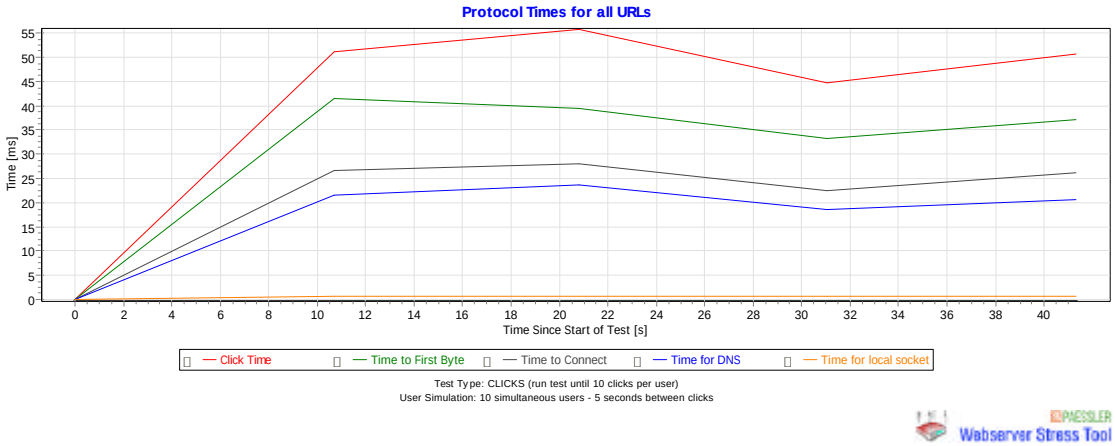


Figura 6-2 Tiempos de protocolo para cada URL.

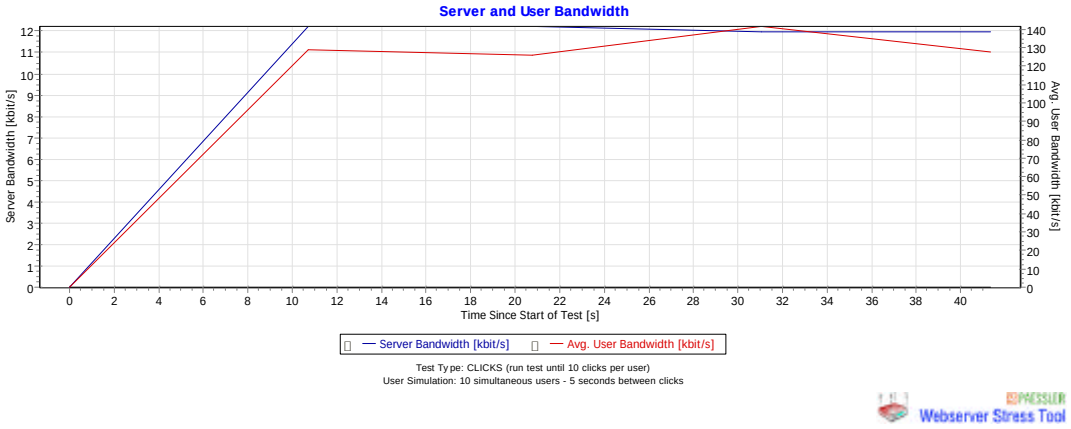


Figura 6-3 Ancho de banda del servidor y usuario.

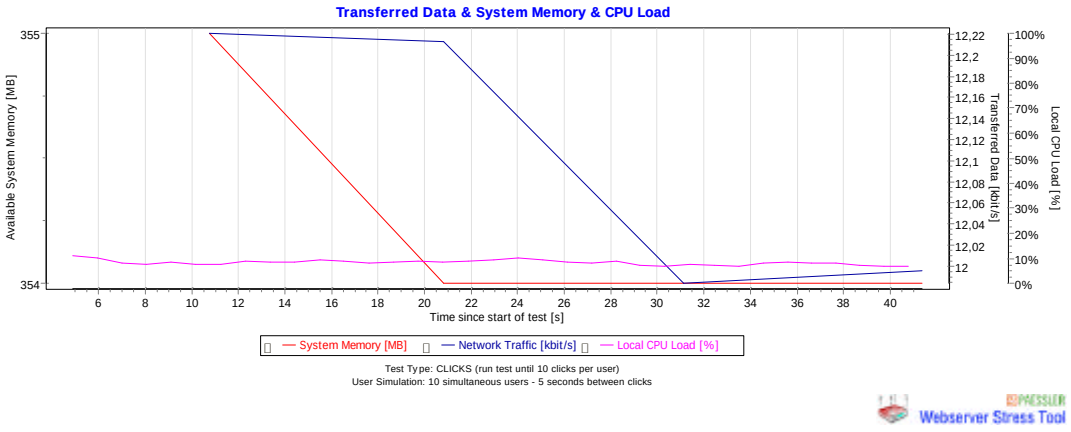


Figura 6-4 Tranferencia de datos, memoria del sistema y carga del CPU.

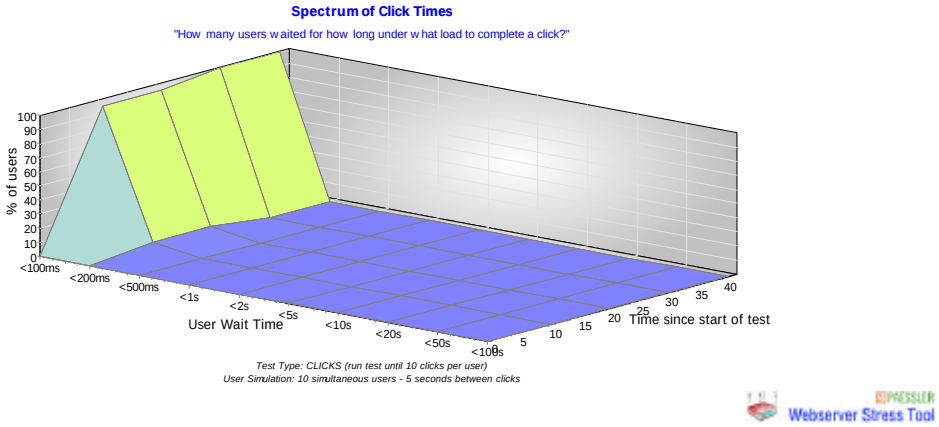


Figura 6-5 Tiempo de espera después de una petición.

¿Cuántos usuarios esperando por cuanto tiempo?

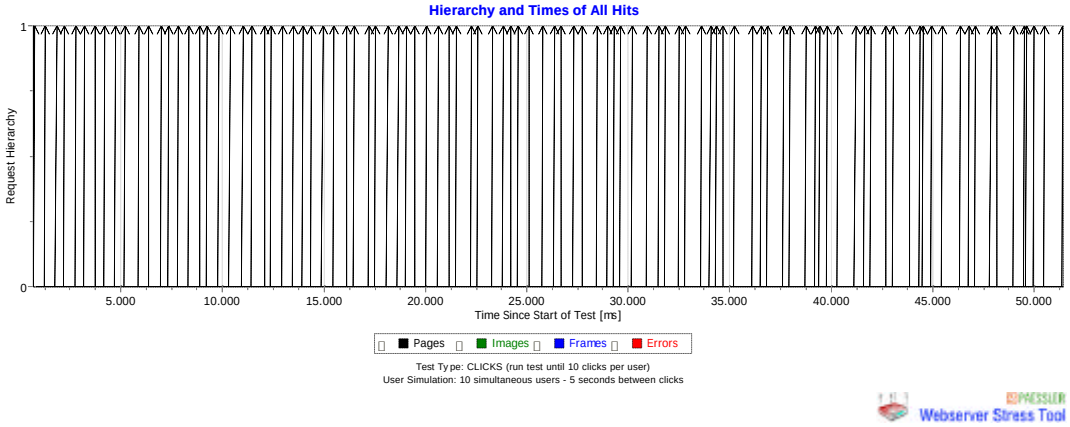


Figura 6-6 Instantes de las peticiones realizadas.

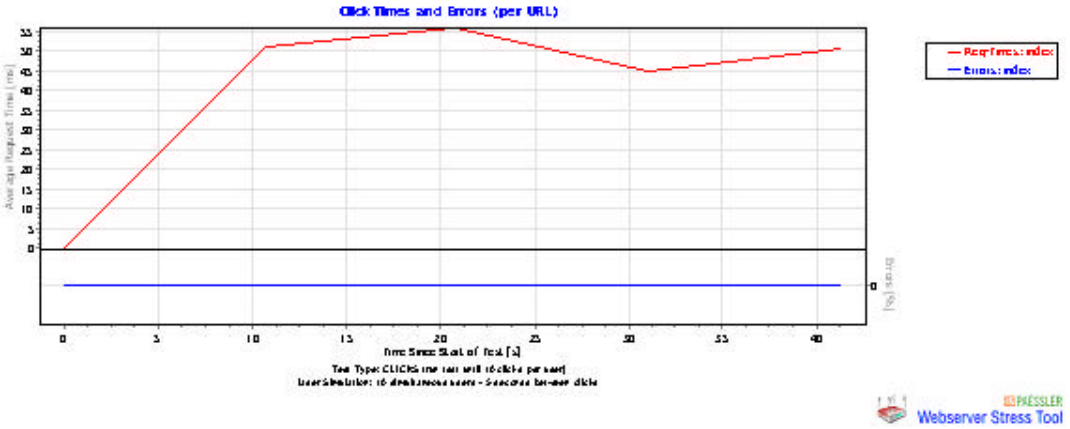


Figura 6-7 Tiempo de la petición y errores por cada URL.

Reporte

```
** Test Logfile by Webserver Stress Tool 7.1.1.233 Trial Version **
© 1998-2006 Paessler AG, http://www.paessler.com

Test run on 11/11/2006 20:40:14

** Project and Scenario Comments, Operator **

Results of period #1 (from 1 sec to 11 sec ):
************************************************************
Completed Clicks: 20 with 0 Errors (=0,00%)
Average Click Time for 10 Users: 122 ms
Successful clicks per Second: 2,00 (equals 7.195,34 Clicks per Hour)

Results of period #2 (from 11 sec to 21 sec ):
************************************************************
Completed Clicks: 20 with 0 Errors (=0,00%)
Average Click Time for 10 Users: 72 ms
Successful clicks per Second: 1,98 (equals 7.123,47 Clicks per Hour)

Results of period #3 (from 21 sec to 31 sec ):
************************************************************
Completed Clicks: 19 with 0 Errors (=0,00%)
Average Click Time for 10 Users: 84 ms
Successful clicks per Second: 1,89 (equals 6.799,05 Clicks per Hour)

Results of period #4 (from 31 sec to 42 sec ):
************************************************************
Completed Clicks: 20 with 0 Errors (=0,00%)
Average Click Time for 10 Users: 162 ms
Successful clicks per Second: 1,93 (equals 6.935,64 Clicks per Hour)

Results of period #5 (from 42 sec to 52 sec ):
************************************************************
Completed Clicks: 20 with 0 Errors (=0,00%)
Average Click Time for 10 Users: 86 ms
Successful clicks per Second: 1,97 (equals 7.100,92 Clicks per Hour)

Results of complete test
************************************************************

** Results per URL for complete test **

URL#1 (cursos): Average Click Time 106 ms, 99 Clicks, 0 Errors

Total Number of Clicks: 99 (0 Errors)
Average Click Time of all URLs: 106 ms
```

URLs a probar

URL#	Nombre	URL
1	Cursos	http://132.248.59.12/portal/registroalumnoservlet?metodo=5

Resultados por Usario

No. Usuario	Clics	Errores	Tiempo promedio entre clics [ms]	Bytes	Kbit/s
1	10	0	174	7.690	35,46
2	10	0	112	7.690	54,76
3	10	0	198	7.690	31,04
4	10	0	144	7.690	42,72
5	10	0	84	7.690	73,21
6	10	0	76	7.690	81,23
7	10	0	63	7.690	97,79
8	10	0	59	7.690	103,84
9	10	0	86	7.690	71,80
10	10	0	57	7.690	108,69

Resultados por URL

URL No.	Nombre	Clics	Errores	Errores [%]	Tiempo promedio entre clics [ms]
1	cursos	99	0	0,00	106

Resultados

`Número total de clics: 99 (0 Errores).`
Tiempo promedio entre clics: 106 ms.

La prueba de estrés muestra el comportamiento que tiene la aplicación en una situación de carga de trabajo alta, como el que se tiene previsto que el periodo de inscripciones a los cursos de PROTECO.

6.2. Pruebas unitarias

Las pruebas unitarias es uno de los métodos encaminados a mejorar la calidad de los sistemas de software.
En un sistema de calidad de software (o Quality Assurance), existen principalmente dos tipos de pruebas:

- **Pruebas unitarias** (o de aceptación): comprobaciones que se realizan a las unidades lógicas del software. Se verifica que una unidad funciona correctamente por sí misma, sin tener en cuenta las relaciones que pueda tener con otras partes del sistema.

- **Pruebas de integración** (o pruebas de sistema o integración funcionales): Se realiza una comprobación del sistema globalmente, haciendo énfasis en las colaboraciones entre unidades. Se prueba cada una de las opciones (o casos de uso) que ofrece el sistema, con la posibilidad de ser procesos automáticos, acciones sobre la interfaz gráfica, etc.

Las unidades lógicas de un programa son aquellas partes en que se dividido el sistema para entenderlo mejor. Pueden ser los módulos, paquetes, clases, subsistemas, funciones o cualquier otro mecanismo que ofrezca el lenguaje de programación que estamos utilizando. Para simplificar, se utilizan las clases como unidades lógicas.

Para aprender a separar un programa en unidades lógicas, es imprescindible aprender análisis y diseño, apoyándonos en técnicas y metodologías como UML, uso de patrones de diseño, Yourdon, Merisse, etc.

6.2.1. Conexión a la base de datos

- **Requerimientos a cubrir**

La conexión por medio de JDBC necesita los siguientes parámetros:

Dirección del servidor.
Puerto de la base de datos.
Nombre de la base de datos.
Nombre de usuario.
Contraseña de usuario.

- **Condiciones prerrequisito**

Proporcionar los parámetros de conexión y estar habilitado como cliente en el servidor de la base de datos.

- **Estradas de prueba**

Esta prueba consiste en establecer una conexión a la base de datos, realizar operaciones básicas de la base de datos (sentencias de tipo “INSERT”,”SELET”,”JOINS”,”DELETE”,”UPDATE”) y cerrar la conexión.

- **Resultados de prueba**

Los datos registrados conservan la integridad deseada, excepto por aquellos datos que contienen el carácter “ ‘ “ (comilla simple), esto se debe a que dicho carácter se utiliza para indicar el inicio y fin de una cadena de caracteres, para evitar esto es necesario escapar el carácter de la siguiente manera” \’ ”.

- **Evaluación de resultados**

La conexión es satisfactoria. Las operaciones ocurren de manera adecuada, obteniendo el resultado esperado para ser evaluado. Es necesario cerrar la conexión a la base de datos después de terminar el proceso para evitar problemas en el rendimiento del sistema y evitar falta de recursos.

6.2.2. Implementación de los Java Beans de entidad

- **Requerimientos a cubrir**

Los beans de entidad son componentes en lenguaje Java destinados a manejar la información proveniente o destinada a la base de datos. Son una representación de la capa de persistencia en el manejo orientado a objetos.

- **Condiciones prerrequisito**

Establecer una conexión con la base de datos.

- **Estradas de prueba**

Datos extraídos de la base de datos de cada una de las tablas de entidad. Estos datos son asignados a través de los métodos de acceso que tienen los beans de entidad.

- **Resultados de prueba**

Los beans tuvieron la capacidad de hacer un mapeo de una línea de la base de datos en un objeto, los métodos get / set de cada objeto de tipo bean funcionaron adecuadamente para asignar y recuperar valores de los atributos.

- **Evaluación de resultados**

Los beans de entidad mostraron el comportamiento esperado, facilitando las tareas de recuperación y almacenado de datos de manera persistente utilizando la base de datos, permitiendo un manejo más simple de representación de las entidades como objetos con sus métodos de asignación y recuperación de información.

6.2.3. Validación del nombre de usuario

- **Requerimientos a cubrir**

Los datos necesarios son los siguientes:
- Nombre de usuario (login).
- Contraseña de usuario (password).

- **Condiciones prerrequisito**

Establecer una conexión con la base de datos.

- **Estradas de prueba**

Se proporcionan los datos de nombre de usuario y contraseña correctos.

- **Resultados de prueba**

Si los datos del usuario son validos se obtiene una respuesta afirmativa o en el caso contrario negativa al validar el nombre y contraseña del usuario. Al tener éxito en la autentificación del usuario se genera un bean de entidad con los datos del usuario que se autentifico.

- **Evaluación de resultados**

La validación del usuario se realiza de una manera simple al implementar la clase que esta diseñada para esta tarea. No es necesario especificar los parámetros de conexión a base de datos cada vez que se requiere autentificar un usuario. La referencia creada al bean de entidad con los datos del usuario recién autentificado es de un modo simple, sin necesidad de generar una nueva conexión a la base de datos para cargar la información del usuario.

6.2.4. Lista de cursos

- **Requerimientos a cubrir**

Se debe mostrar los cursos que están en el periodo de inscripción.

- **Condiciones prerrequisito**

Establecer una conexión con la base de datos.

- **Estradas de prueba**

Ninguna.

- **Resultados de prueba**

Devuelve una colección de beans de entidad que contienen la información de cada uno de los cursos que se encuentran en periodo de inscripción, ordenados por la fecha de inicio del curso de manera descendente. El mostrar los cursos dentro de una tabla de cadenas no permite la operación en tipos de datos como las fechas o el número de alumnos de un modo simple, a diferencia de la colección de los beans que contienen la información en el formato más adecuado permitiendo realizar operaciones o comparaciones de forma simple.

- **Evaluación de resultados**

El procedimiento es exitoso al mostrar la lista de todos los cursos.

6.2.5. Lista de paquetes

- **Requerimientos a cubrir**

Se debe mostrar una lista de los paquetes que están en el periodo de inscripción.

- **Condiciones prerrequisito**

Establecer una conexión con la base de datos.

- **Estradas de prueba**

Ninguna.

- **Resultados de prueba**

Devuelve una colección de beans de entidad que contienen la información de cada uno de los paquetes y los cursos que pertenecen a cada paquete que se encuentran en periodo de inscripción, ordenados por fecha de inicio del curso de manera descendente. El mostrar los cursos en una lista de cadenas no permite la operación en tipos de datos como las fechas o el número de alumnos de un modo simple, a

diferencia de la colección de los beans que contienen la información en el formato más adecuado permite realizar operaciones o comparaciones de forma simple.

- **Evaluación de resultados**

El procedimiento es exitoso al mostrar la lista de todos los paquetes.

6.2.6. Insertar un alumno en el curso

- **Requerimientos a cubrir**

Se registran los datos del alumno y se registra el alumno en el curso y/o paquete seleccionado.

- **Condiciones prerrequisito**

Obtener la colección de los cursos y paquetes que se encuentran en periodo de inscripción.

- **Estradas de prueba**

Datos del alumno.
Lista de cursos disponibles (en periodo de inscripción).
Lista de los cursos en los que se inscribirá el alumno.
Lista de los paquetes disponibles (en periodo de inscripción).
Lista de los paquetes en los que se va a inscribir el alumno.

- **Resultados de prueba**

Colección de los cursos en los que el alumno fue inscrito.

- **Evaluación de resultados**

Los datos se insertaron correctamente, siempre y cuanto la lista de cursos en los que se va a inscribir al alumno se encuentre en la lista de cursos disponibles.

6.2.7. Borrar alumno del curso

- **Requerimientos a cubrir**

Eliminar un alumno de un curso seleccionado

- **Condiciones prerrequisito**

Ninguna.

- **Estradas de prueba**

Clave de registro del alumno.
Identificador del becario que va a realizar la operación.

- **Resultados de prueba**

Elimina el registro que contiene la relación del alumno y curso en el sistema, y se almacena en bitácora el identificador del becario que realizo esta acción.

- **Evaluación de resultados**

El procedimiento funciona correctamente al eliminar el alumno del curso y registrar esta operación en la bitácora de la base de datos.

6.2.8. Actualizar alumno del curso

- **Requerimientos a cubrir**

Actualizar la información del alumno en la base de datos.

- **Condiciones prerrequisito**

Generar una instancia de clase BeanAlumno con los nuevos datos del alumno.

- **Estradas de prueba**

Ingresar el identificador del alumno y el bean con los nuevos datos.

- **Resultados de prueba**

Un valor verdadero si la actualización se efectuó correctamente y un valor falso si no realizo ninguna actualización.

- **Evaluación de resultados**

El registro del alumno en la base de datos se actualiza sin ningún problema con los datos proporcionados.

6.2.9. Generación de ficha de pago en PDF

- **Requerimientos a cubrir**

Crear documento en formato PDF, con el formato de la ficha de pago definido por PROTECO, este documento debe estar disponible para impresión en todo momento.

- **Condiciones prerrequisito**

Seleccionar los cursos de los cuales se requiere imprimir el comprobante de inscripción.

- **Estradas de prueba**

Ninguna.

- **Resultados de prueba**

El sistema genera el documento que contiene la ficha de pago con cada uno de los cursos o paquetes que solicitó el alumno.

- **Evaluación de resultados**

El sistema genera correctamente el documento que contiene las fichas de pago.

6.3. Pruebas integrales

6.3.1. Inscribir alumno

- Requerimientos a cubrir.

Registrar un alumno en el sistema e inscribirlo en uno o varios cursos que se encuentran en proceso de inscripción abierta. Los datos necesarios para inscribir un alumno en un curso son:

- Nombre
- Apellido paterno
- Apellido materno
- E-mail
- Procedencia
- Número de cuenta
- Carrera
- Semestre
- Logia
- Pass Word

- Condiciones prerrequisito

Ingresar al sistema e identificarse como becario.

- Entradas de prueba

Formatos	Valores	Campos
Texto	Caracteres del código ASCII.	Nombre, apellido paterno, apellido materno, carrera, semestre, login y password
Email	Caracteres del código ASCII.	Email
Lista	Caracteres del código ASCII.	Procedencia
Número de cuenta	Numérico.	Número de cuenta
Checkbox		Selección del curso

- Resultados de la prueba

Formatos	Valores	Mensaje
Texto	Vacío	“No dejar campos vacíos”
Texto	Login existente en la base de datos	“Cambiar el Nombre de Usuario”
Checkbox	Vacío	“No dejar campos vacíos”

- Evaluación de resultados

El procedimiento es correcto al llenar el formulario con los datos requeridos y al seleccionar de la lista el curso o cursos en los que se desea inscribir el alumno. En el caso particular en el que se deja uno o varios campos vacíos en el formulario, el sistema muestra el mensaje adecuado. En el caso de que el login se encuentre registrado por otro usuario previamente registrado, el sistema solicita el cambio del mismo por otro login.

6.3.2. Imprimir ficha de pago

- Requerimientos a cubrir.

Imprimir la ficha de pago de cada curso en el que esta inscrito el alumno con la cantidad a pagar en las cajas de la Facultad de Ingeniería. Se deben mostrar los siguientes datos en la ficha de pago:

  - Curso
  - Fecha de inicio
  - Horario
  - Costo
  - No. de cuenta
  - Apellido Paterno
  - Apellido Materno
  - Nombre(s)
  - Clave de Registro

- Condiciones prerrequisito

El alumno debe estar inscrito por lo menos en un curso.

- Entradas de prueba

Formatos	Valores	Campos
Liga (link)		Seleccionar curso Seleccionar clave
Checkbox		Selección del curso

- Resultados de prueba

Formatos	Valores	Mensaje
Checkbox	Vacío	Seleccione el curso

- Evaluación de resultados

El procedimiento es simple para la selección del alumno y del curso o cursos en los que se requiere una impresión del comprobante de inscripción las veces que sea necesario.

6.3.3. Registrar folio de la ficha de pago

- Requerimientos a cubrir

Registrar el número de folio del recibo de pago, después de realizar el pago en la caja de la Facultad de Ingeniería.

- Condiciones prerrequisito

El alumno debe estar inscrito por lo menos en un curso.

- Entradas de prueba

Formatos	Valores	Campos
Liga (link)		Seleccionar curso Seleccionar clave
Checkbox		Selección del curso
Texto	Letras y números	Folio, costo. En el caso de ser un curso con por beca, llenar el campo "Quien otorgo la Beca".

- Resultados de prueba

Formatos	Valores	Mensaje
Texto	Vacío	"No dejar campos vacíos"

- Evaluación de resultados

El procedimiento de registro de folio es correcto después de seleccionar el curso y el alumno. En el caso particular en el que el alumno cuenta con beca el sistema se habilita el campo de captura.

6.3.4. Modificar datos de alumno

- Requerimientos a cubrir.

Modificar los datos registrados de un alumno en el sistema.

- Condiciones prerrequisito

Ingresar al sistema e identificarse como becario.

- Entradas de prueba

Formatos	Valores	Campos
Liga (link)		Seleccionar curso Seleccionar clave
Texto	Caracteres del código ASCII.	Nombre, apellido paterno, apellido materno, carrera, semestre, login y password
Email	Caracteres del código ASCII.	Email
Lista	Caracteres del código ASCII.	Procedencia
Número de cuenta	Numérico.	Número de cuenta

- Resultados de prueba

Formatos	Valores	Mensaje
Texto	Vacío	"No dejar campos vacíos"

- Evaluación de resultados

El proceso para modificar los datos del alumno funciona correctamente. El sistema mostrar el mensaje de alerta cuando se el formulario presenta al menos uno de los campos de nombre, apellidos o email vacíos.

6.3.5. Eliminar la inscripción de un alumno

- Requerimientos a cubrir.

Eliminar el registro de un alumno en uno curso.

- Condiciones prerrequisito

Ingresar al sistema e identificarse como becario.

- Entradas de prueba

Formatos	Valores	Campos
Liga (link)		Seleccionar curso Seleccionar clave
Checkbox		Selección del curso

- Resultados de prueba

Formatos	Valores	Mensaje
Checkbox	Vacío	“Seleccione el curso”

- Evaluación de resultados

El procedimiento para eliminar la inscripción de un alumno en un curso funciona correctamente, es necesario seleccionar al menos un curso debido a que en ocasiones un alumno esta inscrito en dos o más cursos, de este modo se administra individualmente cada uno de los cursos en los que esta inscrito el alumno.

6.3.6. Dar de alta curso

- Requerimientos a cubrir.

Registrar un nuevo curso en el sistema.

- Condiciones prerrequisito

Contar con privilegios de coordinador de cursos.

Para adicionar nuevos cursos al sistema se necesita contar con los privilegios de coordinador con esto se tendrá acceso al menú de creación de cursos. Previo a la adición de algún curso se deben dar de alta los catálogos generales del sistema que permiten agregar los cursos de manera precisa. Los cursos no podrán ser creados si los horarios de estos se traslapan con horarios de cursos anteriormente creados que se encuentren en el sistema.

- Entradas de prueba

Formatos	Valores	Campos
Texto	Números, '-'	Semestre
Lista	Caracteres.	Estado Tipo de curso Sala Nombre de curso Precio Paquete Horario
Lista	Números	Fecha de inicio Fecha de termino Numero máximo de alumnos Numero de clases Semana de inicio

- Resultados de prueba

Formatos	Valores	Mensaje
Lista	Vacío	"No pudo crearse el curso, no pueden existir campos vacíos."
Texto	Vacío	"No pudo crearse el curso, no pueden existir campos vacíos."

- Evaluación de resultados

La adición de cursos tuvo el comportamiento esperado, permitiendo al usuario dar de alta los cursos necesarios siempre y cuando cumplan con los requerimientos de logística necesarios.

6.3.7. Modificar curso

- Requerimientos a cubrir.

Modificar los datos del curso registrado.

- Condiciones prerrequisito

Contar con privilegios de coordinador de cursos.
Para modificar cursos al sistema se necesita contar con los privilegios de coordinador con esto se tendrá acceso al menú de modificación de cursos. Previo a la modificación de algún curso este debe de haber sido dado de alta en el sistema previamente. Los datos modificados no deben traslaparse con datos de cursos previamente creados que se encuentren en el sistema.

- Entradas de prueba

Formatos	Valores	Campos
Liga (link)		Selección del curso
Texto	Números, '-'	Semestre
Lista	Caracteres.	Estado Tipo de curso Sala Nombre de curso Precio Paquete Horario
Lista	Números	Fecha de inicio Fecha de termino Numero máximo de alumnos Numero de clases Semana de inicio

- Resultados de prueba

Formatos	Valores	Mensaje
Lista	Vacío	"No pudo crearse el curso, no pueden existir campos vacíos."
Texto	Vacío	"No pudo crearse el curso, no pueden existir campos vacíos."

- Evaluación de resultados

La modificación de cursos tuvo el comportamiento esperado, permitiendo al usuario modificar los cursos necesarios siempre y cuando cumplan con los requerimientos de logística necesarios.

6.3.8. Eliminar curso

- Requerimientos a cubrir.

Eliminar un curso del sistema

- Condiciones prerrequisito

Para eliminar cursos al sistema se necesita contar con los privilegios de coordinador con esto se tendrá acceso al menú de eliminar cursos. Previo a la eliminación de algún curso este debe de haber sido dado de alta en el sistema previamente. El curso a eliminar no debe haber sido asignado a ningún instructor o estar en proceso de inscripción.

- Entradas de prueba

Formatos	Valores	Campos
Liga (link)		Seleccionar curso

- Resultados de prueba

Formatos	Valores	Mensaje
Texto		El curso no ha podido ser eliminado debido a que existe una lista del curso o instructores asignados

- Evaluación de resultados

La eliminación de cursos tuvo el comportamiento esperado, permitiendo al usuario eliminar los cursos necesarios siempre y cuando cumplan con los requerimientos de logística necesarios.

6.4. Pruebas de usuario

Con un enfoque eminentemente práctico, las pruebas de usuario permiten verificar la usabilidad de los sitios Web, y suponen el mejor servicio para complementar los resultados obtenidos de las revisiones de requerimientos.

Las pruebas de usuario se basan en la observación de las técnicas de navegación que emplean diferentes personas al acceder a la Web, y en el contraste de su experiencia de utilización del sitio. Para su realización, seleccionamos a un grupo de becarios que harían las veces de los actores y realizarán las operaciones más importantes del sistema.

En líneas generales se mide la manera en que el usuario realiza una tarea concreta, el tiempo y número de clics que le supone acabarla y los errores que comete durante proceso.
Todos los datos podrían clasificarse en los siguientes grupos:

- Datos empíricos sobre el funcionamiento de la aplicación

El tiempo que los usuarios emplean en realizar cada una de las distintas tareas es uno de los criterios que se utilizan para medir la eficiencia de la interfaz de usuario.

- Impresiones subjetivas de los usuarios

Después de las pruebas se pide a los usuarios que comenten datos significativos sobre su experiencia al llevarlas a cabo, como complejidad, esfuerzo requerido, etc.

Prueba de usuario: ADMINISTRADOR

Nombre:	Administrar Becarios
Descripción:	Se encarga de dar de alta un becario, baja o modificar la información de este, además de actualizar el login y password de estos.
Curso principal:	1. El usuario selecciona la opción “Becarios” del menú 2. Selecciona las opciones de Alta, Baja, Edición de becario etc. Dependiendo de lo que requiera.

Objetivo: Dar de alta un becario.

1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fue de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Nombre:	Alta del administrador
Descripción:	Se da de alta al nuevo administrador de una lista de becarios.
Curso principal:	1. El usuario selecciona la opción “Administrador” del menú 2. Selecciona al nuevo becario que será administrador.

Objetivo: Cambiar administrador.

1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fue de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Prueba de usuario: COORDINADOR DE CURSOS

Nombre: Alta de un curso
Descripción: El usuario registra los datos para dar de alta un curso en una de las
Curso principal:
1. El usuario selecciona la opción "Cursos".
2. Selecciona la opción "Alta".
3. Captura los siguientes datos del curso:
   - Nombre del curso.
   - Horario.
   - Fecha de inicio.
   - Fecha de terminación.
   - Semestre.
   - Sala.
   - Tipo: intersemestral, para prebecario o sabatino.
   - Costo.
   - Número máximo de alumnos.
4. Se registra el curso

Objetivo: Dar de alta un curso.

1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fué de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Nombre: Edita curso
Descripción: El usuario se encarga de hacer las tareas de alta, baja, o cambio de un curso. Para esta tarea se debe seleccionar el curso a modificar sus datos.
Curso principal:
1. El usuario selecciona la opción "Cursos".
2. Selecciona la opción "Modificar".
3. Se muestra una lista con todos los cursos.
4. Selecciona un curso.
5. Se muestran los datos del curso.
6. Se captura los nuevos datos.
7. Se actualiza la información.

Objetivo: Edita curso.
1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fué de:

0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Prueba de usuario: INSTRUCTOR

Nombre: Publicar material
Descripción: El usuario coloca un documento relacionado al curso en el cual se encuentra asignado como instructor, este documento debe estar disponible solo para que los alumnos del curso lo descarguen.
Curso principal:
1. El usuario selecciona la opción "Administrar material".
2. Se selecciona la opción "Subir material".
3. Se captura el curso en el cual se desea agregar este material.
4. Se capturan una descripción general del material.
5. Se selección el archivo.
6. Se procede a almacenar el material.

Objetivo: Publicar material.

1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fué de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Nombre: Calificar alumnos
Descripción: En el usuario debe registrar las calificaciones de los alumnos, se necesita hacer el registro para cada uno de los cursos en los que el alumno se encuentra como instructor.
Curso principal:
1. El usuario selecciona la opción "Calificar alumnos".
2. Se muestra una lista con los cursos en los que el usuario se encuentra asignado como instructor.
3. Se selecciona un curso.
4. Se muestra una lista de todos los alumnos inscritos el curso.
5. Se captura la calificación que debe ser un número decimal de cero a diez.
6. Se almacena la calificación del alumno.

Objetivo: Calificar alumnos.
1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fué de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Prueba de usuario: BECARIO

Nombre: Inscribir alumnos
Descripción: El usuario registra los datos del alumno y selecciona el curso que el alumno desea tomar.
Curso principal:
1. El usuario selecciona la opción "Inscripción".
2. Se capturan los datos personales del alumno.
3. Se muestra una lista de los cursos disponibles.
4. Se selecciona el curso o cursos que desea tomar.

Objetivo: Inscribir alumnos.

1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fué de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Nombre: Generar comprobante de pago.
Descripción: El usuario genera el comprobante de pago. Para cada curso se debe generar una ficha de pago con el formato que se maneja en PROTECO. Es necesario que se mantenga la integridad de los datos que se presentan el la ficha de pago.
Curso principal:
1. Se usa "*ver datos de un alumno*".
2. Se selecciona la opción "Imprimir comprobante". Se debe generar la ficha de pago con los datos del alumno y del curso.

Objetivo: Generar comprobante de pago.
1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

2.-El tiempo que me llevo realizar la operación fué de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Prueba de Usuario: ALUMNO

Nombre: Consultar apuntes
Descripción: Permite consultar y descargar las notas del curso, estos notas son archivos que deben estar disponibles para los alumnos que lo soliciten, pero solo se permite descargar aquellas notas, que son de los cursos en los que está inscrito el alumno.
Curso principal:
1. El usuario selecciona la opción "Notas del cursos".
2. Se muestra una lista con las notas que están disponibles para el curso en el que esta inscrito.
3. Selecciona una de las notas que aparecen en la lista.
4. Se permite la operación de descarga del archivo.

Objetivo: Consultar apuntes

1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

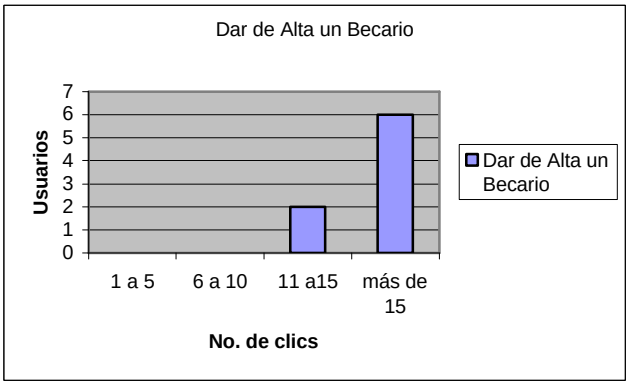
2.-El tiempo que me llevo realizar la operación fué de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

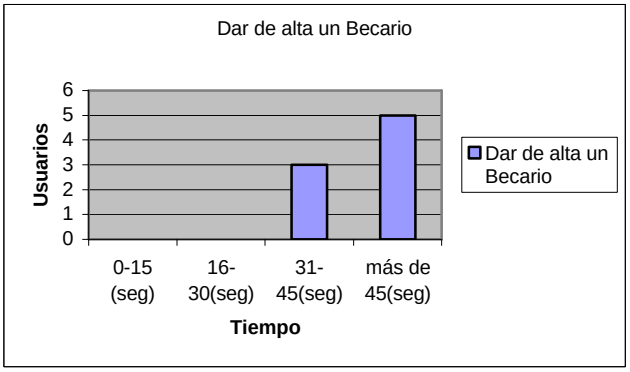
Nombre: Consultar calificación
Descripción: Muestra las calificaciones del alumno.
Curso principal:
1. El usuario selecciona la opción "Calificación(es)".
2. Se muestra una lista con el nombre del curso o de los cursos en los que esta registrado el alumno junto con su calificación final.

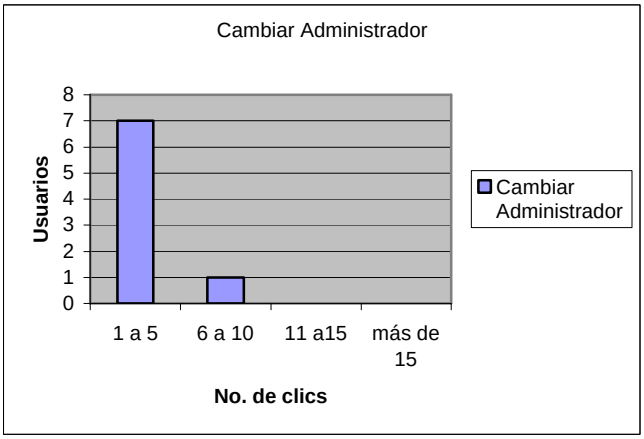
Objetivo: Consultar calificación
1.-Número de clics que hice
1 a 5 6-10 11-15 más de 15

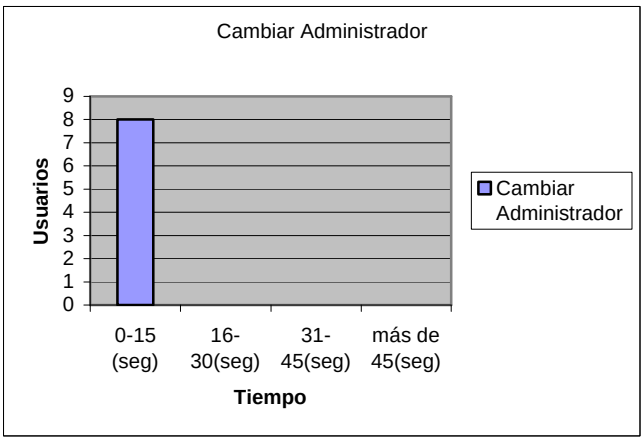
2.-El tiempo que me llevo realizar la operación fué de:
0-15 (seg.) 16-30(seg.) 31-45(seg.) Más de 45(seg.)

Resultados del administrador.

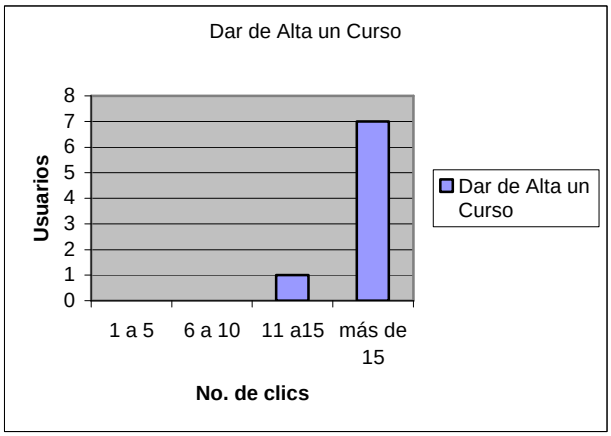


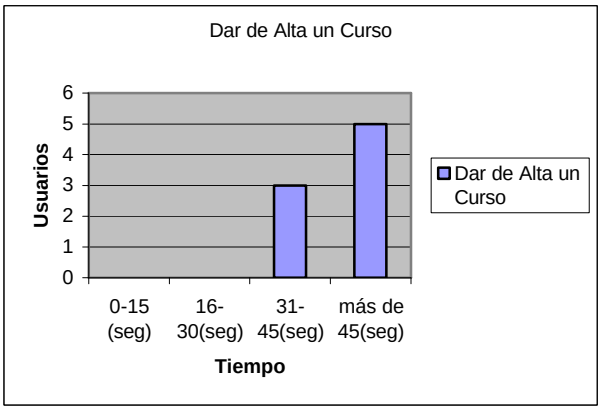


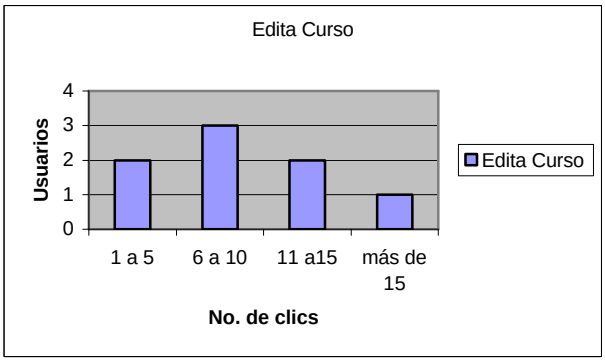


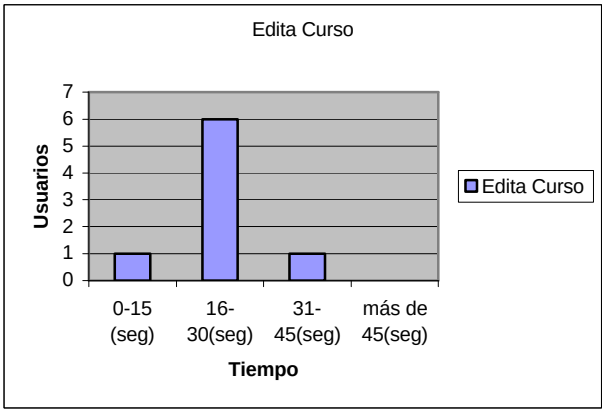


Resultados del coordinador de cursos.

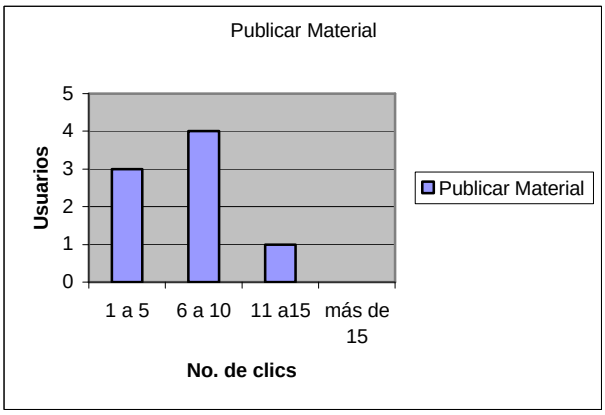


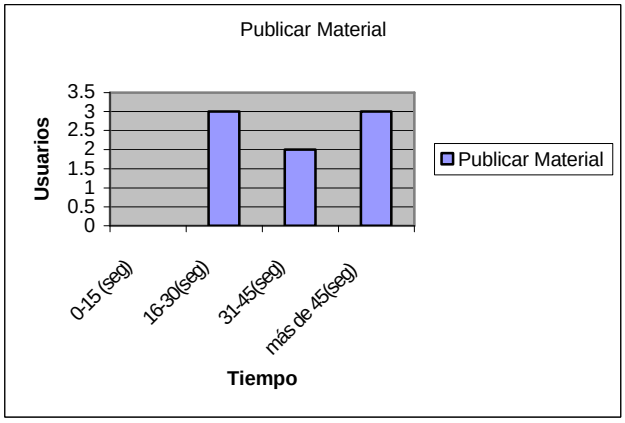


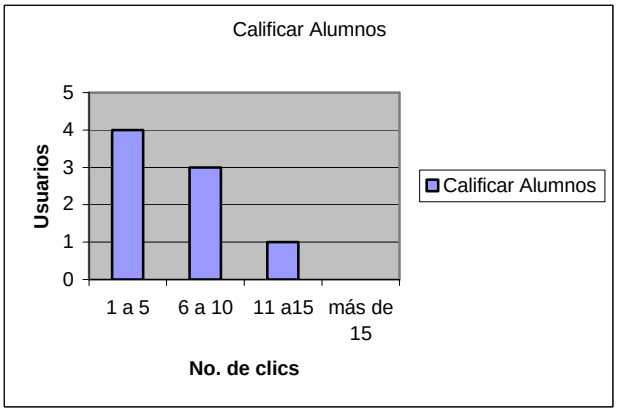


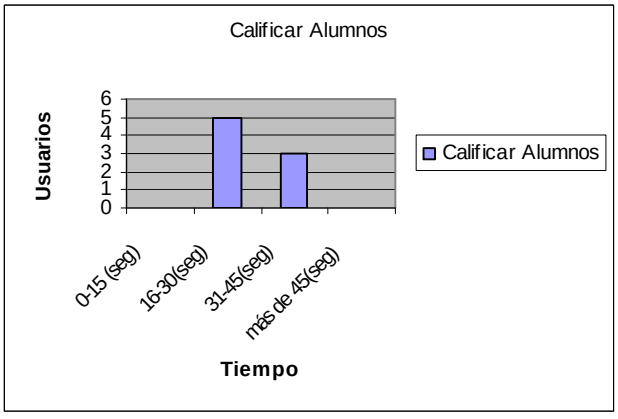


Resultados instructor.

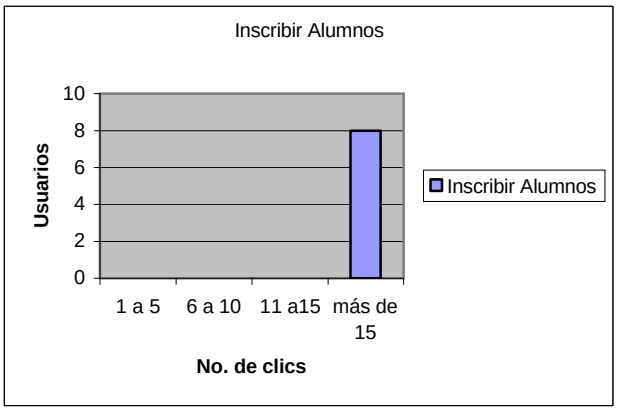


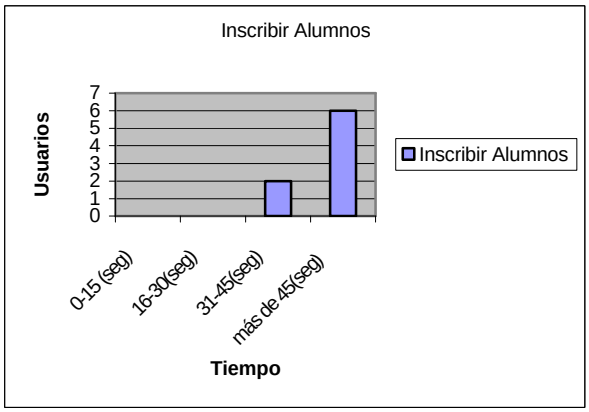


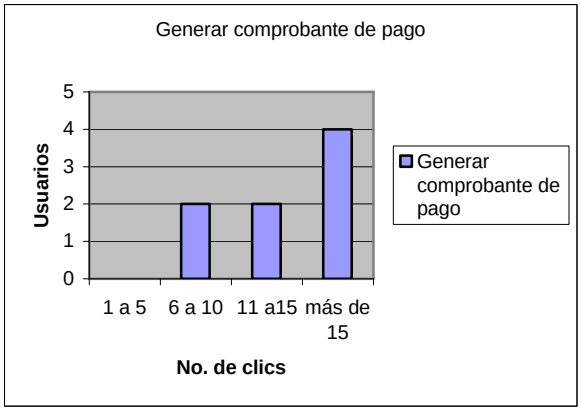


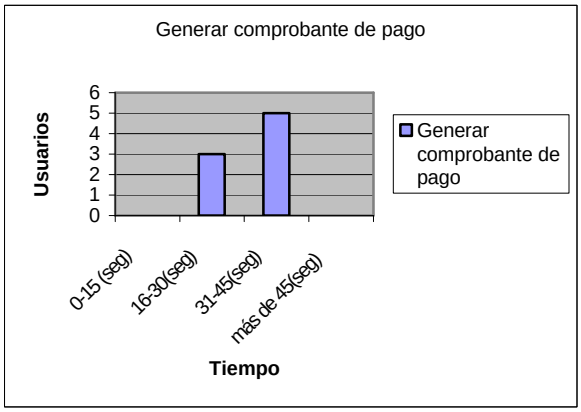


Resultados becario.

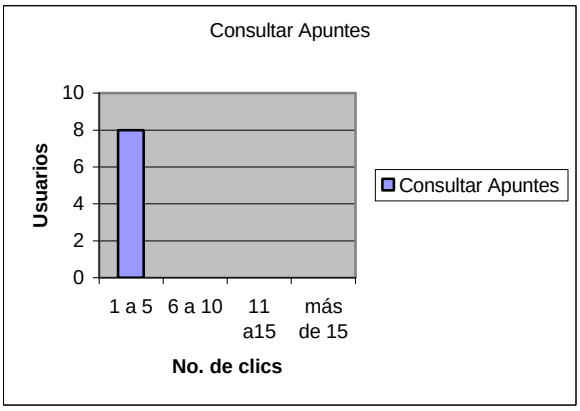


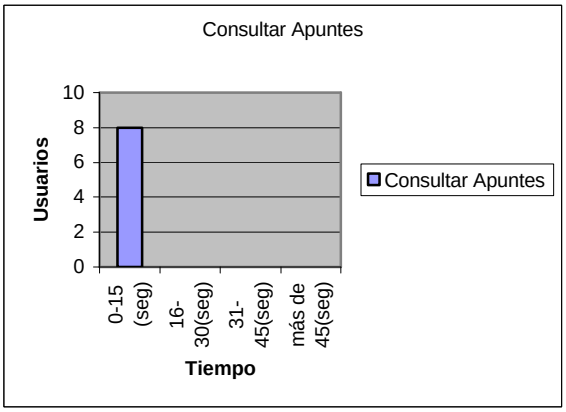


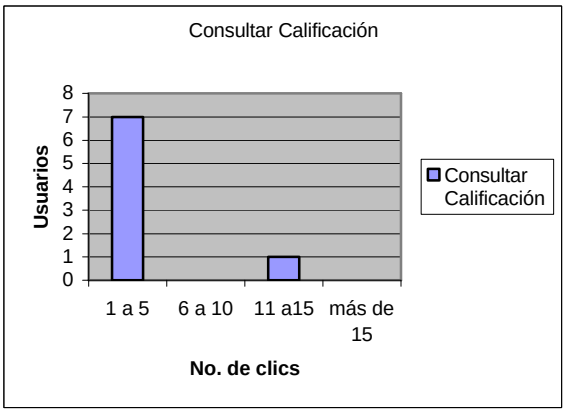


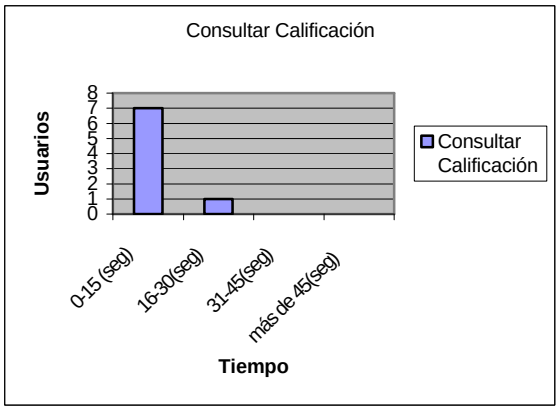


Resultados alumno.


Consultar Apuntes
10
8
6
4
2
0
Usuarios
1 a 5
6 a 10
11 a15
más de 15
No. de clics
Consultar Apuntes


Consultar Apuntes
10
8
6
4
2
0
Usuarios
0-15 (seg)
16-30(seg)
31-45(seg)
más de 45(seg)
Tiempo
Consultar Apuntes


Consultar Calificación
8
7
6
5
4
3
2
1
0
Usuarios
1 a 5
6 a 10
11 a15
más de 15
No. de clics
Consultar Calificación


Consultar Calificación
Usuarios
0-15 (seg)
16-30(seg)
31-45(seg)
más de 45(seg)
Tiempo
Consultar Calificación

Los resultados del estudio de usabilidad arrojan la siguiente información:

Para las actividades desarrolladas por el administrador y el coordinador de cursos el número de clics y tiempos de respuesta del sistema es considerablemente mayor del número de clics y tiempos de respuesta de los actores Becario, Instructor y Alumno. Esto se debe a que las actividades de administración involucran el manejo completo de tablas de la bases de datos, es decir dar de alta información, edición de información etc.

Para las actividades de los actores Becario, Instructor y Alumno los tiempos de respuesta del sistema son considerablemente rápidos así como el número de clics disminuye considerablemente con respecto a los actores administrativos.

7. Liberación

Se han entregado tres versiones del sistema, la primera versión solo contemplaba el proceso de inscripción a los cursos, generaba el recibo de pago, la preinscripción y la inscripción. Esta versión fué utilizada con éxito en los cursos intersemestrales del 2005. La segunda versión contemplaba ya las secciones de becario, instructor, coordinador y administrador, el sistema de inscripción ya se encontraba depurado y con mejoras que se le agregaron. La tercera versión es la versión completa que entregamos junto con la versión escrita de esta tesis, ya contempla todos los aspectos que se pidieron en los requerimientos y contiene las secciones respectivas de todos los actores.

7.1. Producto entregado

La última versión del sistema versión 1.0 se encuentra funcionando, está instalada en un servidor de la DIE que cumple todas las especificaciones del sistema, el sistema por su diseño permitirá su actualización periódica para incrementar la eficiencia de este.

7.2. Requisitos de instalación

Servidor de Base de Datos: Oracle 8.0.5 para Linux

Hardware:

- 32 MB de RAM, en caso de que haga cargas elevadas se requerirá incluso 128 MB
- SWAP, aproximadamente el triple de la memoria RAM instalada
- 400 MB de disco duro para la instalación
- Al menos unas 150 MB de disco duro por defecto por cada base de

Software:

- Al menos el Kernel 2.0.34
- GLIBC 2.0.7, incluida en Red Hat 5.2 y superiores o Debian 2.0
- JDBC JDK 1.0.2 ó 1.1.1
- ProC/C++ gcc 2.7.2.3 o superior
- Tcl8.0

Servidor Web

Hardware:

- 32 MB de RAM, en caso de que haga cargas elevadas se requerirá incluso 128 MB
- SWAP, aproximadamente el triple de la memoria RAM instalada
- 400 MB de disco duro para la instalación
- Al menos unas 150 MB de disco duro por defecto por cada base de

Software:

- JSK 1.3
- Contenedor Servlet/JSP

Conclusiones

La evolución constante del hardware y del software obligan al cliente plantearse nuevas necesidades para mantener vigente su negocio, así mismo estas nuevas necesidades de los clientes son traducidas por nosotros los Ingenieros en Computación en soluciones que permiten continuar con el movimiento evolutivo de los sistemas computacionales. Esta interacción entre necesidades y soluciones es lo que impulsa a la industria del software en el mundo. Desde las primeras aplicaciones stand-alone que corrían en mainframes hasta las actuales aplicaciones de última generación que comunican sistemas y que se ejecutan en tiempo real a través de la red mundial siempre ha existido una constante fundamental: la evolución.

Nada permanece estático ni permanecerá, todo en este negocio cambia, es por esto que nosotros como Ingenieros debemos adaptarnos a este cambio constante. Cambios en las tecnologías, cambios en los paradigmas, cambios en los sistemas, cambios en la electrónica. La adaptación al cambio es la clave de la supervivencia del Ingeniero, esto lo hemos aprendido bien durante la formación que recibimos en nuestra Universidad, el ejemplo más tangible de estos cambios es este trabajo de tesis que realizamos, viendo hacia atrás cuando comenzamos a visualizar el desarrollo de esta tesis, nuestra visión de la Ingeniería de software era muy diferente con la que contamos en estos momentos.

Observando en este momento nuestro trabajo realizado, podemos interrogarnos en muchos aspectos de su realización, de cómo hubiera sido mejor realizarlo, por que no utilizar otro tipo de esquema de desarrollo, etc. Este es el gran regalo que nos dejó la elaboración de este trabajo, darnos cuenta de que con cada desarrollo, con cada aplicación que diseñemos, con cada línea de código que programemos debemos mejorar, debemos evolucionar, nunca quedando conformes con lo que hemos hecho.

Lo mejor a nuestro juicio de este trabajo, es que este sistema esta diseñado para crecer, diseñado para que se le cambien y se le mejoren muchas cosas que se le pueden mejorar, pero esto queda en manos de nuevas generaciones de Ingenieros que vienen detrás de nosotros empujando con fuerza.

Las necesidades con las que convivíamos día a día durante nuestra estancia como becarios del PROTECO nos permitió desarrollar este trabajo de tesis que será utilizado por nuestros compañeros becarios como herramienta de trabajo durante su estancia en el programa, este fue un aliciente a nuestro trabajo, no solo saber que nos titularíamos por la elaboración de este sistema, si no saber que este sistema sería utilizado para beneficiar a nuestro programa de becarios que tanto queremos.

Como estudiantes de la Carrera de Ingeniería en Computación de la Facultad de Ingeniería y como exbecarios del programa de becarios PROTECO sentimos la necesidad de dejar algo en beneficio de nuestra institución y de nuestros compañeros. Realizar este sistema fue la mejor opción para regresar algo de lo mucho que nos dio el PROTECO y que mejor que un sistema que sabemos que será muy utilizado por nuestros compañeros.

El desarrollo de este trabajo nos permitió familiarizarnos con las tecnologías que a nuestro juicio son y serán las principales en el desarrollo de software en el mundo. La

implementación y utilización de software libre nos permitió conocer que podemos trabajar con lo último en tecnología sin tener que gastar un centavo, un concepto que en estos días suena contrastante con la cotidianidad en la que el dinero marca la pausa entre lo que es de “calidad” y lo que no lo es.

El sistema se diseñó bajo el modelo RUP (Rational Unified Process), lo cual a grandes rasgos representó diseñar y programar en un modelo cíclico, siempre envolvente que permite interactuar con todos los actores de una manera más amigable. La diferencia de este proceso de diseño con modelos más tradicionales es que toma en cuenta los problemas que siempre ha presentado la elaboración de software bajo un proceso estructurado, los cambios en todo momento son bien recibidos, ya que estos forman parte del proceso de diseño.

La elección del lenguaje se baso en las ventajas que Java representa para el desarrollo de cualquier sistema, como era obvio para nosotros el diseño y desarrollo lo enfocaríamos al paradigma orientado a objetos por las ventajas que ya mencionamos a lo largo de la tesis.

Quedamos satisfechos con el desempeño del portal, a sabiendas de que este puede y debe ser mejorado por nuestros compañeros que siguen nuestros pasos. Esperamos que la elaboración de este sistema sea solo el comienzo de una larga lista de proyectos que beneficien de todas las maneras posibles a nuestra Facultad y por lo tanto a nuestra Universidad.

Solo nos resta agradecer a todos los que directa e indirectamente fortalecieron este proyecto, ya sea aportando ideas, aportando su tiempo para realizar pruebas o simplemente por soportarnos mientras trabajamos en él. Nadie lo sabe todo, esta es una premisa constante en el mundo de la Ingeniería la cual corroboramos durante la elaboración de este proyecto, lo importante es saber que contamos con amigos y compañeros que nos permitieron acercarnos a ellos para atender nuestros cuestionamientos y así poder avanzar en la consolidación de este proyecto. A todos ellos muchas gracias.

Alfredo Benavides Martínez

El Portal cumple con los objetivos esperados.

Actualmente el sistema tiene en operación dos semestres y con ello se tiene un histórico de los cursos impartidos y al conocer el número de alumnos que llega a tener cada curso, se pueden por ejemplo, organizar grupos paralelos en los cursos con alta demanda.

El proceso de inscripción de alumnos ahora es más simple y rápido, debido a que se elimino la tarea de llenar las fichas de pago y la lista de alumnos manualmente, evitando de ese modo errores en los datos de la ficha de pago, también se evitan problemas tales como inscripción de un número mayor de alumnos a la prevista, la perdida del registro de los alumnos y de sus calificaciones por perdida de los documentos, como se manejaba anteriormente es un problema que ya no se presenta debido a que toda esta información se administra en la base de datos con la opción de hacer el respaldo de la misma.

Para el coordinador de los cursos, ahora se tiene de manera inmediata y organizada la información de los cursos por cada semestre, como ejemplo de esto se tiene un seguimiento de los ingresos percibidos por el concepto de cursos, con la opción de mostrar el registro de los folios de los recibos de pago expedidos en la caja de la Facultad de Ingeniería.

El coordinador puede administrar la asignación de instructores de los cursos entre el grupo de becarios, con esta información posteriormente el sistema tiene la opción de imprimir el recibo de pago de beca para cada becario por los cursos que impartio durante el periodo intersemestral, de una manera más ordenada y automática.

Después de que el instructor del curso ha terminado la evaluación de los alumnos puede registrar las calificaciones finales, de esta manera el coordinador puede hacer una revisión y determinar el grupo de alumnos que cuentan con el derecho a una constancia.

Se controla la asignación de la fecha, horario, número de alumnos y sala de los cursos previniendo posibles errores en la disponibilidad de espacios en la planeación de los cursos.

Desde la puesta en operación del sistema, se comenzó con la creación de una base de datos de alumnos, con esto se puede promover los nuevos cursos y dar continua difusión del programa de becarios dentro de la comunidad estudiantil.

Los alumnos por su parte pueden realizar su inscripción a un curso por medio del Web y obtener los datos ya escritos en la ficha de pago; pueden revisar la calificación obtenida en el curso de manera simple.

El lenguaje de programación seleccionado permitió el desarrollar el sistema.

Para el desarrollo de este sistema se contaba con una amplia variedad de opciones, pero la selección de un lenguaje de programación orientado a objetos, específicamente Java, hizo más simple las etapas de análisis, desarrollo e implementación de lo aplicación.
Para el análisis del sistema el utilizar una arquitectura de tres capas, el describir el sistema mediante los casos de uso y los diagramas de clases como lo señala el estándar

de UML permitió planear el sistema de una manera modular, centrando nuestra atención más en el comportamiento del sistema, que en el como se codificaría una solución para cada uno de los puntos a cubrir del sistema.

Durante la fase inicial del desarrollo de la aplicación tuvimos la sensación de no tener un avance significativo, debido a que se estaban codificando las clases principales que manejarían el encapsulamiento y la persistencia de los datos, pero al termino de esta fase el avance comenzó a ser más substancial debido a que se ensamblan las clases necesarias para tener resuelto un caso de uso, es decir, una función completa del sistema.

El utilizar el API de JSPs de java nos permitió administrar el diseño de las páginas Web del sistema, de manera sumamente sencilla ahorrando tiempo y en ocasiones permitiendo modificaciones posteriores evitando el proceso de compilación por el usuario.

El manejo de la persistencia de los datos fue administrado por la base de datos. El manejador de la base de datos seleccionado permitió una integración simple en el sistema, la razón principal de para utilizarlo es la facilidad que representa implementar una conexión utilizando el API de JDBC en la parte de aplicación y a través de instrucciones en SQL se planearon las consultas, búsquedas y ordenamientos necesarios.

Se realizó un análisis tomando en cuenta un mapeo desde la base de datos hacia el código que manipularía dichos datos y la solución se encontró haciendo uso de los Java Beans, con ello se logró encapsular el estado de las entidades como el alumno, el becario y los cursos.

El servidor Web soporta el nivel de carga esperado para el sistema.

El servidor Tomcat utilizado cumplió con los requerimientos mínimos necesarios para hacer el despliegue de la aplicación; dentro de las pruebas de estrés mostró un comportamiento eficiente durante una demanda constante de páginas del sistema. La instalación y administración es simple y al estar basado en la tecnología de Java es compatible con los APIs utilizados en el desarrollo del sistema.

Por último me queda mencionar la gran satisfacción de haber contribuido de algún modo al PROTECO, poniendo en práctica parte de lo que aprendí durante los semestres de la carrera en los que participe como becario.

Julio César Saynez Fabián

Anexos

Instalación de J2SE SDK para Linux

Requisitos del sistema

Java 2 SDK, edición estándar, 1.4.2 se apoya en i586 Intel y el 100% de las plataformas compatibles que funcionan con Linux.

Se requiere de al menos 75 Mega bytes de espacio en el disco duro libre antes de proceder a instalar el software de Java 2 SDK.

Se requiere un mínimo de 32 Mega bytes de RAM. Recomendado 48 Mega bytes del RAM.

Instrucciones de instalación

Formatos de instalación - Java 2 SDK 1.4.2 está disponible en dos formatos de la instalación.

- Archivo binario self-extracting

Este archivo se puede utilizar para instalar Java 2 SDK en una localización elegida por el usuario. Éste se puede instalar por cualquier persona (no sólo root), y puede ser instalado fácilmente en cualquier directorio. Mientras no tengan privilegios de administrador, no es posible desplazar la versión del sistema de la plataforma de Java usada por Linux. Para utilizar este archivo, ver la instalación de binario self-extracting abajo.

- Paquetes de la RPM

Requiere la cuenta del usuario “root” para instalar, la instalación predefinida lo coloca en un directorio que substituye la versión del sistema de Java proveída por Linux. Para utilizar este paquete, ver la instalación del archivo de RPM abajo.

Elige el formato de la instalación que más convenga a sus necesidades.

Nota: Para cualquier texto en esta página que contiene la notación siguiente, debes sustituir el número de versión apropiado de la actualización de Java 2 SDK para la notación.
<version> Por ejemplo, si se descargo la actualización 1.4.2_01, el comando siguiente:

./j2sdk-1_4_2_ <version>-linux-i586.bin se convirtió:
./j2sdk-1_4_2_01-linux-i586.bin

Instalación de binario auto descompresión

Utilizar estas instrucciones si desea utilizar el archivo binario auto descompresión para instalar Java 2 SDK. Si desea instalar los paquetes de RPM en lugar de otro, ver la instalación del archivo de RPM.

1. Descargar y comprobar el tamaño del archivo para asegurarse de que se haya descargado el autentico.
Se puede descargar en cualquier directorio seleccionado; no tiene que ser el directorio donde deseas instalar la Java 2 SDK.

2. Cerciorase de que los permisos sean los adecuados para el archivo self-extracting.
Ejecutar este comando:

chmod +x j2sdk-1_4_2_<version>-linux-i586.bin

3. Cambiar el directorio a la localización en donde quisiera que los archivos fueran instalados.

El paso siguiente instala la Java 2 SDK en el directorio actual.
4. Ejecute el binario self-extracting.

Ejecutar el archivo descargado, precedido por el PATH. Por ejemplo, si el archivo está en el directorio actual, colocar ". /" (el punto "." es necesario si no está en la variable de entorno PATH):
./j2sdk-1_4_2_ <version>-linux-i586.bin

Se exhibe la licencia del código binario. Aceptar sus términos.

Java 2 SDK está instalado en un directorio llamado j2sdk1.4.2_<version> en el directorio actual.

Ver la documentación del API de las preferencias para más información sobre preferencias en la plataforma de Java.

Instalación del archivo RPM

Utilizar estas instrucciones si desea instalar Java 2 SDK bajo la forma de paquetes RPM. Si desea utilizar el archivo binario self-extracting en lugar de otro, ver la instalación de binario self-extracting.
1. Descargar y comprobar el tamaño del archivo.

Descargar el archivo en cualquier directorio seleccionado.
Antes de que descargar el archivo, revisar tamaño proporcionado en la página de descarga. Comparar una vez ese tamaño del archivo al tamaño del archivo descargado para cerciorarse de que son iguales.

2. Extraer el contenido del archivo descargado.

Cambiar el directorio a donde se localiza el archivo descargado y ejecutar los siguientes comandos para cambiar los permisos de ejecución y después ejecutar el binario para extraer el archivo RPM:

chmod a+x j2sdk-1_4_2_<version>-linux-i586-rpm.bin
./j2sdk-1_4_2_ <version>-linux-i586-rpm.bin

Observar que inicia con ". /" se requiere si no se tiene "." en dentro de las variables de entorno.

Se exhibe un acuerdo de licencia, el cual solicita que se acepte antes de que la instalación pueda proceder. Una vez que haya aceptado la licencia, la instalación crea el archivo j2sdk-1_4_2_<version>-linux-i586.rpm en el directorio actual.

3. Utilizando la cuenta de "root" se ejecuta el comando de RPM para instalar los paquetes que abarcan Java 2 SDK:

rpm - iv j2sdk-1_4_2_<version>-linux-i586.rpm

4. Elimina el archivo bin y el archivo de RPM si desea ahorrar espacio de disco.
5. Salir de la sesión de root.

Instalación Apache – Tomcat para Linux

El siguiente texto explica como realizar la instalación de un servidor Apache con soporte para JSP y Servlets mediante el uso de Tomcat.

El primer paso es la instalación de Java 2SE SDK como se vio en la sección anterior

Se debe descargar la distribución 'jakarta-ant'. Es necesario crear el directorio /usr/local/jakarta, y añadir la variable JAKARTA_HOME=/usr/local/jakarta al .bash_profile correspondiente. Tras descomprimir el archivo descargado mediante
tar -xvzf jakarta-ant-src.tar.gz debera copiarlo todo en /usr/local/jakarta/jakarta-ant, y teclear lo siguiente:

./bootstrap.sh

El resultado será un archivo denominado ant.jar dentro del directorio /lib, que será utilizado durante la compilación de Tomcat.
Descargar desde el ftp de ftp.renr.es el archivo jakarta-servletapi-3.2-src.tar.gz desde el directorio /pub/linux/tomcat, y descomprir el contenido en un directorio denominado /usr/local/jakarta/jakarta-servletapi. Tras hacer esto, ejecutar el siguiente comando:

./build.sh dist

Esto tendrá como resultado el archivo servlet.jar en el subdirectorio lib/, que será necesario para la compilación de WatchDog.

Desde el mismo directorio dentro del ftp descargar el archivo jakarta-tomcat-3.2.1-src.tar.gz, y descomprir este en /usr/local/jakarta/jakarta-tomcat. Dentro de ese directorio ejecutar el siguiente comando:

./build.sh dist

Con esto habrá terminado de instalar Tomcat.

Para probar su funcionamiento ejecutar

/usr/local/jakarta/dist/tomcat/bin/startup.sh

Lo cual iniciara un servidor Web, y usando el navegador para acceder a

http://localhost:8080.

Para detenerlo ejecutar:

/usr/local/jakarta/dist/tomcat/bin/shutdown.sh.

Instalación del componente para Web del sistema PROTECO.

1. Crear una base de datos en Oracle 8i.
2. Ejecutar los archivos de instalación en la base de datos de Oracle 8i en el siguiente orden:

   protecodb.sql
   views.sql
   procedures.sql
   catalogos.sql

   Para ejecutar un script en la base de datos basta con colocar el carácter "@" al inicio del nombre del script, incluyendo la ruta en la que se encuentra.
   Ej.:

   SQL>@/tmp/protecodb/protecodb.sql;

3. Crear la carpeta portal/ dentro del directorio de webapps/ del directorio donde se instaló Tomcat.

6) Quedando del siguiente modo:

7) (Directorio de Tomcat)/webapps/portal

4. Copiar el archivo portal.war, en el directorio porta/
5. Descomprimir el archivo portal.war con el siguiente comando

jar cvf portal.war

6. Modificar el archivo de texto proteco.properties para asignar los valores requeridos. El archivo se encuentra en dentro WEB-INF/

Configuración de las variables del sitio	
url	Dirección URL en la cual se encuentra el sistema. Ej.: http://www.fi-b.unam.mx/portal
DirNotas	Directorio en el que se almacenan las notas de los cursos. La ruta debe asignarse de forma absoluta en cualquier parte del sistema de archivos con permisos de lectura y escritura para el proceso de Tomcat. Ej.: /opt/portalPTC/archivos_web/notas_proteco
DirFotos	Directorio en el que se almacenan las fotografías que los becarios asignaran como parte de su cuenta. La ruta debe asignarse de forma absoluta en cualquier parte del sistema de archivos con permisos de lectura y escritura para el proceso de Tomcat. Ej.: /opt/portalPTC/archivos_web/fotos
ImaBecario	Nombre de la imagen predeterminada en la pantalla de información del becario. Esta imagen debe encontrarse debajo del directorio asignado en *DirFotos***.** Ej.: becario.jpg
logoPDF	Nombre de la imagen utilizada para generar la ficha de pago, esta imagen se debe encontrar en el directorio de imágenes/ del sistema. Ej.: logoPDF.jpg

Configuración JDBC de la base de datos	
BDurl	URL utilizada en la conexión. ODBC:oracle:thin:@132.248.59.3:1521:die
BDuser	Nombre del usuario autorizado en la base de datos. Ej.: proteco
BDpassword	Contraseña del usuario autorizado en la base de datos. Ej.: contraseña
utilizarPool	Si se opta por utilizar un pool de conexiones en la configuración de Tomcat se debe asignar esta variable con la cadena “si” y el nombre del pool debe de ser “jdbc/myoracle”. Esto inhabilita la configuración de la base de datos de las variables DB… del archivo “proteco.properties”. Ver nota de configuración del pool al final del documento. Por defecto el valor es “no”. Ej.:no

Configuración del servicio de correo electrónico	
Email-ip	Dirección IP del servidor de correo utilizado como SMTP. Ej.: 132.248.59.6
Email-host	Dirección URL del servidor de correo utilizado como SMTP. Ej.: aries.fi-b.unam.mx
Email-de	Dirección de correo utilizada como remitente en los mensajes de correo enviados. Ej.: cursos@fi-b.unam.mx

Configuración del servicio de correo electrónico	
Email-asunto-recordatorio	Cadena utilizada en el mensaje de recordatorio para los alumnos no han registrado el número de folio de su ficha de pago. Ej.: Curso de PROTECO
Email-asunto-constancia	Cadena utilizada en el mensaje de aviso para los alumnos que pueden pasar a recoger su constancia del curso que tomaron en PROTECO. Ej.: Constancia de PROTECO
Email-asunto-propaganda	Cadena utilizada en el mensaje de aviso de los próximos cursos de PROTECO. Ej.: Propaganda de PROTECO
Email-mensaje-constancia-path	Directorio en el que se encuentra el mensaje en formato HTML para ser enviado por e-mail a los alumnos que pueden pasar a recoger su constancia. Ej.: /opt/portalPTC/email/constancia.html
Email-mensaje-propaganda-path	Directorio en el que se encuentra el mensaje en formato HTML para ser enviado por e-mail a los alumnos con propaganda para los nuevos cursos de PROTECO. Ej.: /opt/portalPTC/email/propaganda.html
Email-mensaje-recordatorio-path	Directorio en el que se encuentra el mensaje en formato HTML para ser enviado por e-mail a los alumnos para que registren el número de folio de su ficha de pago. Ej.: /opt/portalPTC/email/recordatorio.html

7. Iniciar el servicio del servidor Tomcat con el siguiente comando (Directorio de Tomcat)/bin/startup.sh

8. Verificar la instalación de la aplicación en el URL del servidor donde se instalo la aplicación en el contexto portal/ utilizando el nombre de usuario becario y contraseña becario2005.

Configuración del contexto y del pool de conexiones de Tomcat.

Modificar el archivo server.xml y agregar las siguientes líneas del la aplicación dentro de las etiquetas de <Host></Host>.

```
<Context className="org.apache.catalina.core.StandardContext"
cachingAllowed="true"
charsetMapperClass="org.apache.catalina.util.CharsetMapper" cookies="true"
crossContext="false" debug="1"
displayName="PROTECO"
docBase="/tomcat/webapps/portal"
mapperClass="org.apache.catalina.core.StandardContextMapper"
path="/portal" privileged="false" reloadable="true" swallowOutput="false"
useNaming="true"
wrapperClass="org.apache.catalina.core.StandardWrapper">
<Loader className="org.apache.catalina.loader.WebappLoader"
checkInterval="10" debug="0"
delegate="false" loaderClass="org.apache.catalina.loader.WebappClassLoader"
reloadable="true"/>

<!—Inicio del pool de conexiones de la base de datos.
     Solo necesario si la variables  utilizarPool tiene el valor si   -->

<ResourceParams name="jdbc/myoracle">
   <parameter>
       <name>username</name>
          <value>usuario</value>
   </parameter>
   <parameter>
       <name>password</name>
          <value>contraseña</value>
   </parameter>
   <parameter>
       <name>url</name>
         <value>jdbc:oracle:thin:myschema@132.248.59.3:1521:die</value>
   </parameter>
   <parameter>
       <name>driverClassName</name>
          <value>oracle.jdbc.driver.OracleDriver</value>
   </parameter>
   <parameter>
       <name>maxIdle</name>
          <value>10</value>
   </parameter>
```

```
  <parameter>
       <name>factory</name>
          <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
   </parameter>
   <parameter>
       <name>maxWait</name>
           <value>-1</value>
   </parameter>
   <parameter>
       <name>maxActive</name>
          <value>15</value>
   </parameter>
   </ResourceParams>

<!-- Fin de la configuración  del pool de conexiones -->

</Context>
```

Glosario

Add-ons
Añadidos o módulos que se incorporan a los que ya se tienen y que proporcionan nuevas funcionalidades.

Address [dirección]
Existen tres tipos de direcciones de uso común dentro de Internet: dirección de correo electrónico (e-mail address); IP (dirección Internet); y dirección hardware o dirección MAC (hardware o MAC address). Véase también e-mail address, dirección IP, e Internet address.

Algoritmo
Conjunto de reglas bien definidas para la resolución de un problema. Un programa de software es la transcripción, en lenguaje de programación, de un algoritmo.

Attachment
Véase Archivo adjunto.

ASCII
American Standard Code of Information Interchange: Código normalizado estadounidense para el intercambio de la información. Código que permite definir caracteres alfanuméricos; se lo usa para lograr compatibilidad entre diversos procesadores de texto. Se pronuncia "aski".
Archivo Adjunto
Archivo que acompaña un mensaje de e-mail. Es apropiado para el envío de imágenes, sonidos, programas y otros archivos grandes.

API [Aplication Programa Interface]
(Application Programming Interface) Interfaz para la programación de aplicaciones. Es en lo que se basan lo programadores para hacer compatible un programa con el sistema operativo. DirectX, por ejemplo, es un conjunto de APIs creada por Microsoft para mejorar el funcionamiento de aplicaciones multimedia y juegos en Windows 95/98/2000. Las funciones DirectX pueden ser utilizadas libremente por los programadores para mejorar sus productos.

Apache
Servidor Web de distribución libre. Fue desarrollado en 1995 y ha llegado a ser el más usado de Internet.

ANSI
American National Standards Institute. Organización que promueve el desarrollo de estándares en los Estados Unidos. Es miembro de la ISO (International Organization for Standardization).

Browser
Un visualizador de documentos WWW. En su forma más básica son aplicaciones hipertexto que facilitan la navegación en Internet, los más avanzados cuentan con funcionalidades plenamente multimedia y permiten la navegación indistintamente en

browsers HTTP (WWW), ftp, gopher, lectura de News, correo, ...- Véase WWW y servidor WWW.

Chat
Es un sistema reconocido para hablar (mediante texto) en tiempo real con personas que se encuentran en otras computadoras conectados a la red. En algunas versiones permite el uso de la voz.

Ciberespacio
Término creado por Wilian Gibson en su novela fantástica "Neuromancer" para describir el "mundo" de las computadoras y la sociedad creada en torno a ellos.

CGI
Del idioma ingles Common Gateway Interface una tecnología de la World Wide Web que permite a un cliente (explorador Web) solicitar datos de un programa ejecutado en un servidor Web.

Cliente [Client]
Un sistema o proceso que solicita a otro sistema o proceso que le preste un servicio. Una estación de trabajo que solicita el contenido de un archivo o servidor de archivos es un cliente de este servidor. Véase modelo Cliente-Servidor. Equipo que utiliza los recursos compartidos de los servidores.

Cliente/Servidor [Client/Server]
Red en la que el procesamiento está distribuido entre un servidor y un cliente, cada uno de ellos con funciones específicas. También se utiliza para describir a las redes que tienen servidores dedicados. Es lo opuesto a de igual a igual.

Dirección IP [IP Address]
Dirección exclusiva adjudicada a un lugar concreto en la red formada por cuatro números separados por puntos, con valores entre 0 y 255 Ejemplo:222.123.15.21. También se define como una dirección de 32 bits asignada por el Protocolo Internet en STD 5, RFC 791. Se representa usualmente mediante notación decimal separada por puntos.

Download [Transmision]
El traslado de un archivo o información de un nodo de la red a otro. Generalmente se refiere a transferir un archivo de un servidor, como un host, a un "pequeño" nodo. Descargar, bajar. Transferencia de información desde un computador a otro mediante una red de transmisión de datos.

Dynamic HTML
Variante del HTML (Hyper TextMark-up Language) que permite crear páginas Web más animadas.

E-MAIL [Electronic Mail]
El correo electrónico es el servicio más básico, antiguo, y el más utilizado dentro de Internet. Permite intercambiar mensajes, programas, audio, vídeos e imágenes.

Encryption [cifrado]
Encriptación. Es el tratamiento de los datos contenidos en un paquete a fin de impedir que nadie excepto el destinatario de los mismos, pueda leerlo. (por supuesto, este debe conocer la clave de desencriptado). Existen muchísimos tipos de cifrado.

Espacio Web
Mega bytes en el servidor dedicados a alojar tú página Web y/o otros documentos.

GNU
Conjunto de programas desarrollados por la Free Software Foundation (Fundación por el Software Libre); es de uso libre.

HTML [Hyper Text Markup Language]
Lenguaje en el que se describen los documentos que se exportan a través de WWW. Admite componentes hipertexto y multimedia. Véase también WWW.

HTTP [HyperText Transmission Protocol]
Protocolo usado para la transferencia de documentos WWW.

HTTPS [HyperText Transmission Protocol Secured]
URL creada por Netscape Communications Corporation para designar documentos que llegan desde un servidor WWW seguro. Esta seguridad es dada por el protocolo SSL (Secure Sockets Layer) basado en la tecnología de encriptación y autentificación desarrollada por la RSA Data Security Inc. Internet Explorer también usa esta tecnología.

IP
Es el protocolo de envío de paquetes donde el paquete tiene una dirección destino, y éste se envía sin acuse de recibo. Cuando una persona se conecta a Internet, se le asigna una dirección IP.

ISO [International Organization for Standardization]
Organización Internacional para la Normalización. Fundada en 1946, es responsable de la creación de estándares internacionales en muchas áreas, incluyendo la informática y las comunicaciones. En la actualidad la componen 89 países.

Linux
Sistema operativo gratuito para computadoras personales derivado de Unix.

Link
Enlace. Imagen o texto destacado, mediante subrayado o color, que lleva a otro sector del documento o a otra página Web.

Login
Conectarse a una computadora con identificación de usuario y contraseña. Acción de introducir el nombre a través del teclado para acceder a otra computadora.

Página Web
Es un documento realizado en HTML y que es parte de un sitio Web. Si por Ej. Se tienen más de dos páginas Web, entonces la forma correcta de nombrar sería sitio Web, ya que un sitio Web es aquel sitio de Internet que contiene más de una página Web.

Password
Contraseña, Clave Secreta.

Perl
Lenguaje de programación muy utilizado para la elaboración de aplicaciones CGI (véase).

Portal
Sitio Web que nos da un conjunto de links de diferentes sitios Web dirigidas por diferentes temas. Suelen tener la posibilidad de hacer búsquedas de una o más palabras y darnos las páginas Web que las contienen.

Prebecario
Alumno inscrito al programa PROTECO que se encuentra en la prime etapa de desarrollo.

Protocolo
Descripción formal de formatos de mensaje y de reglas que dos computadores deben seguir para intercambiar dichos mensajes.

PROTECO
Programa de Tecnología en Cómputo

Pruebas de Integración:
Los casos de prueba que la sustentan permiten probar la interacción de las unidades con el fin de verificar el comportamiento final.

Pruebas Unitarias
En esta etapa se desarrollan casos de prueba que permiten probar una aplicación sin tener todos sus componentes desarrollados. Se asegura que las piezas de software funcionan individualmente para después poder realizar una prueba integral más estable.

Query
Consulta. Búsqueda en una base de datos.
Sistema Operativo
Programa que administra los demás programas en una computadora.

Unix
Sistema operativo multiusuario, fue muy importante en el desarrollo de Internet.

Upload [cargar, subir]
En Internet, proceso de transferir información desde un computador personal a un servidor de información. Ver también: "download".

URL/URI
(Universal Resource Locators/Universal Resource Identifiers) Sistema unificado de identificación de recursos en la red. Las direcciones se componen de protocolo, FQDN, y dirección local del documento dentro del servidor. Este tipo de direcciones permite identificar objetos WWW, gopher, ftp, etc.

Username
También denominado user ID, es el nombre de usuario por el que cada usuario es identificado en la red.

XML
Siglas de eXtesible Markup Lenguaje. XML es un metalenguaje, es decir, un lenguaje para la definición de otros lenguajes.

Bibliografía

Entendiendo UML: La guía del desarrollador, con una aplicación java basada en Web, por Paul Harmon y Mark Watson; Morgan Kauffman Publishers, Inc., 1998

Objetos, componentes y Estructuras con UML: The Catalysis Aproach, por Desmond F. D'Souza y Alan C. Wills, Addison Wesley Longman, 1998.

Patrones de diseño aplicados a Java, por Stephen Stelting y Olav Maassen; Pearson Prentice may, 2003.

Java 2 Fundamentos, Vol. I, por Cay S. Hosrtmann y Gay Cornell; Prentice Hall, 2003.

Oracle 9i, Manual de referencia, por Kevin Loney y George Koch; Mc Graw Hill, 2003.

Referencias

Sitio Web de Java
http://java.sun.com

Sitio Web de Oracle
http://www.oracle.com

Medidor de estrés
http://www.paessler.com

Ejemplos de Programación en java
http://www.idevelopment.info/

Lenguajes de Programación
http://www.etse.urv.es/EngInf/assig/lp/

Ejemplos de Oracle
http://www.psoug.org/library.html
http://www.techonthenet.com/oracle/index.php

Manual de CCS (Cascading Style Sheets)
http://www.w3.org/Style/CSS/learning

Modelado de Sistemas con UML
http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/index.html

Arquitectura del JSP (Java Server Page)
http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html

Organización de Proyecto de Software
http://readyset.tigris.org/nonav/es/