UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE REGISTRO EN LÍNEA AL SERVICIO DE ACCESO REMOTO A MATERIAL DOCUMENTAL EN FORMATO DIGITAL.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERIO EN COMPUTACIÓN

P R E S E N T A:

EFRAIN JACOBO ÁVILA JUÁREZ

DIRECTORA DE TESIS:
MTRA. SILVIA SOCORRO BALLESTEROS ESTRADA
2015

AGRADECIMIENTOS.

A mis padres Guadalupe e Isaías, por el apoyo que desde mi nacimiento me dieron. Sin él nunca hubiera alcanzado mis logros. Por su paciencia, perseverancia y cariño.

En especial a mi madre, por ser roca solida a la cual me asi en tiempos de tormenta, fue difícil entender lo que me decías pero siempre tuviste razón.

A mis hermanos Eunice, Elías e Isaías, por ser parte de mi vida.

A Eunice por enseñarme que aunque las posibilidades estén en tu contra siempre es posible salir adelante.

A Elías por enseñarme a ser fiel a quien eres y que siempre es posible perdonar.

A Isaías por enseñarme a reír en cualquier momento.

A mis abuelos Gertrudis, José Gil y Jeremías que aunque ya no estén conmigo lo que me enseñaron lo llevare siempre.

A la comunidad Taimada Oscar, Enrique, Daniel y Linda por estar conmigo cuando más lo necesite.

A Amalia por encontrarme en el camino, donde a veces sentí que no encontraría a nadie. Por ayudarme a madurar. Te quiero mucho glovia.

A mi asesora la Maestra Silvia Socorro Ballesteros Estrada, por su apoyo en la realización de este trabajo.

A mis compañeros de trabajo el Maestro Rafael Ibarra Contreras y el Maestro Dante Ortiz Ancona, por su apoyo en la revisión ortográfica y técnica.

A la Facultad de Ingeniería, los conocimientos que adquirí me permitieron enfrentar los retos que me presentó el desarrollo del proyecto. La resolución de problemas y la búsqueda de información, así como el adaptarse y aprender nuevas cosas son parte de la formación que recibí de los profesores de la facultad.

A ese ser que la humanidad llama Dios, que aunque es difícil cada día trato de entender un poquito más.

Contenido

Introducción.

La UNAM ha encomendado a la Dirección General de Bibliotecas coordinar el Sistema Bibliotecario conforme a las políticas generales que establece el Consejo del Sistema Bibliotecario, determinando las medidas que relacionan y desarrollan a las bibliotecas.

Es de competencia de la Dirección General de Bibliotecas:

- Fijar las normas técnicas y de servicio del Sistema Bibliotecario.
- Mantener un sistema de información sobre el acervo de las bibliotecas a través de los catálogos colectivos de libros (LIBRUNAM), revistas (SERIUNAM), mapas (MAPAMEX) y tesis (TESIUNAM).
- Contribuir a la comunicación científica desarrollando índices sobre la producción científica mexicana y latinoamericana en ciencias sociales (CLASE) y ciencia y tecnología (PERIÓDICA).
- Colaborar en proyectos internacionales y nacionales relacionados con el control bibliográfico universal.
- Brindar servicios bibliotecarios y de información a través de la Biblioteca Central.
- Establecer los criterios y mecanismos apropiados para el desarrollo de la colección de recursos electrónicos y los servicios especializados para beneficio de la comunidad universitaria.
- Construir una hemeroteca electrónica de libre acceso que incluya las publicaciones científicas mexicanas más importantes denominada SciELO-México.
- Promover la actualización de personal profesional mediante programas de educación continua e intercambio académico, y la capacitación de personal bibliotecario auxiliar.
- Proporcionar asesorías sobre aspectos relacionados con edificios y automatización de bibliotecas, servicios bibliotecarios y de información, desarrollo de colecciones, organización de colecciones, entre otros.
- Definir criterios y procedimientos para la utilización racional de los recursos presupuestarios y de diverso tipo.

La Dirección General de Bibliotecas ocupa un tercer nivel dentro de la estructura universitaria, después de la Rectoría y de la Secretaría de Desarrollo Institucional, de la cual forma parte. Está conformada por una secretaría académica, cinco subdirecciones, tres secretarías técnicas, 23 departamentos, una unidad administrativa, una secretaría particular y una auxiliar, coordinados todos ellos por la Dirección General. Ver Figura I.[1]

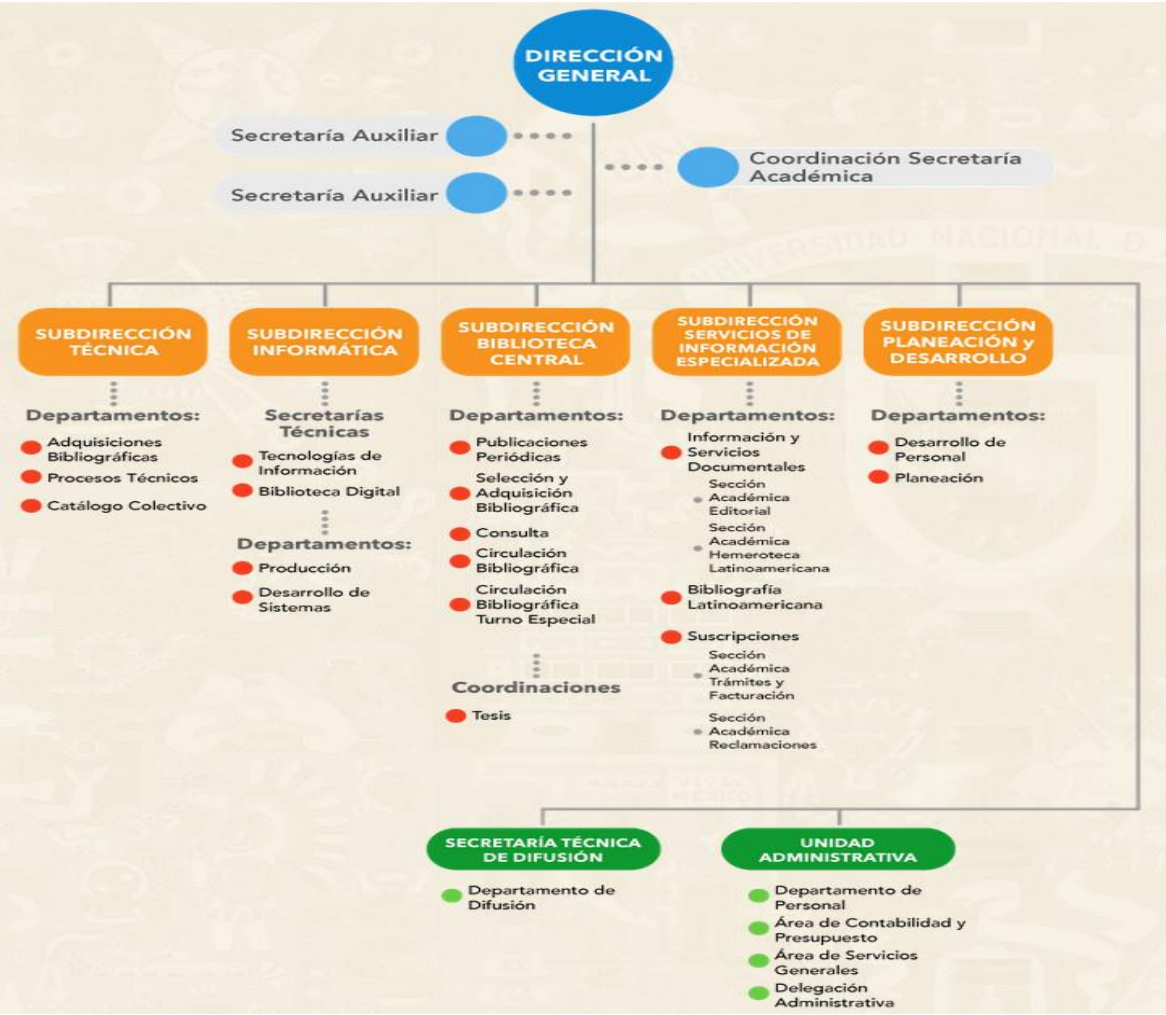

Figura I. Estructura organizacional de la Dirección General de Bibliotecas.

[1] (PORTAL DGB, 2013)

Una de las secretarías técnicas que conforman la Dirección General de Bibliotecas es la Secretaría Técnica de Biblioteca Digital responsable de la Biblioteca Digital.

El proyecto de la Biblioteca Digital de la Universidad Nacional Autónoma de México comenzó en el año de 2001, son algunos de sus objetivos:

- Desarrollar acervos digitales de alta calidad y pertinencia para apoyar la formación integral de los universitarios a través del acceso a la información académica digital, vía Internet, de manera ágil, dinámica e interactiva.
- Generar políticas de preservación y conservación de recursos digitales.
- Racionalizar los recursos humanos, económicos y de infraestructura destinados a la Biblioteca Digital.

Sus funciones son:

- Seleccionar, adquirir, organizar, resguardar y diseminar información académica digital.
- Analizar los requerimientos de hardware y software y determinar su adquisición y base en los planes y programas de estudio e investigación y difusión de la UNAM. actualización.
- Diseñar interfaces para facilitar al usuario la búsqueda de información.
- Emplear formatos múltiples de extensión, según la clase de documentos a consultarse.
- Establecer la localización física del material digital o digitalizado y proporcionar la nomenclatura y dirección de los objetos de la colección: URLs, URNs, PURLs.
- Crear o adaptar los procesos técnicos bibliotecarios para la descripción de los objetos electrónicos.
- Establecer políticas y procedimientos basados en normas y estándares internacionales para el desarrollo de bibliotecas digitales en las dependencias que lo soliciten.
- Establecer políticas de cooperación con dependencias universitarias, así como instituciones nacionales y extranjeras.
- Promover la creación de órganos asesores para orientar y reorientar los objetivos y tareas de la Biblioteca Digital y apoyar en la selección y adquisición de materiales.

- Proporcionar a la comunidad académica los medios para que haga uso de los recursos electrónicos y digitales.
- Sugerir procedimientos para facilitar el uso de la información digital.
- Evaluar sistemas, modelos y herramientas para el manejo de las tecnologías de la información.
- Establecer convenios de trabajo que ayuden a fortalecer el área de Biblioteca Digital.
- Mantener en proceso evolutivo la Biblioteca Digital, en relación con los avances en materia de tecnologías de la información.
- Contribuir a la conformación de una biblioteca Universitaria Digital.
- Administrar los recursos humanos y materiales de la Secretaría.
- Coordinar y supervisar las actividades del personal a su cargo.
- Elaborar estadísticas y proyecciones sobre las actividades que se desarrollan en esta Secretaría Técnica y presentar diferentes informes.
- Presentar anualmente el programa de actividades, objetivos y metas a realizar, así como informar los avances obtenidos.
- Aquellas que se deriven o relaciones con sus objetivos.

Para el año 2010, la Biblioteca Digital se incorpora a los acervos regulares en línea de la DGB, las funciones más importantes del portal de la Biblioteca Digital se trasladan al nuevo portal de búsquedas. De esta forma la comunidad universitaria sólo consulta una página donde puede hacer búsquedas tanto de material físico como electrónico, a esta búsqueda se le conoce como búsqueda consolidada.

I.- Antecedentes.

a- SALUAR.

Cumpliendo con una de sus funciones la Secretaría Técnica de Biblioteca Digital (en lo sucesivo **BiDi**) planteo la forma de acceder a los recursos digitales bajo su custodia en la forma de un acceso remoto, así es como surge el servicio de acceso remoto (**AR**) a los recursos digitales de la Dirección General de Bibliotecas. Se define como recurso digital a todo aquel material documental (revistas, tesis, libros, mapas, artículos científicos, imágenes, videos, bases de datos, etc.) de interés para la comunidad universitaria y que se encuentra disponible en algún formato electrónico y que debe ser accedido a través de una computadora o dispositivo móvil con conexión a Internet para su visualización. Para poder administrar a los usuarios se diseñó el Sistema de Administración Local de Usuarios de Acceso Remoto.

SALUAR son las siglas del Sistema de Administración Local de Usuarios de Acceso Remoto, tiene por objetivo registrar, actualizar y borrar a los usuarios de Acceso Remoto, el sistema fue diseñado y construido en la Secretaría Técnica de Biblioteca Digital. Este sistema fue escrito en lenguaje Java utilizando el modelo de desarrollo llamado Modelo-Vista-Controlador o MVC, apoyado sobre el framework de Apache Struts que implementa dicho modelo de desarrollo. El sistema fue pensado para que cada dependencia de la UNAM tuviese control del acceso en su comunidad particular y pudiera decidir quiénes tienen derecho a usar el servicio.

La primera entidad académica de la UNAM en solicitar el acceso remoto a los recursos electrónicos fue la Facultad de Medicina en el año 2006. En esta solicitud se incluyeron a los alumnos de licenciatura, maestría y doctorado y a todos los académicos de la misma. Los alumnos que tenían la posibilidad de usar el servicio eran alrededor de 20,000 y los profesores otros 3,000 aproximadamente.

El ejemplo de la Facultad de Medicina demostró lo importante y eficaz que podía ser el servicio para toda la comunidad universitaria.

El servicio de Acceso Remoto es el mecanismo por el cual un usuario registrado puede tener acceso a los recursos digitales que se encuentran en los catálogos de la DGB sin estar físicamente en algún campus de la UNAM, esto es cuando el usuario no accede al portal de la DGB por medio de la RedUNAM sino por medio de una conexión privada a Internet como son los servicios

de las compañías telefónicas o las compañías de comunicación con servicios tripleplay (telefonía, Internet y televisión de paga).

b- Funcionamiento de SALUAR.

El sistema funciona de la siguiente forma: el usuario acude con el administrador local de Acceso Remoto y le solicita una cuenta al servicio, el administrador local puede ser el coordinador de biblioteca, bibliotecario, responsables de cómputo, etc., el administrador debe validar que el usuario sea un alumno con sus derechos vigentes o un académico de la comunidad universitaria; una vez validado el administrador procede a inscribirlo en el sistema de Acceso Remoto pidiéndole sus datos como número de cuenta en el caso de los alumnos o número de empleado en el caso de los académicos, su nombre completo, si es alumno su nivel (iniciación universitaria, bachillerato, licenciatura, especialidad, maestría o doctorado), un correo electrónico que era opcional y su fecha de nacimiento que también es opcional. Una vez ingresados los datos se le pide firmar una responsiva por el servicio y aceptar las condiciones. Los datos de acceso son su número de cuenta o de empleado, según sea el caso, y una contraseña generada automáticamente, estos datos le son proporcionados al usuario en ese mismo momento.

Esta forma de operar tiene las siguientes limitantes:

- El usuario debe presentarse a la biblioteca, lo que a veces resulta complicado o imposible, como en el caso de las de licenciaturas a distancias, donde los usuarios pueden estar en cualquier parte del mundo.
- El usuario pierde tiempo en traslados, esperas, fallas técnicas, etc.
- Si no cuenta con los documentos probatorios de identidad y de pertenencia a la comunidad universitaria, por extravío o robo, deberá en primera instancia realizar el trámite para obtenerlos nuevamente, para posteriormente inscribirse en el servicio de Acceso Remoto.
- No todas las dependencias cuentan con un responsable o administrador local de Acceso Remoto.
- El servicio sólo se otorgaba a estudiantes de posgrado, investigadores y académicos.

Para utilizar el servicio de Acceso Remoto el usuario debe visualizar, con ayuda de un navegador de Internet, el portal de la DGB una vez ahí hacer una búsqueda de algún tema de su interés seleccionando si se busca por título, por autor, por tema, etc. Entonces el metabuscador de la DGB regresa los registros que coinciden con su búsqueda categorizada por libros, revistas o tesis además de identificar aquellos que están disponibles en algún formato digital o sólo se encuentran en formato impreso. Un metabuscador es un servicio de búsqueda que recupera información de diferentes fuentes o bases de datos, en el caso de la DGB permite recuperar información de sus diferentes bases de datos (TESIUNAM, LIBRUNAM, PERIODICA, SERIUNAM, etc.). Si se selecciona algún recurso digital y el acceso al portal de la DGB es hecho desde RedUNAM (bibliotecas, RIU, institutos, centros de cómputo, etc.) el acceso remoto validará automáticamente al usuario y mostrará, según sea el funcionamiento del portal del proveedor, el texto citado en el registro. Si por el contrario el usuario accede a algún recurso electrónico de la DGB desde su casa, oficina, café Internet, etc., el servicio de Acceso Remoto le pedirá validarse como usuario del mismo, es en este momento que aparecen dos casillas, la primera solicita la cuenta del usuario que como ya se mencionó es la cuenta de alumno o el número de empleado según sea el caso, y la segunda la contraseña que obtuvo al momento de registrarse. Si ambos datos coinciden el sistema de Acceso Remoto re-direccionará al usuario al portal del proveedor donde podrá visualizar el recurso digital. Si por el contrario ha cometido un error al momento de ingresar los datos, o es un usuario no válido el sistema le informará que sus datos son incorrectos y no re-direccionará al portal del proveedor.

c- Módulos de SALUAR.

El sistema SALUAR es un sistema web escrito en Java y está compuesta por varios módulos:

Proxy-ejb3; es el encargado de hacer la manipulación de la información contenida en la base de datos proxybidi donde se guardan los datos de los usuarios válidos, así como de un histórico de movimientos y usuarios que ya no son válidos por razones como término de carrera, artículo 22 de la legislación universitaria, etc. Puede agregar, actualizar y borrar registros de la base de datos. La mayoría de los módulos dependen de él.

Proxy-admin; es el módulo de administración de usuarios, desde este módulo cada

administrador local puede ver a sus usuarios, su información como número de cuenta, contraseña, nombre, correo electrónico, etc. También puede actualizar o modificar esta información, así como renovar o extender la vigencia a algún usuario, o borrar a un usuario. Este módulo sólo permite al administrador ver a los usuarios de su comunidad local, si trata de dar de alta o busca a algún usuario que no pertenezca a su comunidad el sistema le informará que no puede realizar dicha operación.

Proxy-manager; es el módulo desde donde se administran todas las dependencias que se encuentran registradas en la base de datos. Desde este módulo se puede obtener la información de los administradores de dichas dependencias, así como ver a los usuarios de las mismas. Contiene un submódulo de altas masivas, el cual recibe un archivo de texto en cierto formato con la información de los usuarios a ingresar a la base de datos. Se puede dar de alta nuevas dependencias.

Proxy-mail; Este módulo crea los mensajes vía correo electrónico que se envían a los usuarios, estos mensajes son acerca de hacerle llegar por primera vez su contraseña, avisos de próxima fecha de fin de acceso, actualización de datos y término de vigencia de acceso remoto.

Proxy-scheduler2; Es el módulo encargado de enviar los mensajes vía correo electrónico invocando sucesivamente al módulo proxy-mail para crear el cuerpo de los correos electrónicos. Utiliza el API de java mail y en su primera versión estaba basada en el framework de Jboss para ejecutarse cada 2 minutos , debido a los cambios sucesivos de los requerimientos se decidió hacer una nueva versión basada sobre un API que no dependiera de Jboss pero que pudiera trabajar con este servidor de aplicaciones y con otros.

El conjunto de estos módulos forman el Sistema de Administración Local de Usuarios de Acceso Remoto en una versión inicial.

II.- Objetivo.

Dadas las características del SALUAR, que el acceso sólo era para alumnos de posgrado, ya fuera maestría o doctorado, y para académicos y que la misma DGB tiene entre sus funciones dar acceso a los acervos bibliográficos a cualquier miembro académico de la comunidad universitaria, sin distinguir nivel escolar, se planteó extender la funcionalidad de SALUAR por medio de una aplicación que permitiera a los alumnos y académicos de esta casa de estudios registrarse en el SALUAR sin necesidad de acudir a su biblioteca.

Por las características descritas antes y la forma de operar del SALUAR se propone crear un registro en línea al Acceso Remoto que amplié el uso del servicio a toda la comunidad académica universitaria.

El registro en línea al Acceso Remoto supera las características descritas, es un servicio que se encuentra disponible las 24 horas del día, los 365 días. No depende de una tercera persona para tramitarla. Los datos de validación son fáciles de reconocer y comunes a la comunidad universitaria gracias a los servicios de información digital vía Internet de la Dirección General de Administración Escolar (DGAE) y de la Dirección General de Personal (DGP) que validan la información de alumnos y académicos respectivamente.

Para registrarse al Acceso Remoto a través del portal de la DGB el usuario debe primeramente entrar al portal, una vez en el portal dar clic en “Solicita tu cuenta” el portal mostrará inicialmente un formulario para académicos, si se es alumno en la parte superior inmediata al formulario se encuentra dicha opción.

Los formularios para registrarse deben ser llenados en su totalidad, en el caso de los académicos deberán proporcionar RFC con homoclave, número de empleado UNAM, su sexo y un correo electrónico; en caso de que la información proporcionada no sea correcta el sistema le pide al solicitante corroborar dicha información, de lo contrario el sistema le pide al académico aceptar los términos y condiciones de uso del servicio para poder hacer uso del mismo, al aceptarlo el registro se realiza y el sistema avisa del éxito del registro y que la contraseña será enviada al correo electrónico proporcionado en un periodo de 10 a 15 minutos.

En el caso de los alumnos debe proporcionar número de cuenta UNAM, nivel de estudios,

plantel y carrera; como en el caso de los académicos si la información no es correcta el sistema pedirá revisarla, de lo contrario le pedirá proporcionar un correo electrónico de contacto, al cual se enviará la contraseña, una vez hecho esto le pedirá aceptar los términos y condiciones de uso, al aceptarlos el sistema avisará que el registro fue exitoso y que su contraseña será enviada al correo electrónico proporcionado en un periodo de 10 a 15 minutos.

Cabe destacar también que los requisitos para poder utilizar esta modalidad son: ser alumno inscrito en alguna de las carreras de la UNAM con sus derechos vigentes y en el caso de ser académico dar clases en la UNAM como titular, académico de tiempo completo o emérito.

Para el caso de los alumnos de intercambio, ya sean nacionales o extranjeros, pueden realizar su trámite de contraseña por la modalidad original, también para investigadores invitados o como en el caso de la Facultad de Medicina para los tutores, estos usuarios son casos especiales y es la institución donde labora o estudia la responsable de establecer el periodo de vigencia de dichos accesos, así como de su renovación.

A continuación se describen brevemente los capítulos que conforman el presente trabajo:

El capítulo 1 explica los métodos y modelos de ingeniería de software utilizados para analizar, diseñar, desarrollar e implementar este trabajo, así como las bases del mismo

El capítulo 2 resume el estado actual del SALUAR, cómo funciona y de qué partes se compone, así como los nuevos requerimientos a los que se enfrenta.

El capítulo 3 analiza los nuevos requerimientos y de este análisis se obtendrá una propuesta de solución a implementar.

El capítulo 4 muestra el diseño y desarrollo de los nuevos módulos derivados del análisis,

El capítulo 5 describe la implementación de los nuevos módulos y sus funcionalidades, así como la tecnología utilizada para ponerlos en marcha, muestra las pruebas hechas al diseño y los resultados que arrojaron dichas pruebas.

El capítulo 6 presenta las conclusiones y el impacto que han tenido los nuevos módulos en la inscripción de nuevos usuarios a servicio de Acceso Remoto, así como posibles formas de extenderlo aún más o cambios requeridos.

Capítulo 1
Conceptos básicos de ingeniería de software

1.1 Ingeniería de software.

La ingeniería de software se define, según la IEEE, como:

> "La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir la aplicación de ingeniería de software"[2]

Para poder aplicar la ingeniería de software se debe seguir un proceso de software. El proceso de software es un conjunto de actividades y resultados asociados, los cuales deben asegurar que el producto de software cumpla, siga y demuestre características de calidad. El proceso de software, independientemente del modelo que se siga, está compuesto por:

- Fase de definición. En esta fase el ingeniero y el cliente deben definir los alcances del software, características, limitaciones, datos a procesar, rendimiento, interfaces de usuario, criterios de validación, etc. Además está compuesto por 3 actividades: ingeniería de información, planificación de proyecto de software y análisis de los requisitos.
- Fase de desarrollo. La fase de desarrollo comprende la especificación gráfica de los requisitos obtenidos en la fase de definición, cómo es que se diseñará la estructura de datos, cómo se implementará la arquitectura de software, cómo ha de traducirse el diseño en un lenguaje de programación, cómo se realizaran las pruebas, etc. Tiene 3 actividades principales: diseño de software, generación de código y prueba de software.
- Fase de mantenimiento. La última fase se centra en la adaptación del software a cambios debido a requerimientos por parte del cliente, corrección de errores, a adaptaciones requeridas a medida que evoluciona el ambiente de software (cambios de versión de servidores de aplicaciones, en el caso de aplicaciones basadas en web). Dentro de este grupo se identifican 4 tipos principales de cambios: corrección, adaptación, mejora y prevención

[2] Roger S. Pressman. *Ingeniería de software.* 6ta ed. Mcgraw Hill, p. 23.

Dado que hoy en día existen una gran variedad de sistemas informáticos, también existen una gran variedad de procesos de desarrollo de software. Cada proceso de desarrollo de software puede organizar las partes principales que componen al proceso de software de manera diferente según le convenga, pero la selección errónea de este proceso de desarrollo de software puede conducir a la reducción en la calidad del producto de software o al incremento del mismo.

La mayor parte de los modelos de procesos de software se basa en unos de los siguientes tres modelos generales: el modelo en cascada o lineal secuencial, modelo iterativo o evolutivo y modelo basado en componentes.

1.1.1 Modelo en cascada o lineal secuencial.

Fue el primer modelo formal que surgió, se deriva de prácticas generales de la ingeniería de sistemas de los años 70's, ver Figura 1.1. Se puede identificar de 3 a 5 actividades principales, a continuación listo las más importantes:

- Análisis y definición de requerimientos. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces se definen en detalle y sirven como especificación del sistema.
- Diseño del sistema y software. El proceso de diseño del sistema divide los requerimientos en sistemas de hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema de software y sus relaciones.
- Implementación y pruebas. Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación
- Mantenimiento. Una vez entregado el software al cliente se producirán cambios debido a que se han encontrado errores, el software debe adaptarse para acoplarse a los cambios de su entorno externo o que el cliente requiere mejoras funcionales o de rendimiento.

Debido a que fue el primer modelo formal en aparecer contiene problemas que afectan al desarrollo del software. Algunos de estos problemas son:

1. Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo. Aunque el modelo lineal puede acoplar interacción, lo hace indirectamente. Como resultado, los cambios pueden causar confusión cuando el equipo del proyecto comienza.
2. A menudo es difícil que el cliente exponga explícitamente todos los requisitos. El modelo lineal secuencial lo requiere y tiene dificultades a la hora de acomodar la incertidumbre natural al comienzo de muchos proyectos.[3]

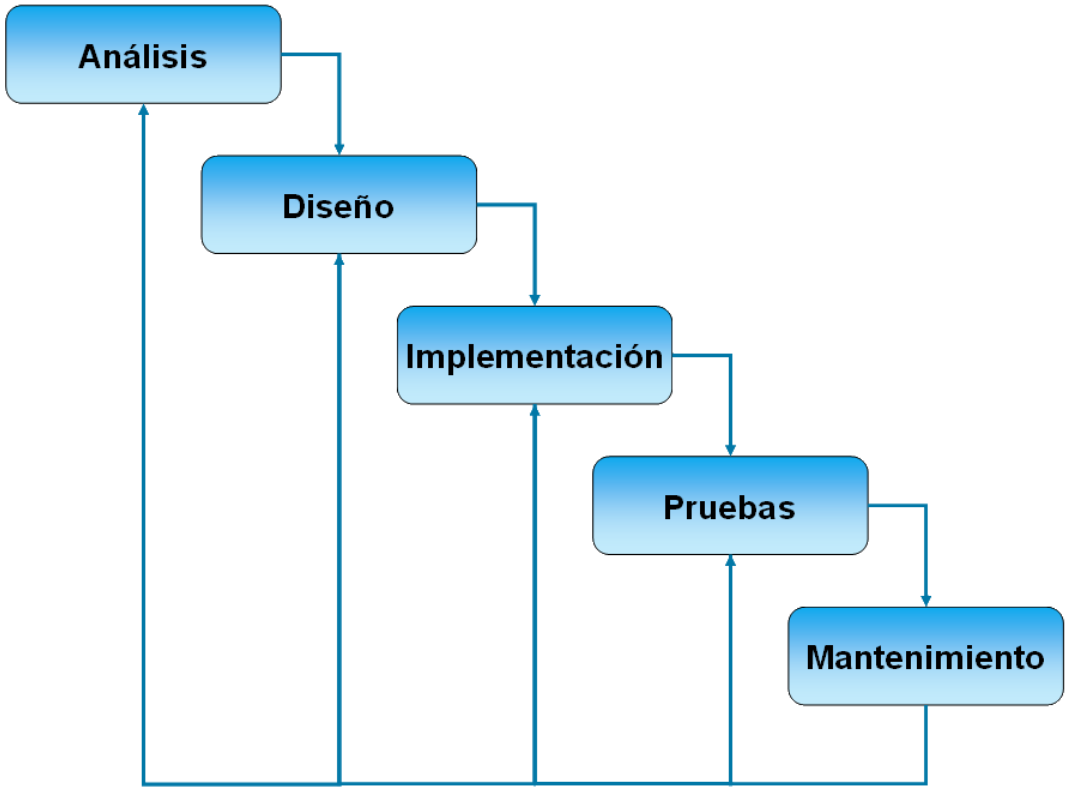

Figura 1.1. Modelo en cascada.

1.1.2 Modelo basado en componentes.

El modelo de desarrollo basado en componentes está basado en el paradigma de

[3] Ian Sommerville, *Ingeniería de software.* 2005 5ta ed. Pearson p. 62.

programación orientado a objetos, en el cual los datos y los algoritmos para procesar esos datos se encapsulan en clases. Si estas clases son diseñadas e implementadas de forma correcta pueden ser reutilizadas en cualquier proyecto de desarrollo. El modelo también sigue el patrón de comportamiento de un modelo evolutivo por lo que tiene en cuenta la naturaleza misma del software, ver Figura 1.2. Se identifican 4 actividades:

1. *Análisis de componentes*. Dada la especificación de requerimientos, se buscan los componentes para implementar la especificación. Por lo general, no existe una concordancia exacta y los componentes que se utilizan sólo proporcionan una funcionalidad parcial requerida.

2. *Modificación de requerimientos*. En esta etapa, los requerimientos se analizan utilizando información acerca de los componentes que se han descubierto. Entonces, estos componentes se modifican para reflejar los componentes disponibles. Si las modificaciones no son posibles, la actividad de análisis de componentes se puede llevar a cabo nuevamente para buscar soluciones alternativas.

3. *Diseño de sistemas con reutilización*. En esta fase se diseña o se reutiliza un marco de trabajo para el sistema. Los diseñadores tienen en cuenta los componentes que se reutilizan y organizan el marco de trabajo para que los satisfaga. Si los componentes reutilizables no están disponibles, se puede tener que diseñar nuevo software.

4. *Desarrollo e integración*. Para crear el sistema, el software que no se puede adquirir externamente se desarrolla, y los componentes y los sistemas comerciales se integran. En este modelo, la integración de sistemas es parte del proceso de desarrollo, más que una actividad separada.[4]

El modelo basado en componentes presenta muchas ventajas, reduce el desarrollo de software nuevo al reutilizar código, por lo tanto reduce los tiempos de entrega ya que no hay que reescribir todo el código y reduce los costos al requerir un menor esfuerzo por parte de los programadores. Sin embargo, si no se lleva una correcta administración del proceso se puede perder el control sobre la evolución del sistema y puede dar lugar a un sistema que no cumpla al 100% los requisitos del cliente.

[4] Ian Sommerville, *Ingeniería de software.* 2005 5ta ed. Pearson p. 65.

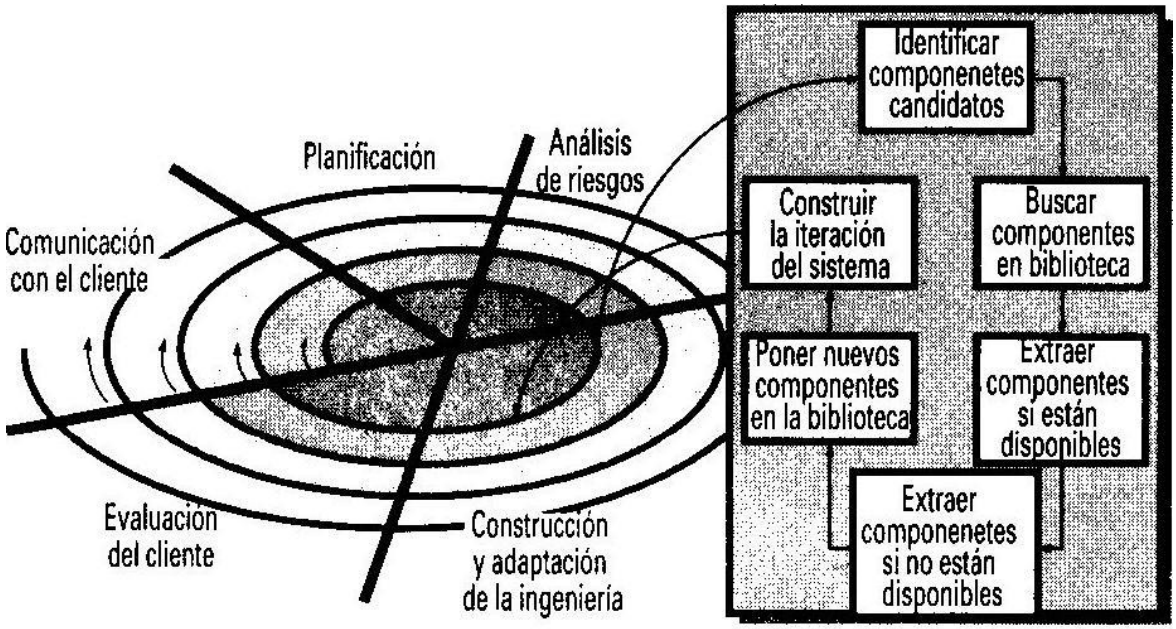

Figura 1.2. Modelo basado en componentes.

1.1.3 Modelo evolutivo o modelo ágil.

Ya que los requisitos del entorno del software y del mismo software cambian, se requiere de un modelo que tenga en cuenta la evolución del software. El modelo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios de los usuarios y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado. La ventaja de un proceso de software que se basa en un proceso evolutivo es que la especificación se puede desarrollar de forma creciente. Una ventaja de este modelo es que se desarrollan versiones funcionales, más no completas, del software aliviando presión sobre la gestión del proceso. Además el cliente puede ver el desarrollo del software e ir modificando el comportamiento y las vistas del mismo. Este modelo es recomendable cuando el sistema es de tamaño medio (500,000 líneas de código), y es mucho más recomendable que el modelo de cascada, ver Figura 1.3.

Aunque presenta muchas ventajas en comparación al modelo de cascada, presenta algunos problemas:

1. Si el sistema se desarrolla de forma rápida, no es rentable producir documentos del mismo. Por lo tanto la documentación es poca o nula.

2. Los cambios frecuentes corrompen la estructura del software. Conforme avanza el proceso se vuelve cada vez más difícil incorporar cambios.

Figura 1.3. Modelo Ágil o Evolutivo.

1.1.3.1 Proceso Unificado de desarrollo de software.

El proceso unificado de desarrollo de software (UP por sus siglas en inglés) es un paradigma de desarrollo de software, que a diferencia de otros paradigmas, toma en cuenta los cambios que la especificación del software pueda sufrir, es más los cambios guían el desarrollo del software y son la parte central del proceso. Estos cambios se especifican en casos de uso que son parte de la especificación del lenguaje unificado de modelado (por sus siglas en ingles UML). El UML es una parte esencial del Proceso Unificado. Dado que los cambios guían el proceso de desarrollo, éste también es iterativo e incremental desarrollando componentes de software reutilizable.

Entonces son 3 ideas claves las que componen el proceso:

1. Casos de uso
2. Centrado en la arquitectura
3. Iterativo e incremental

Por su naturaleza iterativa e incremental está basado en el enfoque orientado a objetos ya que éstos tienen la capacidad de cambiar y adaptarse sin necesidad de grandes cambios.

Por cada ciclo que se lleva a cabo en el Proceso Unificado se entrega un producto de software que aunque no es un producto terminado, si contiene funciones terminadas que deben ser evaluadas por los usuarios, ver Figura 1.4. Con esta evaluación y las opiniones de los usuarios el ingeniero de software comenzará un nuevo ciclo en el Proceso Unificado, a diferencia de otros métodos de desarrollo donde se trata de especificar todo antes de codificar y mostrar a los usuarios.

Este método requiere de la participación activa de los usuarios interesados en el software, ya que son ellos los que definen los cambios a seguir y si el proceso va por buen camino.

Su ciclo de diseño tiene 4 fases:

1. **Inicio**, durante esta fase se desarrolla una descripción del producto a entregar en ese ciclo y se presenta un análisis de negocio.
2. **Elaboración**, se especifica los casos de uso y se diseña la arquitectura del software.
3. **Construcción,** se codifican los casos de uso siguiendo la arquitectura.
4. **Transición,** los usuarios prueban el producto de software terminado en este ciclo y lo evalúan. Esta evaluación es entregada a los ingenieros de software que analizan las observaciones y cambios.

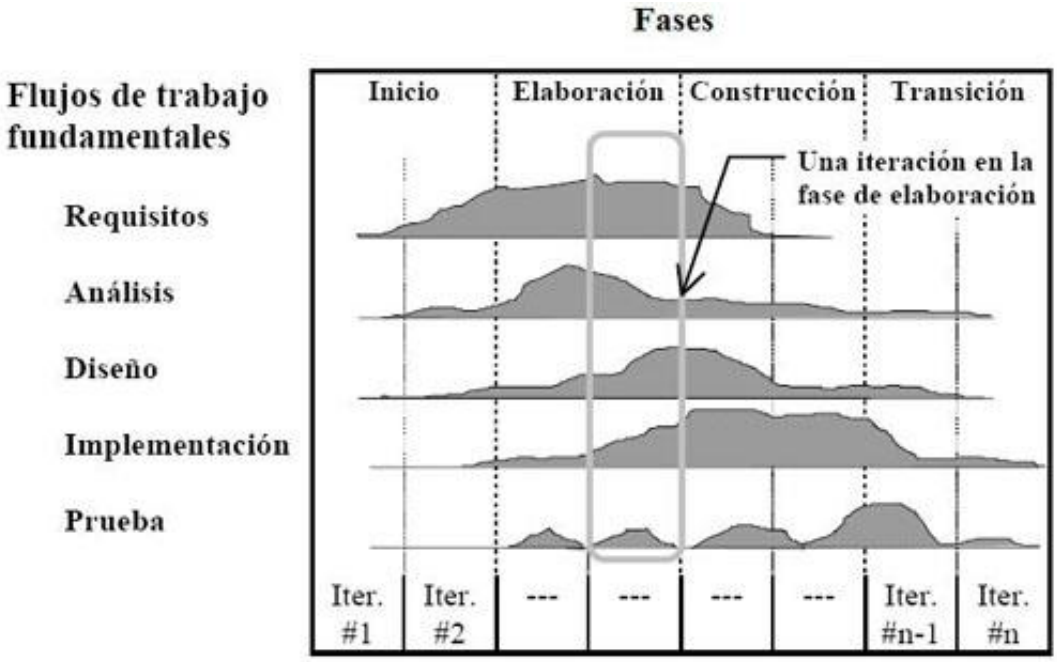

Figura 1.4. Fases de Proceso Unificado.

1.1.4 Arquitectura Modelo-Vista-Controlador.

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

El estilo fue descrito por primera vez en 1979 por Trygve Reenskaug, entonces trabajando en Smalltalk en laboratorios de investigación de Xerox. La implementación original está descrita en *Programación de Aplicaciones en Smalltalk-80(TM): Cómo utilizar Modelo Vista Controlador*.

- **Modelo.** Es la representación específica de la información con la cual el sistema opera.
- **Vista.** Este presenta el modelo en un formato adecuado para interactuar,

usualmente la interfaz de usuario.

- **Controlador.** Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo, y probablemente, a la vista.

En la Figura 1.5 se muestra un esquema de la arquitectura MVC.

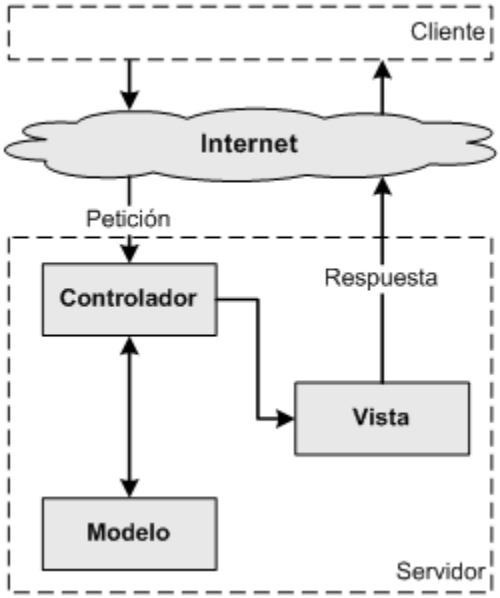

Figura 1.5. Arquitectura MVC.

1.2 Bases de datos.

Una base de datos es una colección o depósito de datos integrados y un conjunto de programas para acceder a ellos, almacenados en memoria no volátil, con redundancia controlada. Los datos han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cual ha de permitir captar las interrelaciones y restricciones en el mundo real.

Uno de los propósitos principales de las bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y

mantienen los datos.

1.2.1 Niveles de abstracción de una base de datos.

Dado que los sistemas de bases de datos son usados por diferentes usuarios con diferentes conocimientos en el área de las tecnologías de información, el sistema de bases de datos se divide en 3 niveles de abstracción que esconden la complejidad del sistema según se avanza a través de los niveles para simplificar la interacción de los usuarios con el sistema. Además la abstracción de la base de datos en estos tres niveles tiene como principal objetivo conseguir la independencia entre datos y aplicaciones:

Nivel físico. El nivel más bajo de abstracción describe cómo se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos complejas de bajo nivel, las estrategias de partición de datos, compresión de datos, encriptación, técnicas de ajuste o afinación (tunnig), etc.

Nivel lógico. El siguiente nivel de abstracción describe que datos se almacenan en la base de datos y que relaciones existen entre estos datos, así como las restricciones de integridad y confidencialidad. Los administradores de bases de datos, que deben decidir la información que se mantiene en la base de datos, usan el nivel lógico de abstracción.

Nivel de vistas. El nivel más alto de abstracción describe solo parte de la base de datos completa. A pesar del uso de estructuras más simples en el nivel lógico queda algo de complejidad, debido a la variedad de información almacenada en una base de datos. En el nivel de vistas deberán encontrarse reflejados sólo aquellos datos e interrelaciones que necesite el correspondiente usuario. Así el sistema puede proporcionar muchas vistas para diferentes usuarios de la misma base de datos.[5]

[5] Silberschatz, Abraham; Korth, Henry F.; S. Sudarshan; *Fundamentos de bases de datos.* 4ta ed. MCgraw Hill, 2002, p. 4.

1.2.2 Sistema manejador de bases de datos (SMBD).

Un sistema manejador de bases de datos es un conjunto coordinado de programas, procedimientos, lenguajes, etc., que suministran a los diferentes tipos de usuarios los medios necesarios para describir y manipular los datos almacenados en la base de datos, garantizando su seguridad.

1.2.3 Lenguajes de bases de datos.

Las distintas funciones que ha de cumplir un SMBD hacen necesario disponer de diferentes tipos de lenguajes y procedimientos que permitan la comunicación con la base de datos; unos están orientados hacia la función (definición o manipulación), y otros dirigidos a diferentes tipos de usuarios o de aplicaciones.

- **Lenguaje de definición de datos (LDD).** Se denomina lenguaje de definición de datos al instrumento que permite describir los datos con facilidad y precisión, especificando sus distintas estructuras.
- **Lenguaje de manipulación de datos (LMD).** Un lenguaje de manipulación de datos es un lenguaje que permite a los usuarios acceder o manipular los datos organizados mediante modelos de datos apropiados. Existen 2 tipos: procedimentales, requieren que el usuario especifique que se necesitan y cómo obtener estos datos; declarativos o no procedimentales, requieren que el usuario especifique qué datos se necesitan sin especificar cómo obtener estos datos.
- **Lenguaje de control de datos (LCD).** Las exigencias respecto a la forma de utilizar la base de datos son muy diferentes según los tipos de procesos y según los usuarios, siendo preciso que el lenguaje de control de datos responda a todas ellas. En especial, este lenguaje debe integrar una serie de instrumentos que faciliten las tareas de administración principalmente las relacionadas con la seguridad física y de protección frente a accesos no autorizados.[6]

[6] Silberschatz, Abraham; Korth, Henry F.; S. Sudarshan; *Fundamentos de bases de datos.* 4ta ed. MCgraw Hill, 2002, p. 7.

1.2.4 Modelos de datos.

El modelo de datos es una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de una base de datos. Una opción bastante usada a la hora de clasificar los modelos de datos es hacerlo de acuerdo al nivel de abstracción que presentan:

- Modelos de datos conceptuales. Son orientados a la descripción de estructuras de datos y restricciones de integridad. Se usan fundamentalmente durante la etapa de análisis de un problema dado y están orientados a representar los elementos que intervienen en este problema y sus relaciones. El ejemplo más típico es el Modelo Entidad-Relación.
- Modelo de datos lógicos. Son orientados a las operaciones más que a la descripción de una realidad. Usualmente están implementados en algún manejador de bases de datos. El ejemplo más típico es el modelo relacional, que cuenta con la particularidad de contar también con buenas características conceptuales.
- Modelos de datos físicos. Son estructuras de datos a bajo nivel implementadas dentro del propio manejador. Ejemplos de esas estructuras son los Arboles B+, las estructuras hash.

1.2.4.1 Modelo Entidad-Relación.

Los modelos de datos convencionales no ofrecen la suficiente capacidad de abstracción ni el poder expresivo como para captar la semántica del mundo real, haciendo difícil la comunicación del diseñador con el usuario. Un modelo es propuesto a fin de superar estos problemas, el modelo entidad-relación, propuesto por Peter Chen. Según Chen “el modelo entidad-relación puede ser usado como base para una vista unificada de los datos”, adoptando “el enfoque más natural del mundo real que consiste en entidades y relaciones”.

Una entidad es una cosa u objeto en el mundo real que se distingue de otros objetos. Las entidades se describen mediante un conjunto de atributos y una relación es una asociación entre varias entidades.

1.2.4.2 Modelo Relacional.

El modelo relacional es un ejemplo de un modelo basado en registros. Fue propuesto por E.F. Codd en los laboratorios IBM en California, USA. Se trata de modelo lógico que establece una estructura de registros sobre los datos, aunque posteriormente puedan ser almacenados de múltiples formas para aprovechar características físicas concretas de la máquina sobre la que se implementa la base de datos. Cada tipo de registro define un número fijo de campos, o atributos. Los registros son ordenados en tablas y las columnas de la tabla corresponden a los atributos del tipo de registro.

El modelo de datos relacional es el modelo de datos más ampliamente usado hoy en día.

1.3 Lenguajes de programación.

Un lenguaje de programación es un lenguaje artificial diseñado para comunicar instrucciones a una máquina, particularmente a las computadoras. Los lenguajes de programación son usados muy frecuentemente para crear programas que controlen el comportamiento de una máquina o para expresar algoritmos.

Los lenguajes de programación están compuestos por 2 componentes: la sintaxis, que es la forma en cómo se construyen las oraciones; y la semántica, que se refiere al significado del lenguaje de programación. Algunos lenguajes son definidos por un documento de especificación, mientras que otros como perl, tienen una implementación dominante que sirve como referencia.

1.3.1 Lenguajes procedimentales o imperativos.

En las ciencias de la computación el lenguaje procedimental o imperativo es un paradigma de programación que describe cómputos o programas en términos de sentencias que cambian el estado del programa. Los programas imperativos definen comandos que la máquina debe ejecutar.

La implementación de hardware de casi cualquier computadora es imperativo; y casi todas las computadoras están diseñadas para ejecutar código máquina, que está escrito en

estilo imperativo. Desde la perspectiva de bajo nivel, el estado del programa es definido por el contenido de la memoria, y las sentencias son las instrucciones en el lenguaje máquina de la computadora. Lenguajes imperativos de alto nivel usan variables y sentencias más complejas, pero aún siguen el mismo paradigma.

Los primeros lenguajes imperativos fueron los lenguajes máquina de las primeras computadoras. En estos lenguajes, las instrucciones eran muy simples, que hacían la implementación del hardware más fácil, pero impedían la creación de programas más complejos. FORTRAN, desarrollado por John Backus en IBM en 1954, fue el primer lenguaje de programación en superar el obstáculo presentado por el lenguaje máquina en la creación de programas complejos.

1.3.2 Lenguajes declarativos.

Los lenguajes de programación declarativos son aquellos donde se expresa la lógica de lo que se quiere obtener más no de cómo llegar al resultado deseado. Esta diferencia con los lenguajes procedimentales permite a los usuarios de lenguajes declarativos realizar su trabajo con un mínimo de código, esto gracias a que el programa describe qué es lo que se quiere en vez de cómo se quiere llegar al resultado. Entre los objetivos de este tipo de lenguajes se encuentra el eliminar los efectos colaterales de la programación funcional. Algunos ejemplos de lenguajes son: Haskell, Lisp, Clojure, SQL, etc.

Dentro de estos lenguajes se encuentra Ruby, aunque implementa varios paradigmas de programación es capaz, implementando programación declarativa, de construir aplicaciones web fácilmente usando sólo algunas cuantas instrucciones. Sin embargo no tiene mucha aceptación en el campo de aplicación y el registro en línea que se basa en el SALUAR está programado en lenguaje Java orientado a objetos.

1.3.3 Lenguajes orientados a objetos.

Los lenguajes orientados a objetos se centran en representar la información en forma de clases que contiene datos y métodos para manipular dichos datos. Son de los tipos de

lenguajes más extensamente usados en la industria por su filosofía de representar el mundo real en forma de cápsulas que contiene la información pertinente. Los objetos permiten reutilizar código o extender su funcionamiento ya que una vez que algo es representado sus datos y funciones pueden ser utilizados en cualquier programa que lo solicite.

También permiten modularizar los sistemas de software con lo que su mantenimiento, actualización y modificación es realmente fácil.

De entre estos lenguajes se encuentra Java, que se ha convertido en el lenguaje más utilizado para desarrollar aplicaciones web. Java implementa la mayoría de las características teóricas de los lenguajes orientados a objetos. Además al ser libre y de código abierto cualquiera puede aprender a usarlo, a excepción de una porción del código que aún es cerrado. Una de las características de Java es que al ser compilado no genera código para alguna arquitectura en particular, sino genera código intermedio denominado "código binario java" que corre en una máquina virtual. De esta manera ya no se requiere recompilar el software cada vez que se cambia de arquitectura pues se crea una máquina virtual para cada una de ellas.

Ya que el sistema de SALUAR fue diseñado en un principio con marco de trabajo basado en Java, y que ese desarrollo se puede reutilizar, cumpliendo de esta manera con uno de los objetivos de la ingeniería del software, decidí continuar desarrollando en este lenguaje.

Por sus características escogí a la arquitectura MVC, se ajusta a el desarrollo de las aplicaciones web, ya que su separación de vista del código de ejecución principal permite trabajar simultáneamente sin que el código se vea afectado por la vista y al revés, que la vista se vea entorpecida por el código. Además de que se respeta la visibilidad de las funciones principales.

El Proceso Unificado de desarrollo de software es un método que permite a proyectos como este generar solo la documentación necesaria, además muestra a los usuarios finales como se va desarrollando el software y les permite decidir si es lo que ellos necesitan, es por estas razones que lo utilice para el presente trabajo.

Capítulo 2
Registro en línea.

En este capítulo se presenta el proceso de desarrollo del registro en línea, en las siguientes secciones se presenta la captura de los nuevos requisitos para la ampliación de la funcionalidad del SALUAR original, en los capítulos siguientes se define el análisis, diseño e implementación del nuevo desarrollo y al final se presentan las pruebas efectuadas a este nuevo sistema.

2.1 Visión.

Facilitar a toda la comunidad académica universitaria el registro para el servicio de Acceso Remoto a los recursos electrónicos que a través de la DGB UNAM puede aprovechar desde los estudiantes de iniciación universitaria, para sus tareas escolares, hasta los investigadores para su trabajo diario de investigación.

2.2 Nuevos requerimientos.

Para el desarrollo de la nueva funcionalidad del registro en línea se requiere un proceso de desarrollo de software, el cual permite tener una planeación y control del desarrollo. Durante el proyecto se utiliza el Proceso Unificado, ya que cuenta con las actividades necesarias para transformar los nuevos requerimientos en software útil para el usuario.

El UP es un marco de desarrollo de software incremental e iterativo guiado por los casos de uso y centrado en la arquitectura.

Por su naturaleza el UP permite al desarrollador de software ir mostrando a los clientes y usuarios cómo funciona el sistema en etapas tempranas y a partir de ello refinar el diseño y volver a desarrollar; así el cliente puede evaluar el desarrollo y descubrir que es lo que en realidad necesita.

Como se mencionó anteriormente el SALUAR es un sistema de administración local para usuarios de Acceso Remoto. Sin embargo por sus características la mayor parte de la comunidad universitaria no tenía acceso a los materiales electrónicos que se pueden acceder a través de la

DGB, y a la cual dicha comunidad tiene derecho.

2.3 Descripción del sistema.

Según lo observado en el funcionamiento del SALUAR el usuario pierde tiempo en traslados a su biblioteca correspondiente a su facultad o instituto donde puede tramitar el registro al servicio de Acceso Remoto, trámites de documentos, disponibilidad de horario, etc. Para poder ampliar el acceso a toda la comunidad universitaria se requiere que el registro al Acceso Remoto no dependa del horario del administrador local, que por algún medio de comunicación estuviera disponible para su uso desde cualquier punto del país o en su caso del mundo. La Internet que desde sus inicios tiene como principal objetivo el poder compartir recursos desde cualquier parte del mundo cumple con la característica requerida, permite al SALUAR agregar nuevas funcionalidades habilitando a la comunidad universitaria desde cualquier punto geográfico del mundo que tenga una conexión a Internet con la posibilidad de registrarse al servicio de Acceso Remoto.

El usuario se ve limitado a solicitar su contraseña de Acceso Remoto por los horarios. Si se toma como referencia el punto anterior al escoger a la Internet como medio de comunicación por el cual los usuarios podrán tanto solicitar como usar el sistema de Acceso Remoto, nos lleva a que el nuevo registro en línea estará también basado en los sistemas informáticos, base de la propia Internet, que cuentan con alta disponibilidad y en el caso de los grandes servidores que trabajan las 24 horas del día, los 365 días del año.

Por lo tanto la nueva funcionalidad del sistema automatiza el registro de los usuarios del servicio de Acceso Remoto, permite recuperar la contraseña en caso de pérdida u olvido y permite al usuario actualizar o cambiar el correo electrónico y la contraseña del usuario, dado que son las operaciones más importantes que el usuario requiere para poder acceder a los recursos electrónicos. El nuevo sistema también permite a la Secretaría Técnica de Biblioteca Digital actualizar la información de vigencia de los alumnos registrados en el servicio de Acceso Remoto, de forma trasparente para el usuario, y así poder ampliar la vigencia o eliminar a aquel

alumno que ya no pertenezca a la comunidad universitaria. El sistema debe operar todos los días a todas horas, debe mostrarse correctamente en los navegadores de Internet más populares (Mozilla Firefox, Google Chrome, Internet Explorer, Safari, Opera, etc.), debe mostrar una interfaz de usuario de fácil manejo, además debe facilitar el registro a los usuarios enviando su contraseña generada vía correo electrónico.

Los usuarios permitidos a usar el servicio de Acceso Remoto son los alumnos que cursan en alguno de los diferentes niveles y planteles de la UNAM, y los académicos, ya sean de carrera o sólo de asignatura, que laboran actualmente en la UNAM.

Para recuperar la contraseña de Acceso Remoto se debe proporcionar a través del portal de la DGB sólo el número de cuenta de alumno o el número de trabajador de la UNAM.

Los datos para validar la identidad de los alumnos y permitirle su registro al Acceso Remoto deben ser fáciles de recordar y fuertemente asociados a la UNAM, en este caso su número de cuenta, el nivel educativo que cursa, la dependencia de la UNAM a la que pertenece y la carrera que cursa. Elementos comunes a todos los alumnos de la UNAM ya sean de iniciación universitaria hasta doctorado. Todos los alumnos que pretendan registrarse en el Acceso Remoto deberán tener sus derechos universitarios vigentes, esto es estar inscritos según los planes y programas de estudio de cada dependencia.

Para validar la identidad de los académicos de la UNAM se utilizara el Registro Federal de Causantes o RFC con la homoclave única que el Servicio de Administración Tributaria (SAT) del gobierno federal asigna a todos los profesionistas o personas que realicen alguna actividad que genere ingresos monetarios. Se escogió este dato por tener la característica de identificar unívocamente a todo el personal académico de la UNAM y por homologación con el portal de oficina virtual. También tendrá que discernir entre personal administrativo y académico ya que por el momento solo está disponible a los académicos de la UNAM.

Para actualizar la contraseña o el correo electrónico el usuario debe introducir la cuenta y la contraseña actuales. Los cambios deben ser notificados vía correo electrónico.

Para actualizar la vigencia de los usuarios (alumnos y académicos) se usan los datos registrados en la base de datos como es cuenta de alumno, cuenta de académico, RFC, etc. Esta actualización de vigencia se hará cada 6 meses para los alumnos y cada año para los académicos.

De acuerdo a lo anterior el sistema SALUAR tendrá 5 módulos nuevos:

1. Módulo de registro para alumnos
2. Módulo de registro para académicos
3. Módulo de recuperación de contraseña
4. Módulo de cambio de contraseña
5. Módulo de actualización de datos y vigencia.

A continuación se listan los requerimientos que después de haber visto las características actuales del SALUAR se infieren:

1. Minimizar los tiempos para solicitar y entregar la contraseña de Acceso Remoto.
2. Que el registro de usuarios a Acceso Remoto este siempre disponible y no dependa de los horarios de los empleados de la biblioteca.
3. El registro se debe poder hacer desde cualquier punto del mundo, evitando que el usuario se desplace físicamente a la biblioteca.
4. El registro debe validar que el solicitante de la contraseña a Acceso Remoto es un miembro de la comunidad universitaria académico o alumno vigente y con sus derechos universitarios vigentes.

5. Que el registro sea de fácil realización y disponible para cualquier sistema de cómputo.
6. Asignar automáticamente fecha de fin de vigencia según sea alumno o académico.
7. Recibe la información que la DGAE y la DGP proporcionan acerca de los alumnos y académicos para su posterior procesamiento.
8. Enviar la contraseña generada vía correo electrónico al alumno o académico interesado.
9. Recuperar de manera fácil la contraseña en caso de pérdida.
10. Que el usuario cambie su contraseña asignada por una de su preferencia pero que cumpla con las reglas de seguridad. Además de poder actualizar su correo electrónico.
11. La Secretaría Técnica de Biblioteca Digital debe actualizar la vigencia a los alumnos inscritos en el Acceso Remoto evitando que éstos se registren cada semestre y llevando un control de aquellos a los cuales su periodo de vigencia ha expirado.

En el siguiente capítulo se presentan los casos de uso que muestran los escenarios donde la nueva aplicación funciona, dado que se adoptó el Proceso Unificado de desarrollo de software y este a su vez utiliza al Lenguaje de Modelado Unificado (UML por sus siglas en inglés), sólo se muestran los documentos finales y no los diferentes documentos que se obtienen de cada una de las interacciones que se hacen a lo largo del desarrollo del sistema.

Capítulo 3
Análisis del registro en línea.

3.1 Casos de uso.

Los casos de uso son herramientas del UML (Unified Modeling Language) que permiten registrar y visualizar los requisitos funcionales que indican que hará el sistema. Captan funciones visibles para el usuario del sistema. Los casos de uso son documentos de texto más que diagramas.

A continuación se presentan los casos de uso que se obtuvieron en las reuniones para definir los requisitos:

Caso de uso CU1: Registrar nuevo Usuario.

Actor principal: Usuario (alumno).

Personal involucrado e intereses:

- Usuario: quiere registrarse para obtener acceso a los recursos digitales que resguarda la DGB a los cuales tiene derecho. Es alumno de la UNAM con sus derechos vigentes.
- Administrador local de Acceso Remoto: quiere que el registro del usuario sea sencillo y rápido de alta disponibilidad y un método de autenticación que demuestre que la persona que solicita el registro es un alumno de la UNAM con sus derechos vigentes.
- BiDi: quiere cumplir con su obligación de dar acceso a los recursos digitales a toda la comunidad académica universitaria de forma fácil y rápida.

Precondiciones: El usuario solicitante de contraseña de Acceso Remoto debe ser alumno de la UNAM con sus derechos vigentes.

Garantías de éxito (Postcondiciones): Se registra al usuario en la base de datos de Acceso Remoto. Se le hace llegar vía correo electrónico su contraseña con la cual acceder. Se visualiza

en los sistemas de SALUAR el registro del usuario.

Escenario principal de éxito (o flujo básico):

1. El usuario solicitante accede al portal de la DGB.
2. El usuario ingresa su información para validar su identidad y su vigencia, si es alumno o exalumno en la UNAM.
3. El sistema valida la vigencia y le pide al usuario proporcionar un correo electrónico de contacto.
4. El usuario proporciona correo electrónico de contacto y acepta los términos y condiciones de uso del servicio de Acceso Remoto.
5. El sistema registra la información del usuario, le asigna una vigencia y le comunica que su registro se ha llevado de manera exitosa y que su contraseña se enviara vía correo electrónico de contacto.
6. El usuario recibe el correo electrónico con su contraseña.

Extensiones (o flujos alternativos)

El asterisco * marca un evento que puede ocurrir en cualquier momento del flujo principal)

*a. En cualquier momento el sistema falla.

Para dar soporte a la recuperación y registro correcto, asegura que todos los estados y eventos significativos de una transacción puedan recuperarse desde cualquier paso del escenario.

1a. La BiDi a través de los encargados de los sistemas reinician el sistema y permiten que las funciones habituales se restablezcan.

1b. El sistema no responde, no se visualiza la página web y arroja un código de error.

1. El usuario se comunica a la Secretaría Técnica de la Biblioteca Digital.

2. El sistema envía un mensaje al usuario.

2a. El sistema detecta, al validar la información, que el usuario ya no tiene sus derechos

vigentes (porque es exalumno o tesista).

1. El sistema le comunica al usuario solicitante que sus derechos no están vigentes y que por tanto no puede tener acceso a los recursos electrónicos.

2b. El sistema detecta que la información proporcionada no es correcta y le pide al usuario verificar los datos ingresados.

1. El usuario verifica sus datos y los reingresa en el sistema.

3. El sistema le indica al usuario que el correo electrónico ingresado no es válido.

1. El usuario verifica nuevamente el correo electrónico y corrige.

4. El sistema le indica al usuario que aún no ha aceptado los términos y condiciones de uso del servicio.

1. El usuario acepta los términos y condiciones de uso del servicio.

5. El sistema le reporta al usuario que hubo un problema para registrar sus datos.

1. El sistema instruye al usuario a insertar de nuevo o si falla nuevamente comunicarse con la Secretaría Técnica de Biblioteca Digital.

6. El usuario no recibió su contraseña.

6a. El usuario registró un correo electrónico que ya no usa.

1. El usuario se comunica a la Secretaría Técnica de Biblioteca Digital para reportar que no recibió su contraseña.

6b. El usuario no ingresó correctamente su correo electrónico a pesar de la validación.

1. El usuario se comunica a la Secretaría Técnica de Biblioteca Digital para reportar que no recibió su contraseña.

Requisitos especiales.

- Interfaz web que se pueda visualizar en varias páginas web.

- Tiempo de respuesta corta, de 1 a 5 segundos.
- Envío de correo electrónico en menos de 15 minutos.
- Poder configurar el texto de términos y condiciones sin necesidad de detener la aplicación.
- Los datos de verificación del usuario debe ser datos comunes a todos los alumnos de la UNAM y que sólo ellos conozcan.

Lista de tecnología y variaciones de datos.

- La página web debe ser compatible con la mayoría de los navegadores web de hoy en día.
- Los datos que cualquier alumno de la UNAM conoce y que lo identifican unívocamente es el número de cuenta.
- De alguna forma validar que el usuario ingrese un correo electrónico válido.
- El servidor de correo debe ser capaz de enviar varios correos electrónicos a la vez.

Caso de uso CU2: Registrar nuevo usuario (académico).

Actor principal: Usuario (académico).

Personal involucrado e intereses:

- Usuario: quiere registrarse para obtener cuenta de acceso a los recursos digitales que resguarda la DGB a los cuales tiene derecho. Es académico de la UNAM con sus derechos vigentes.
- Administrador local de Acceso Remoto: quiere que el registro del usuario sea sencillo y rápido de alta disponibilidad y un método de autenticación que demuestre que la persona que solicita el registro es un profesor de la UNAM con sus derechos vigentes.
- BiDi: quiere cumplir con su obligación de dar acceso a los recursos digitales a toda la comunidad académica universitaria de forma fácil y rápida.

Precondiciones: El usuario solicitante de cuenta de Acceso Remoto debe ser profesor de la

UNAM con sus derechos vigentes.

Garantías de éxito (Postcondiciones): Se registra al usuario en la base de datos de Acceso Remoto. Se le hace llegar vía correo electrónico su contraseña con la cual acceder. Se visualiza en los sistemas de SALUAR el registro del usuario.

Escenario principal de éxito (o flujo básico):

1. El usuario solicitante accede al portal de la DGB.
2. El usuario ingresa su información para verificar su identidad y su vigencia, si es profesor activo o ya no labora más en la UNAM.
3. El sistema valida la vigencia y le pide al usuario proporcionar un correo electrónico de contacto.
4. El usuario proporciona correo electrónico de contacto y acepta los términos y condiciones de uso del servicio de Acceso Remoto.
5. El sistema registra la información del usuario, le asigna una vigencia y le comunica que su registro se ha llevado de manera exitosa y que su contraseña se enviara vía correo electrónico de contacto.
6. El usuario recibe el correo electrónico con su contraseña.

Extensiones (o flujos alternativos):

*a. En cualquier momento el Sistema falla.

Para dar soporte a la recuperación y registro correcto, asegura que todos los estados y eventos significativos de una transacción puedan recuperarse desde cualquier paso del escenario.

1a. La BiDi a través de los encargados de los sistemas reinician el sistema y permiten que las funciones habituales se reestablezcan.

1b. El sistema no responde, no se visualiza la página web y arroja un código de error.

1. El usuario se comunica a la Secretaría Técnica de la Biblioteca Digital.

2. El sistema envía un mensaje al usuario.

2a. El sistema detecta, al validar la información, que el usuario ya no tiene sus derechos vigentes (porque ya no labora en la UNAM, es jubilado, etc.).

1. El sistema le comunica al usuario solicitante que sus derechos no están vigentes y que por tanto no puede tener acceso a los recursos electrónicos.

2b. El sistema detecta que la información proporcionada no es correcta y le pide al usuario verificar los datos ingresados.

1. El usuario verifica sus datos y los reingresa en el sistema.

3. El sistema le indica al usuario que el correo electrónico ingresado no es válido.

1. El usuario verifica nuevamente el correo electrónico y corrige.

4. El sistema le indica al usuario que aún no ha aceptado los términos y condiciones de uso del servicio.

1. El usuario acepta los términos y condiciones de uso del servicio.

5. El sistema le reporta al usuario que hubo un problema para registrar sus datos.

1. El sistema instruye al usuario a insertar de nuevo o si falla nuevamente comunicarse con la Secretaría Técnica de Biblioteca Digital.

6. El usuario no recibió su contraseña.

6a. El usuario registro un correo electrónico que ya no usa.

1. El usuario se comunica a la Secretaría Técnica de Biblioteca Digital para reportar que no recibió su contraseña.

6b. El usuario no ingreso correctamente su correo electrónico a pesar de la validación.

1. El usuario se comunica a la Secretaría Técnica de Biblioteca Digital para reportar que no recibió su contraseña.

Requisitos especiales.

- Interfaz web que se pueda empotrar en varias páginas web.

- Tiempo de respuesta corta, de 1 a 5 segundos.
- Envío de correo electrónico en menos de 15 minutos.
- Poder configurar el texto de términos y condiciones sin necesidad de detener la aplicación.
- Los datos de verificación del usuario debe ser datos comunes a todos los profesores de la UNAM y que sólo ellos conozcan.

Lista de tecnología y variaciones de datos.

- La página web debe ser compatible con la mayoría de los navegadores web de hoy en día.
- Los datos que cualquier académico de la UNAM conoce y que lo identifican unívocamente es el número de empleado o su RFC con homoclave.
- De alguna forma validar que el usuario ingrese un correo electrónico valido.
- El servidor de correo debe ser capaz de enviar varios correos electrónicos a la vez.

Caso de uso CU3: Recuperar contraseña de Acceso Remoto.

Actor principal: Usuario (alumno o académico).

Personal involucrado e intereses:

- Usuario: quiere recuperar su contraseña de Acceso Remoto para poder continuar usando el servicio.
- BiDi: desea seguir prestando el servicio de Acceso Remoto y hacer más fácil la recuperación de acceso al usuario.

Precondiciones: El usuario debe estar registrado previamente en el servicio de Acceso Remoto a los recursos de la DGB.

Garantías de éxito (Postcondiciones): Vía correo electrónico el usuario recibe su contraseña para continuar usando el servicio de Acceso Remoto. El usuario recibe la contraseña en su correo electrónico.

Escenario principal de éxito (o Flujo Básico):

1. El usuario ingresa a la página web de la DGB y da clic en el link “recuperar contraseña”.
2. El usuario ingresa su cuenta de alumno UNAM o número de trabajador.
3. El sistema busca el registro del usuario en la base de datos.
4. El sistema avisa al usuario que su contraseña será enviada al correo electrónico proporcionado por el usuario al momento de registrarse.
5. El usuario recibe su contraseña en su correo electrónico.

Extensiones (o flujos alternativos):

1. La página web de la DGB no carga o se tarda en responder.

    1. El usuario se comunica a la Secretaría Técnica de Biblioteca Digital y avisa del incidente. El personal del departamento revisa y levanta nuevamente el sistema.

2. El usuario no ingresa ningún dato al sistema.

    1. El sistema le avisa al usuario que debe proporcionar su número de cuenta para llevar a cabo la búsqueda y recuperación de la contraseña.

3a. El sistema detecta que el usuario no está registrado en la base de datos de Acceso Remoto.

    1. El usuario realiza su registro al servicio de Acceso Remoto.

3b. El sistema detecta que el usuario ha ingresado un número de cuenta no valido.

    1. El usuario verifica su número de cuenta o número de trabajador y lo ingresa nuevamente.

Requisitos especiales.

- Tiempo de respuesta corta, de 1 a 5 segundos.

- Envío de correo electrónico en menos de 15 minutos.
- La contraseña no se muestra en pantalla por motivos de seguridad

Lista de tecnología y variaciones de datos.

- El número de cuenta debe ser el usuario/cuenta que se utilizará para recuperar la contraseña.
- El servidor de aplicaciones debe tener alta disponibilidad.

Caso de uso CU4: Cambiar contraseña.

Actor principal: Usuario.

Personal involucrado e intereses:

- Usuario, requiere cambiar su contraseña por cuestiones de seguridad o de facilidad.
- BiDi, desea que la operación de actualizar el correo electrónico del usuario sea fácil, rápida. y confiable.

Precondiciones: el usuario debe estar previamente registrado en el servicio de Acceso Remoto.

Garantías de éxito (Postcondiciones): el usuario recibirá vía correo electrónico una confirmación de su cambio de contraseña y un aviso informando de la situación.

Escenario principal de éxito (o flujo básico):

1. El usuario ingresa a la página de la DGB y da clic en el link "cambiar contraseña".
2. El sistema le pide ingresar cuenta de alumno UNAM o número de trabajador y contraseña actual.
3. El sistema busca el registro del usuario en la base de datos.
4. El sistema muestra al usuario su información resguardada en la base de datos, nombre completo, nivel, plantel, carrera, un enlace o botón para cambiar la contraseña ya, que la

contraseña no se debe mostrar.

5. El usuario pulsa o selecciona la opción de cambiar contraseña.
6. El sistema pide al usuario proporcionar una nueva contraseña con ciertas características y después confirmar la nueva contraseña.
7. El usuario pulsa aceptar, el sistema registra en la base de datos los cambios avisa al usuario que la operación fue exitosa y envía un correo electrónico al usuario avisando de un cambio a su contraseña.

Extensiones (o flujos alternativos):

1. La página de la DGB no carga o se tarda en responder.

   1. El usuario se comunica a la Secretaría Técnica de Biblioteca Digital y avisa del incidente. El personal del departamento revisa y levanta nuevamente el sistema.

2. El usuario no ingresa ningún dato al sistema.

   1. El sistema le avisa al usuario que debe proporcionar su número de cuenta para llevar a cabo la búsqueda y recuperación de la contraseña.

3a. El sistema detecta que el usuario no está registrado en la base de datos de Acceso Remoto.

   1. El usuario realiza su registro al servicio de Acceso Remoto.

3b. El sistema detecta que el usuario ha ingresado un número de cuenta no valido.

   1. El usuario verifica su número de cuenta o número de trabajador y lo ingresa nuevamente.

3c. El sistema detecta que el usuario se encuentra en lista negra por no haber cumplido con el reglamento y le informa que no tiene derecho a utilizar el servicio.

4. El sistema le reporta al usuario un error con la base de datos y sus datos no pueden ser desplegados.

   1. El usuario se comunica a la Secretaría Técnica de Biblioteca Digital y avisa del

incidente.

6a. El usuario ingresa una contraseña que no cumple con las características requeridas.

6b. El usuario ingresa incorrectamente la confirmación de su nueva contraseña.

7. El usuario pulsa cancelar y la operación de cambiar contraseña se cancela.

Requisitos especiales.

- Tiempo de respuesta corta, de 1 a 5 segundos.
- Envío de correo electrónico en menos de 15 minutos.
- La contraseña no se muestra en pantalla por motivos de seguridad

Lista de tecnología y variaciones de datos.

- El número de cuenta será el usuario/cuenta que se utilizará para cambiar la contraseña.
- La nueva contraseña debe cumplir con ciertas características como llevar números intercalados con letras, usar letras mayúsculas y minúsculas, usar ciertos caracteres especiales como #, %, &, @, etc. Deben definirse.
- El servidor de aplicaciones debe tener alta disponibilidad.

Caso de uso CU5: Actualizar vigencia.

Actor principal: BiDi.

Personal involucrado e intereses:

- Usuario, necesita seguir haciendo uso del servicio de Acceso Remoto.
- BiDi, requiere saber que usuarios del servicio de Acceso Remoto aún son vigentes para descartar aquellos que ya no tienen derecho al servicio y extender la vigencia a aquellos con derechos universitarios vigentes.

Precondiciones: la información de los usuarios debe estar previamente registrada en la base

de datos del servicio de Acceso Remoto.

Garantías de éxito (Postcondiciones): BiDi actualiza la vigencia del usuario para que pueda seguir haciendo uso del servicio o en dado caso de que ya no sea parte de la comunidad universitaria se le notifica que su acceso ha sido revocado, el usuario vigente no es notificado pero su acceso tampoco es interrumpido.

Escenario principal de éxito (o flujo básico):

1. BiDi pide al sistema verificar las cuentas de los alumnos y RFC de los académicos registrados en la base de datos de Acceso Remoto.
2. El sistema envía la información de los alumnos a la DGAE y los RFC a la DGP.
3. DGAE y DGP regresan la información de los alumnos al Sistema.
4. El sistema verifica la vigencia de los alumnos y de los académicos.
5. El sistema actualiza la vigencia de los alumnos y de los académicos.
6. El sistema genera un reporte de actualizados.
7. El sistema notifica, vía correo electrónico, a los usuarios que no son vigentes de su acceso será revocado.

Extensiones (o flujos alternativos):

1. El sistema no responde.
    1. BiDi reinicia el sistema
2. La DGAE o la DGP o ambas no responden de recibido.
    1. BiDi reenvía la información esperando por respuesta.
    2. Si no hay respuesta BiDi se comunica con una o con la otra o con ambas.
3. DGAE o DGP o ambas responden que la información es incorrecta o que no tiene el formato solicitado.
    1. BiDi verifica la información y la reenvía.

4. La información que envía DGAE o DGP o ambas no puede ser leída.

    1. BiDi se comunica con la DGAE o DGP o ambas.

    2. BiDi reenvía la información.

5. El sistema no puede acceder a la base de datos.

    1. El sistema notifica a BiDi que la base de datos no está disponible.

    2. BiDi verifica la base de datos y reintenta la actualización de datos.

Requisitos especiales.

- El sistema debe ser resistente a fallos, si se presenta algún error el sistema debe ser capaz de recuperarse.

Lista de tecnología y variaciones de datos.

- Para obtener la información actualizada de los alumnos se requiere enviar padrón de usuarios a la DGAE con número de cuenta, nivel de estudios, clave de plantel y clave de carrera.
- En el caso de los académicos es el RFC con homoclave el que se debe enviar a la DGP.
- Se actualiza la fecha de fin de vigencia en la tabla correspondiente de la base de datos de Acceso Remoto.

En la Figura 3.1 se muestra el diagrama general con los casos de uso presentados.

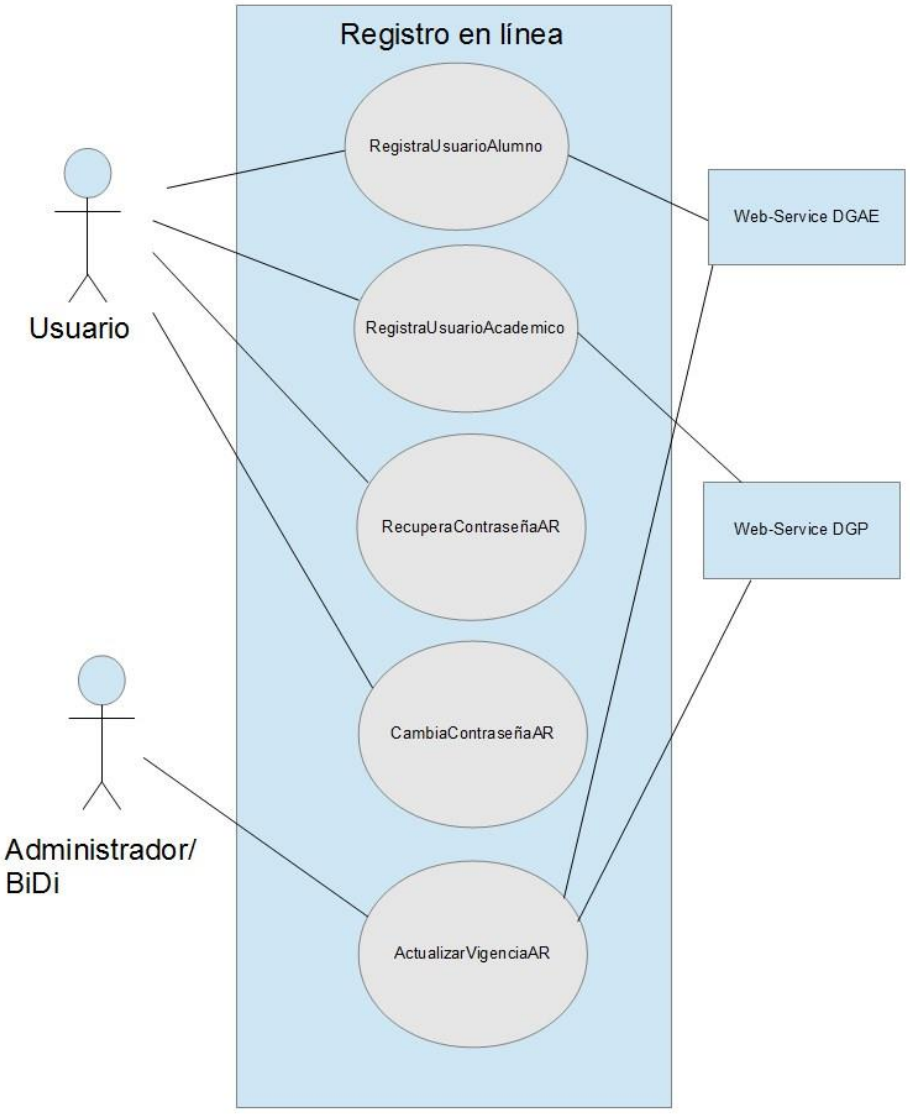

Figura 3.1 Diagrama de casos de uso.

3.2 Diagramas de secuencia de sistema.

Un diagrama de secuencia es un artefacto UML creado de manera rápida y fácil que muestra los eventos de entrada y salida relacionados con el sistema que se está estudiando. Muestra para un curso específico de eventos en un caso de uso los actores externos que interaccionan directamente con el sistema, el sistema (como caja negra) y los eventos del sistema que genera el actor. La secuencia avanza hacia abajo y deben seguir el orden según el caso de uso.

Los diagramas de secuencia, que se muestran de la Figura 3.2 a 3.6, modelan el comportamiento de la nueva funcionalidad de registro en línea, basados en los casos de uso anteriores en su escenario principal de éxito:

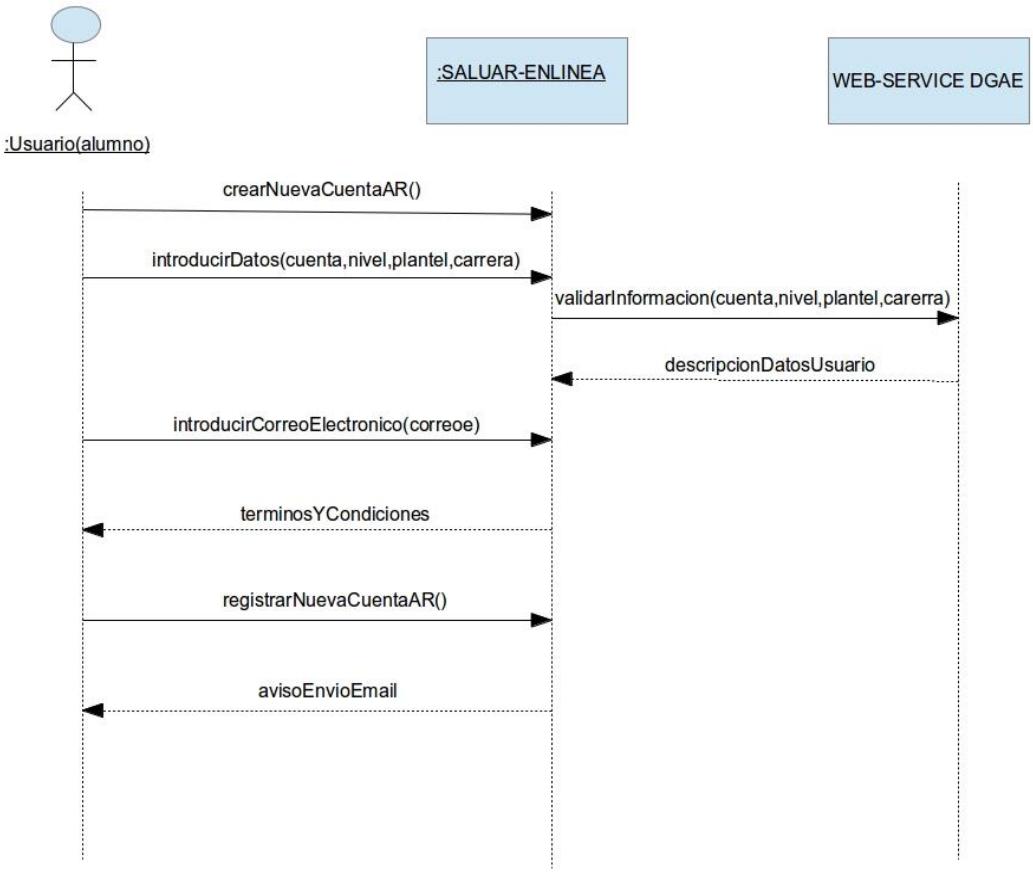

Figura 3.2 Diagrama de secuencia para el caso de uso CU1.

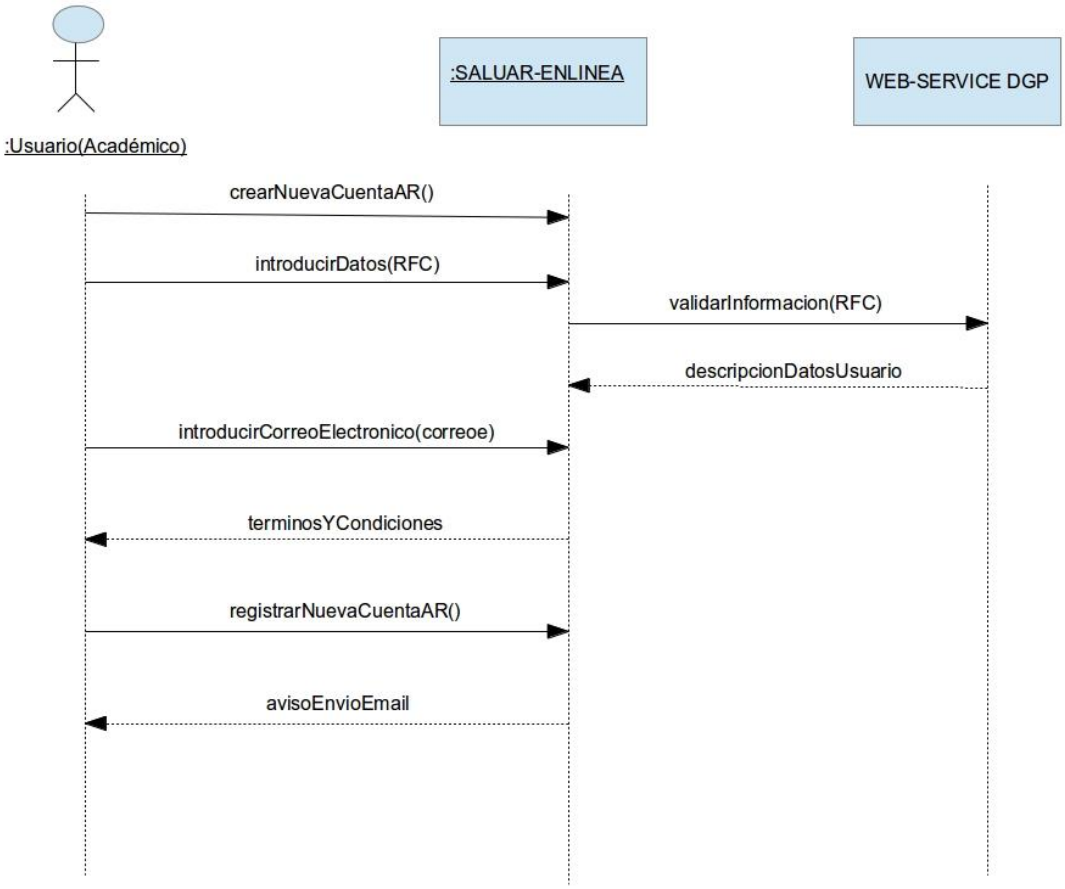

Figura 3.3 Diagrama de secuencia de caso de uso CU2.

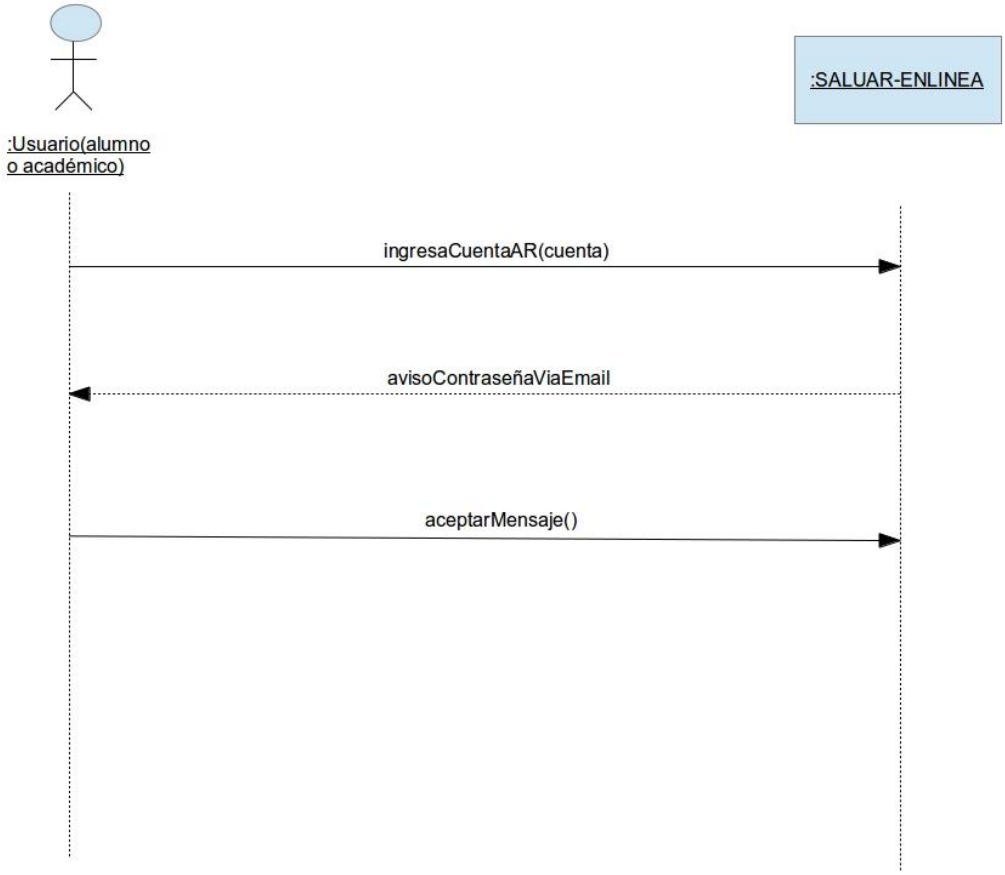

Figura 3.4 Diagrama de secuencia de caso de uso CU3.

Figura 3.5 Diagrama de secuencia de caso de uso CU4.

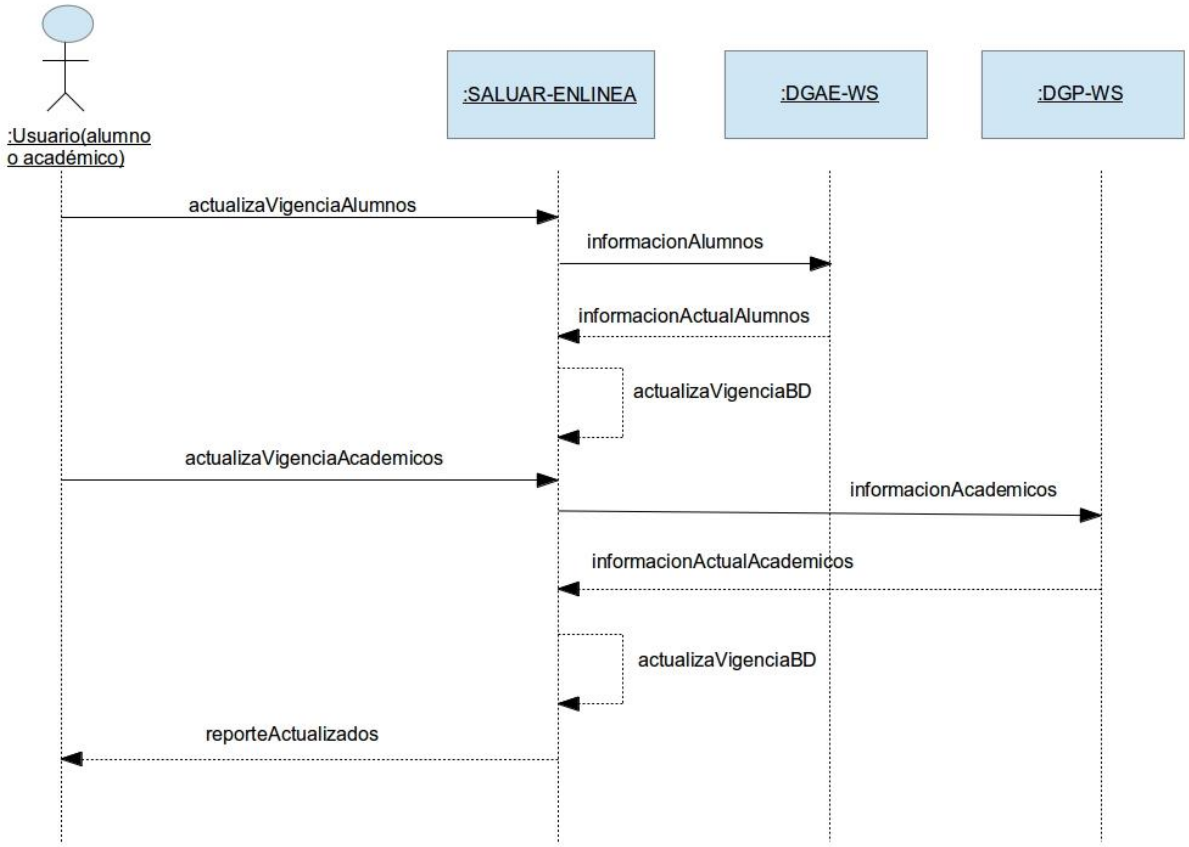

Figura 3.6 Diagrama de secuencia de caso de uso CU5.

El propósito de los casos de uso como los diagramas de secuencia es mostrar cómo funcionará el nuevo sistema de registro en línea para el acceso remoto. Se puede ver que los usuarios (académicos y alumnos) serán los únicos que interactúen con el sistema sin intervención de una tercera persona. El sistema dará una respuesta pronta a la petición del usuario y éste recibirá su contraseña a la brevedad posible, sin importar donde se encuentre o a qué hora del día realice su trámite, cumpliendo de esta manera con los objetivos planteados.

3.3 Diagrama de clases conceptuales o modelo de dominio.

Lista de categorías de clases conceptuales.

La siguiente es una lista de categorías donde se incluyen las clases conceptuales candidatas del dominio del Sistema de Administración Local de Usuarios de Acceso Remoto.

Categorías de clases conceptuales	**Ejemplos**
Transacciones	Registro, actualización
Roles de la gente	Alumno UNAM, académico UNAM, usuario vigente, Administrador BiDi
Otros sistemas informáticos	Web Service DGAE, Web Service DGP
Organizaciones	Facultades, Institutos, Direcciones Generales
Hechos	Acceso remoto, registro, actualización, recuperación
Procesos	RegistrarNuevoAlumno, RegistrarNuevoAcademico, RecuperarContraseña, CambiarContraseña, IniciaSesion, ActualizarVigenciaAlumnos, ActualizaVigenciaAcademicos
Catálogos	CatálogoDependencias, CatálogoCarreras
Lugares.	Facultades, Institutos, Direcciones Generales, Bibliotecas
Especificaciones, diseños o descripciones	InformacióndeAlumno, InformacióndeAcademico

Identificación de frases nominales.

La siguiente es una lista de clases conceptuales candidatas obtenida por medio de la técnica de frases nominales basados en los casos de uso descritos anteriormente.

- Usuario
- Vigencia
- InformaciónUsuario
- Registro
- PaginaWeb
- Cuenta
- Contraseña

Muchas de estas clases ya se encuentran en el esquema actual de la base de datos de Acceso Remoto, por lo tanto sólo se modelaran y no se tendrán en cuenta para el diseño o la implementación.

A partir del análisis de clases anterior el diagrama de clase conceptuales o modelo de dominio que representa al SALUAR queda como se muestra en la Figura 3.7, las clases que se agregan al modelo existente se presentan oscuras. La Figura 3.8 presenta las clases con sus atributos.

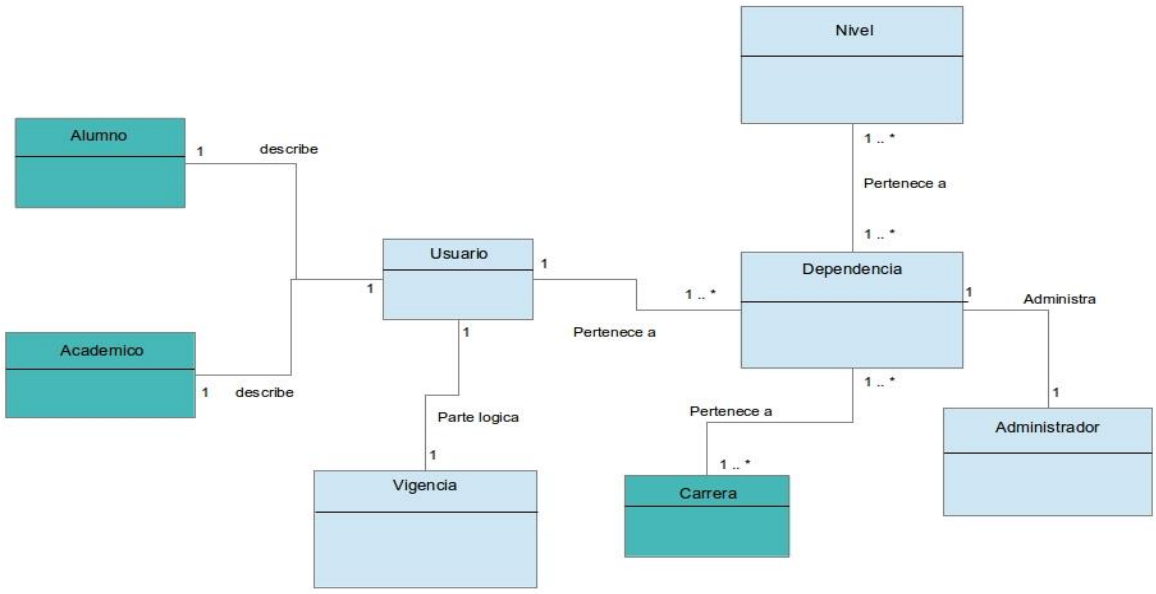

Figura 3.7 Diagrama de clases conceptuales.

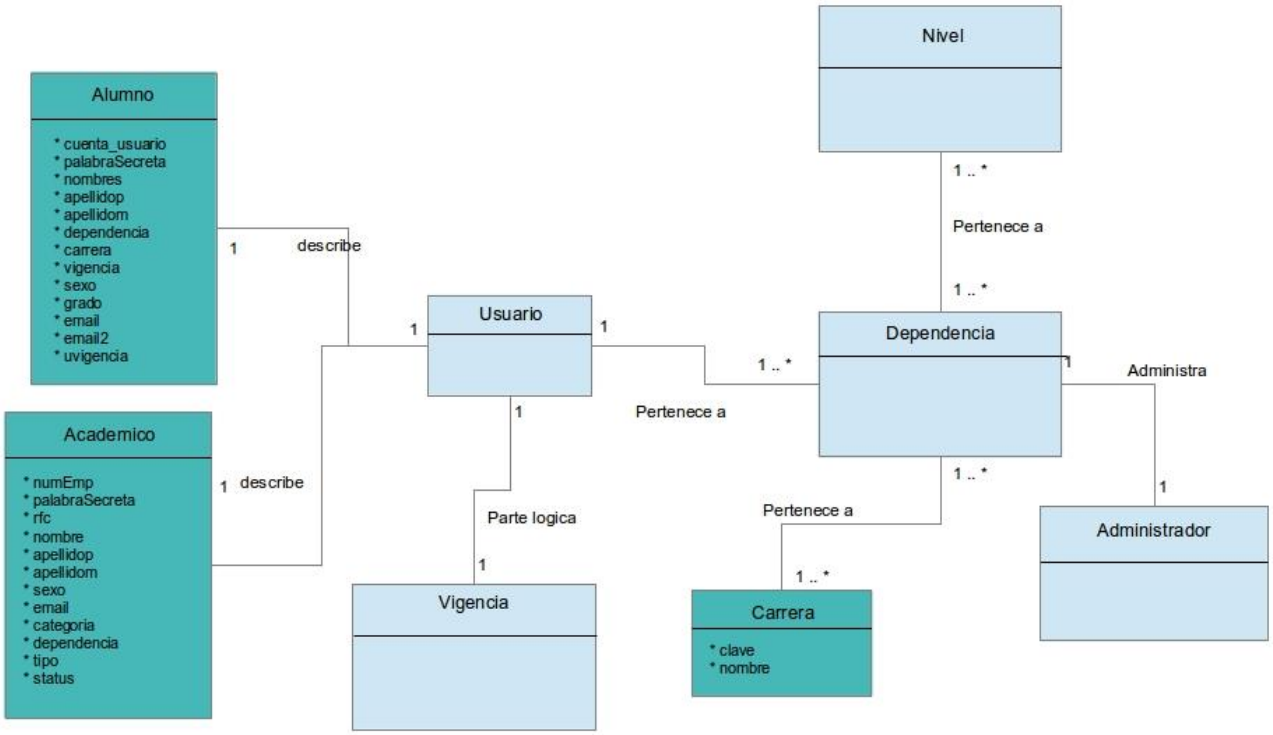

Figura 3.8 Diagrama de clases conceptuales con atributos

Capítulo 4
Diseño del registro en línea.

Dadas las características del sistema (alta disponibilidad, basado en web, acceso desde cualquier punto desde cualquier computadora con conexión a Internet, etc.) y el hecho de que el sistema de donde se deriva, el SALUAR, está hecho bajo el patrón Modelo-Vista-Controlador (MVC) se tomará éste como arquitectura de software a seguir.

4.1 Diagramas de interacción.

Los diagramas interacción son parte de la especificación del UML, ayudan con el diseño de las clases software, sus métodos y relaciones. Expresan de forma gráfica las relaciones y métodos de las clases que se comunican por medio de mensajes. Se apoyan sobre los casos de uso.

Las Figuras de la 4.1 a la 4.5 muestran los diagramas de interacción de cada uno de los casos de uso descritos en el capítulo 3.

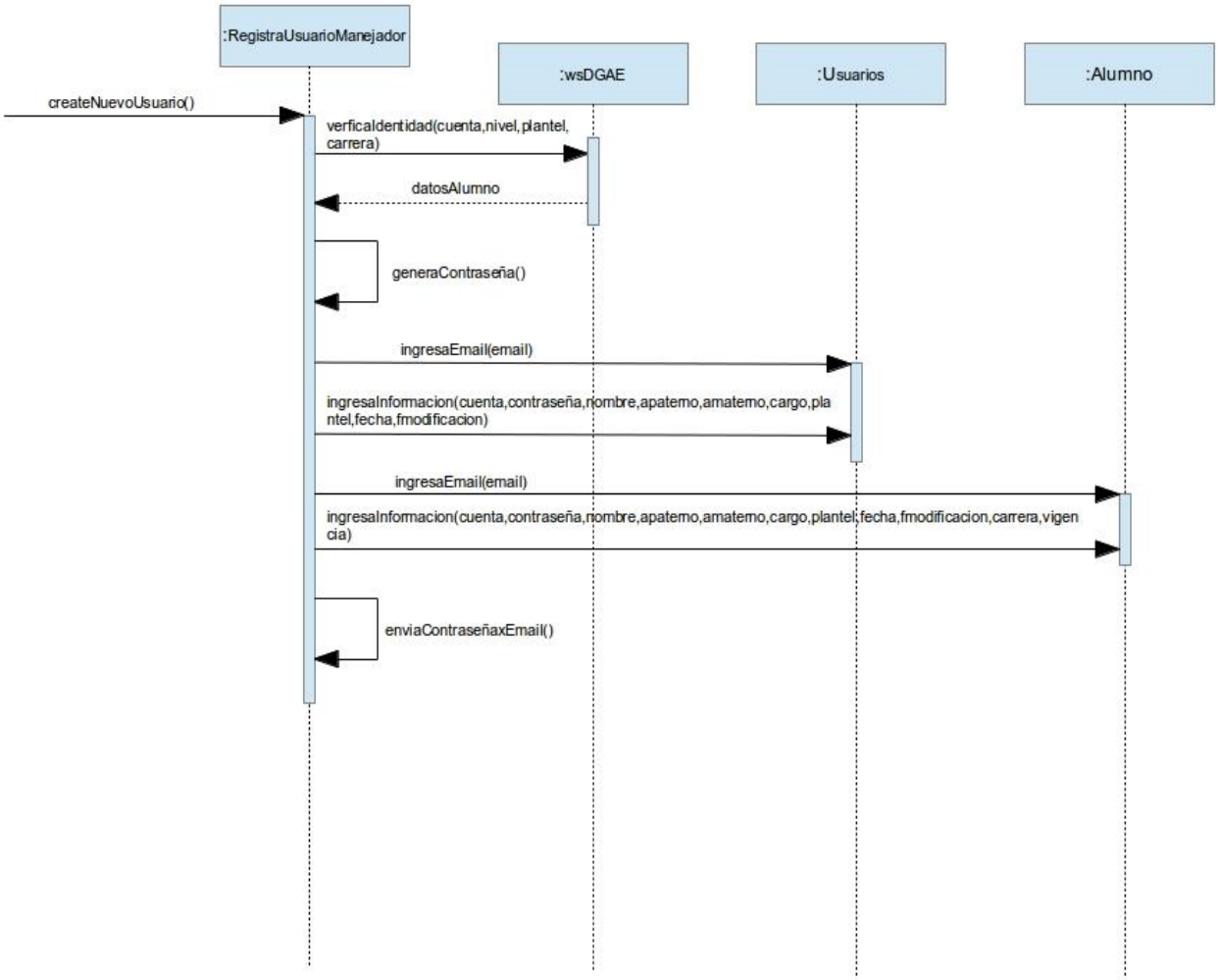

Figura 4.1. Diagrama de secuencia para caso de uso "Registra nuevo usuario (alumno)"

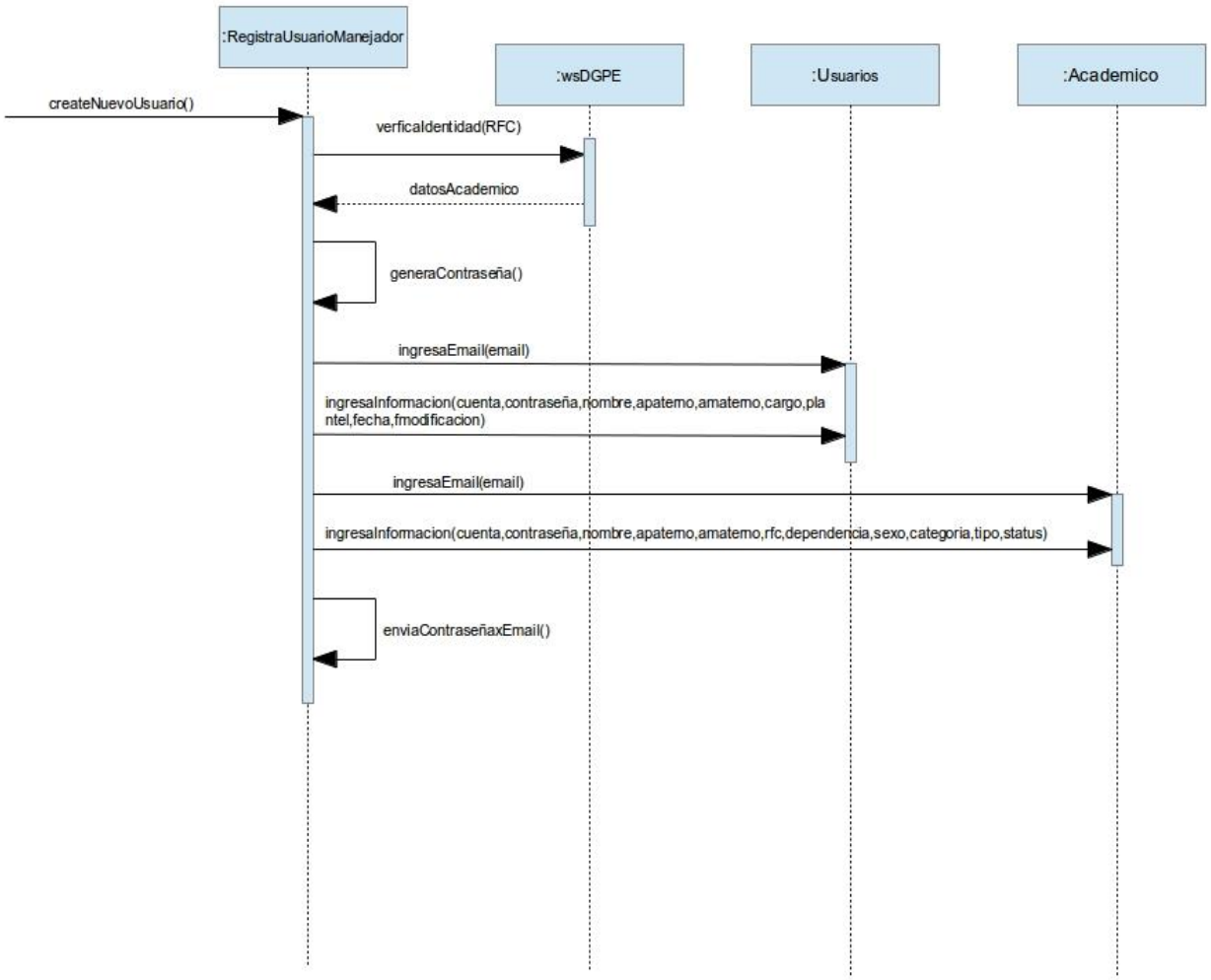

Figura 4.2. Diagrama de secuencia para caso de uso "Registra nuevo usuario (académico)"

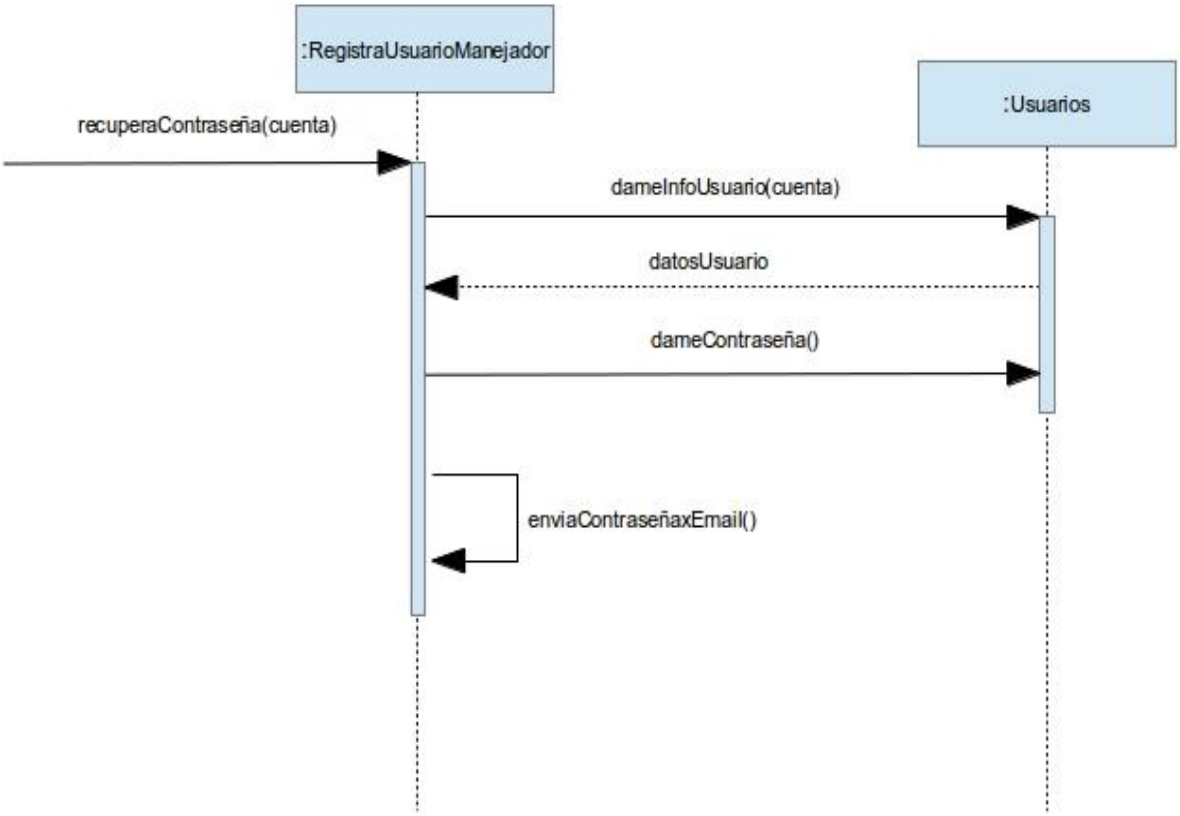

Figura 4.3. Diagrama de secuencia para caso de uso "Recupera contraseña"

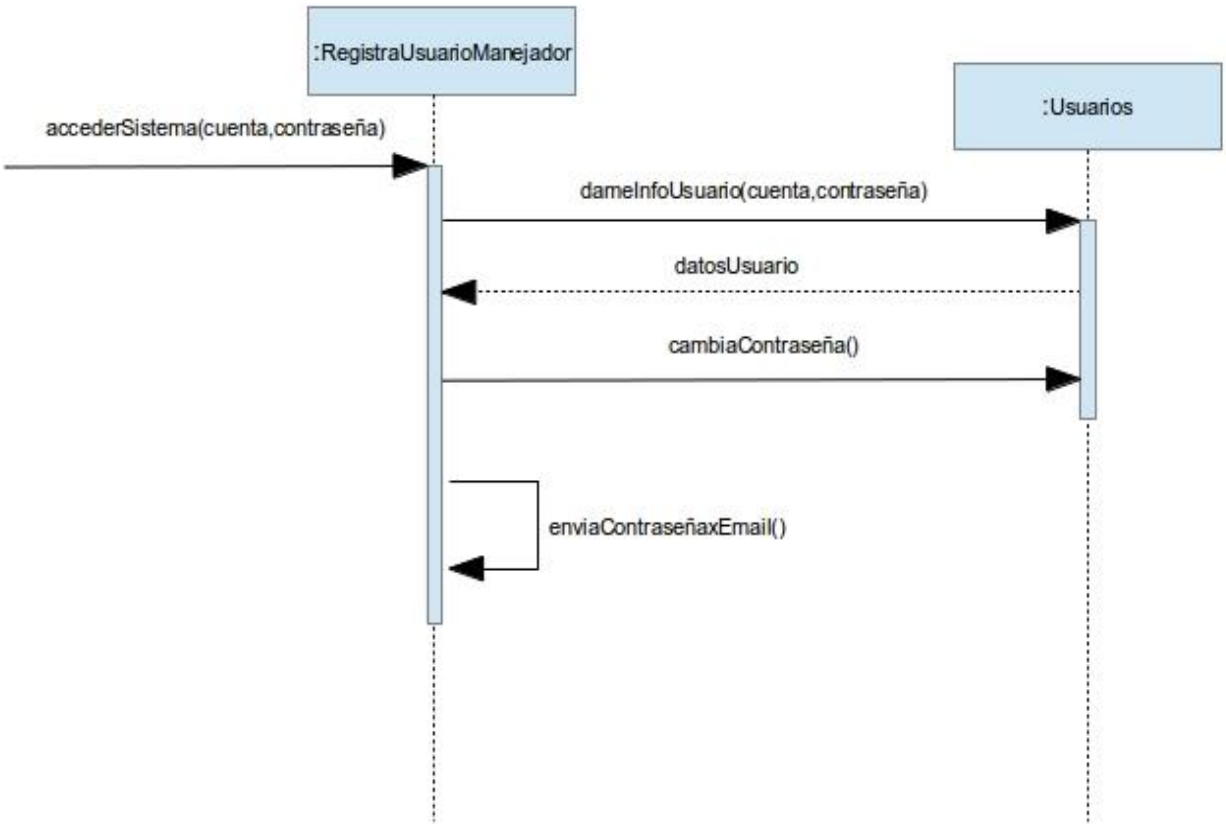

Figura 4.4. Diagrama de secuencia para caso de uso "Cambia contraseña"

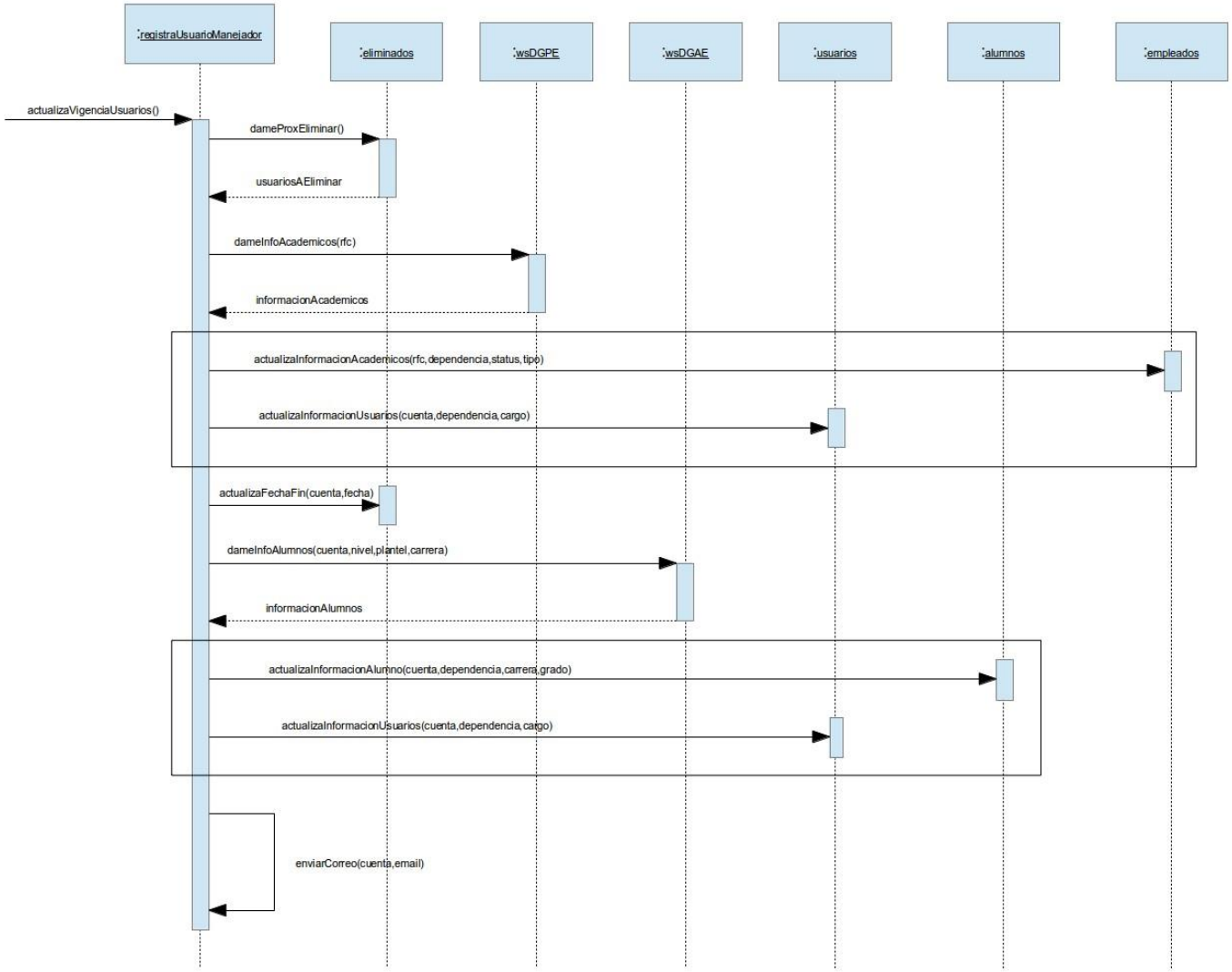

Figura 4.5. Diagrama de secuencia para caso de uso "Actualiza vigencia"

4.2 Lenguaje de programación.

Java es un lenguaje de programación creado por James Gosling y un grupo de ingenieros en Sun Microsystems a principios de la década de los 90's, el grupo se hacía llamar “Green Project”. La primera versión de Java fue liberada por Sun Microsystems en 1995, para finales de los 90's Java se había convertido en unos de los lenguajes de programación más usados gracias a su portabilidad y a la filosofía con que fue creado. Gosling decidió usar la sintaxis de C y C++ que ya muchos programadores conocían aunque Java tiene poco acceso a niveles inferiores de comunicación con los que cuentan estos dos.

Por sus características Java se ha convertido en el lenguaje más popular para desarrollar aplicaciones web de tipo cliente-servidor.

Fueron cinco los principales objetivos en la creación de Java.

- Debía ser simple, orientado a objetos y familiar.
- Debía ser robusto y seguro.
- No debía depender de la arquitectura y debía ser portable.
- Debía ejecutarse con alto desempeño.
- Debía ser interpretado, multiprocesos y dinámico.

Por las características descritas anteriormente y los requerimientos de la aplicación es que se decide optar por el uso de Java como lenguaje de programación. Se utiliza también los Enterprise Java Beans (EJB), Java Servlet Pages (JSP's), Hibernate y el API de Apache Struts 1.3 para poder seguir la arquitectura MVC.

Las secciones siguientes muestran las capturas de pantalla de los diseños interfaz de usuario para cada uno de los módulos desarrollados.

4.3 Módulo de inscripción de académicos.

A continuación se presentan las figuras de las interfaces de usuario para el módulo de

inscripción para académicos. La Figura 4.6 muestra la pantalla inicial del módulo de inscripción donde se solicitan los datos del académico para verificar su identidad.

Figura 4.6. Formulario principal.

Una vez ingresados los datos, se verifican con la DGP y en caso de ser correctos se muestra la información del académico, que proporciona la DGP, y se pregunta si son correctos; como se muestra en la Figura 4.7.

Figura 4.7. Verificación de datos.

Si los datos son correctos se le presenta una liga con las “Políticas de uso”. Para registrarse en el servicio de Acceso Remoto es necesario aceptarlas, ver Figura 4.8.

Figura 4.8. Aceptación de condiciones de uso.

Se aceptan las políticas de uso y el sistema avisara que se ha llevado con éxito el registro al servicio de Acceso Remoto como se muestra en la Figura 4.9.

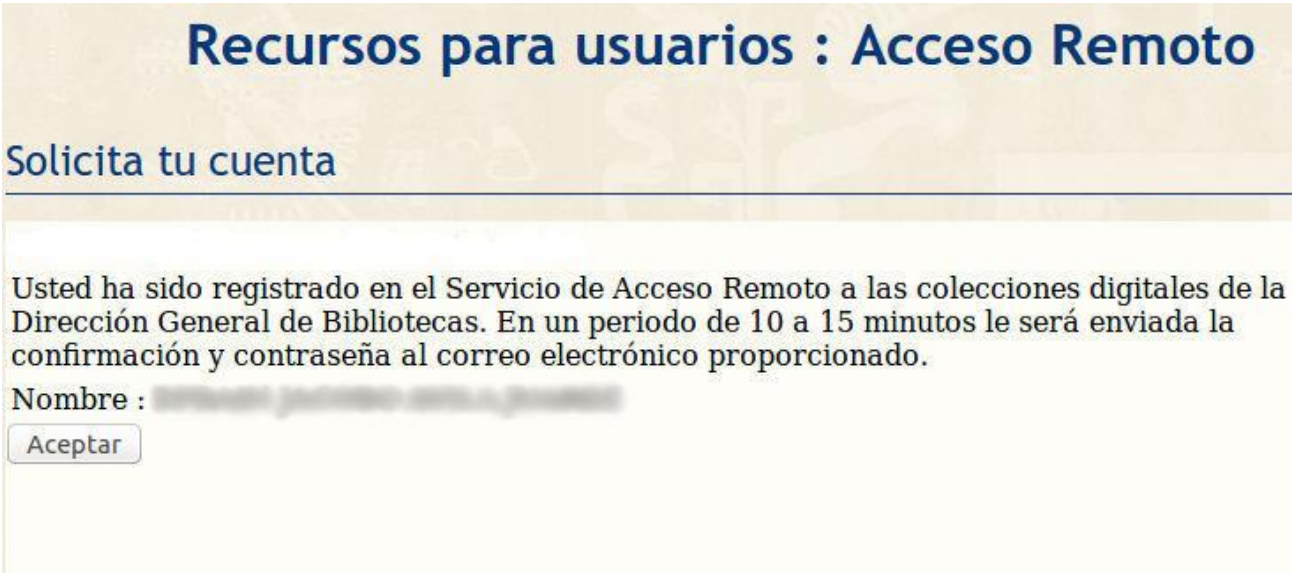

Figura 4.9. Aviso de registro exitoso al servicio de Acceso Remoto.

4.4 Módulo de inscripción de alumnos.

A continuación se presentan las figuras de las interfaces de usuario para el módulo de inscripción para alumnos. La Figura 4.10 muestra la pantalla inicial del módulo de inscripción donde se solicitan los datos del académico para verificar su identidad.

Figura 4.10. Formulario principal.

Una vez ingresados los datos, se verifican con la DGAE y en caso de ser correctos se muestra la información del académico, que proporciona la DGAE, y se pregunta si son correctos; como se muestra en la Figura 4.11.

Figura 4.11. Verificación de datos y correo electrónico.

Si los datos son correctos se le presenta una liga con las “Políticas de uso”. Para registrarse en el servicio de Acceso Remoto es necesario aceptarlas, ver Figura 4.12.

Figura 4.12. Aceptación de políticas de uso.

Se aceptan las políticas de uso y el sistema avisara que se ha llevado con éxito el registro al servicio de Acceso Remoto como se muestra en la Figura 4.13.

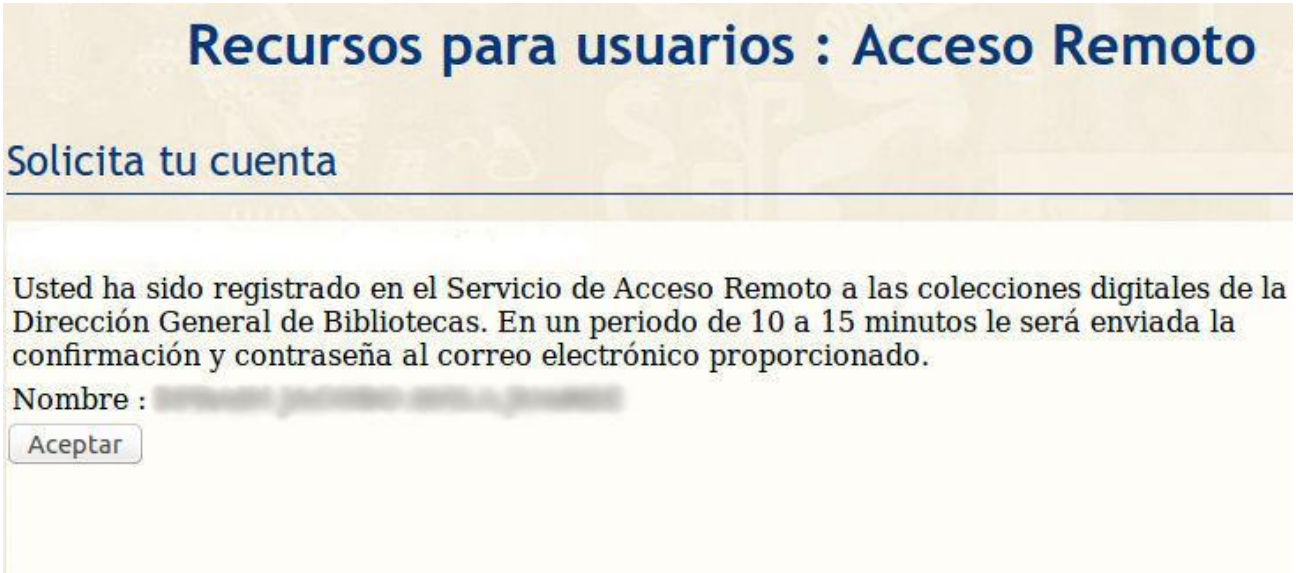

Figura 4.13. Aviso de registro exitoso al servicio de Acceso Remoto.

4.5 Módulo de recuperación de contraseña.

En las siguientes figuras se dan ejemplos de la interfaz de usuario del módulo de recuperación de contraseña. En la Figura 4.14 se muestra la pantalla

Figura 4.14. Ingresar cuenta de usuario.

Una vez que se ha ingresado el número de cuenta de alumno o número de trabajador de la UNAM y si el usuario ya está registrado en la base de datos de Acceso Remoto, el sistema responderá con el mensaje de que la contraseña será enviada vía correo electrónico en un tiempo de 10 a 15 min, como se muestra en la Figura 4.15. Además mostrara el nombre completo y número de cuenta o número de trabajador.

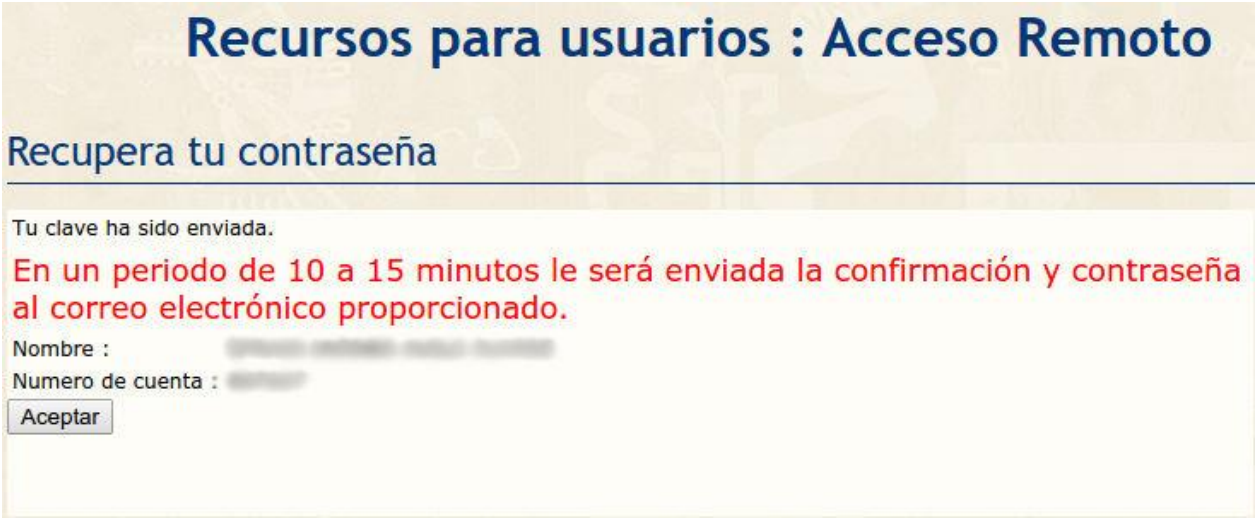

Figura 4.15. Respuesta del sistema.

4.6 Módulo de cambio de contraseña.

El módulo de cambio de contraseña permite al usuario cambiar la contraseña de Acceso Remoto, según el patrón descrito en el caso de uso "Cambiar contraseña" del capítulo 3. La pantalla inicial se muestra en la figura 4.16, se debe ingresar con el número de cuenta de alumno o número de trabajador y la contraseña actual.

Figura 4.16. Entrada al sistema de actualización de contraseña.

Una vez que se ha validado correctamente el sistema mostrara la información del usuario como: nombre completo, número de cuenta o trabajador, cargo o nivel, correo electrónico, dependencia y opcionalmente fecha de nacimiento. Además se mostrara un botón con el título "Cambiar contraseña", ver Figura 4.17.

Figura 4.17. Vista de la información del usuario.

Al dar clic en el botón el sistema pedirá al usuario ingresar una nueva contraseña con el patrón descrito, como se muestra en la Figura 4.18. Deberá ingresar 2 veces la contraseña por cuestiones de verificación.

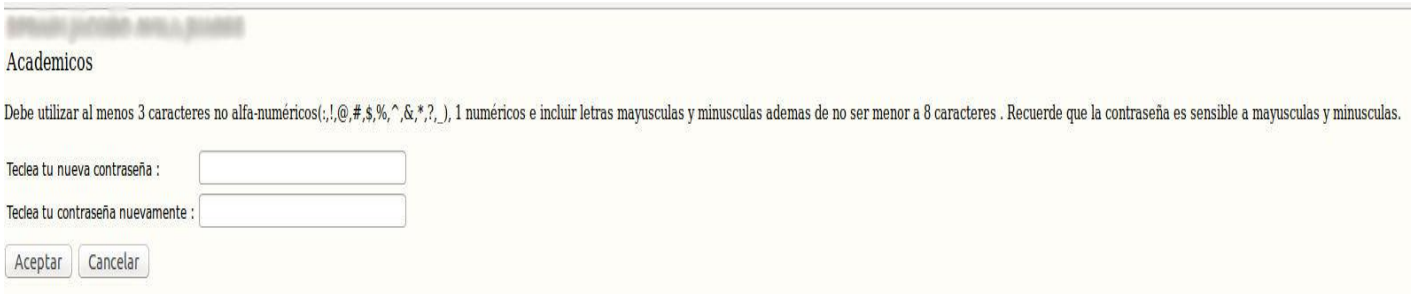

Figura 4.18. Cambio de contraseña.

Si se ha ingresado la contraseña con el patrón requerido y la verificación es correcta el sistema mostrará la Figura 4.19 con la respuesta de “Nueva Contraseña guardada”.

Figura 4.19. Respuesta al cambio de contraseña.

4.7 Módulo de renovación de vigencia.

El módulo de renovación de vigencia no tiene una interfaz de usuario, por su naturaleza de solo entregar resultados se plantea como una aplicación de línea de comandos. A continuación se muestra un ejemplo de código:

```
/**
 *
 * @author Efraín Jacobo Ávila Juárez <efra2012@gmail.com>
 */
public class MonitorCheck implements Schedulable {

   private Usuarios remoteSession;
   private Alumnos alumnosSession;
   private Emp empleadosSession;
   private Administradores admSession;
   private Context ctx;
   private int cuentaCiclos;
   private List<Usuario> allUsers;
   private List<Alumno> alumnoUsers;
   private List<Empleado> empleadoUsers;
   private ListIterator emplIter;
   private List<Cargo> listCargos;
   private Vector<String> ctaList;
   private Properties alumnoInf;
   private mx.unam.dgpe.bd.servicios.EmpleadoBean empleadoInf;
   private boolean found;
   private Date noFN;
   private long beginProc;
   private long endProc;
   private long beginProc1;
   private long endProc1;
   private long beginTime;
   private long endTime;
   private int empI;
   private int alumI;
   private int alumVigente;
   private int alumNoVigente;
   private int alumOtro;
```

```
        private int emplVigente;
        private int emplNoVigente;
        private int emplOtro;
        private int cntdr,  repetirCnt;
        private String cta;
        private String logPath;
        private File log;
        private FileWriter logWriter;
        private BufferedReader bfReader;
        private int opcn;
        private static String AVISO = "AVISO : OPCION NO VALIDA FAVOR DE INTRODUCIR OPCION VALIDA Y REINICIAR SERVICIO";
        private static String DGAEWSFALLA = "El DGAE-WS no responde más a las peticiones. Volcando archivos y cerrando procesos...";

        public MonitorCheck(String arg) {
           try {
              configFile cfg = new configFile();
              logPath = cfg.File().getProperty("logPath");
              cuentaCiclos = 0;
              if (arg == null || arg.equals("TODOS")) {
                 opcn = 1;
                 log = new File(logPath + "check.log");
              } else if (arg.equals("ALUMNOS")) {
                 opcn = 2;
                 log = new File(logPath + "checkAlumnos.log");
              } else if (arg.equals("EMPLEADOS")) {
                 opcn = 3;
                 log = new File(logPath + "checkEmpleados.log");
              } else {
                 opcn = 0;
                 log = new File(logPath + "checkFail.log");
              }

              System.out.println("Users check system ON.....");
           } catch (Exception e) {
              e.printStackTrace();
           }
        }

        public MonitorCheck(String op, String arg) {
           configFile cfg = new configFile();
           logPath = cfg.File().getProperty("logPath");
           ctaList = new Vector<String>();
           String line;
           cuentaCiclos = 0;
           //System.out.println("op "+op+" arg "+arg);
           if (op == null) {
              opcn = 0;
              log = new File(logPath + "checkFail.log");
           } else if (op.equals("ALTAS")) {
              try {
                 bfReader = new BufferedReader(new FileReader(arg));
                 while ((line = bfReader.readLine()) != null) {
                    ctaList.add(line);
                 }
              } catch (FileNotFoundException fnfe) {
                 System.out.println("ARCHIVO NO ENCONTRADO FAVOR DE VERIFICAR");
                 opcn = -4;
              } catch (java.io.IOException ioe) {
                 ioe.printStackTrace();
              }
```

```
            opcn = 4;
            log = new File(logPath + "altas.log");
        } else if (op.equals("ACTIVACIONES")) {
            try {
                bfReader = new BufferedReader(new FileReader(arg));
                while ((line = bfReader.readLine()) != null) {
                    ctaList.add(line);
                }
            } catch (FileNotFoundException fnfe) {
                System.out.println("ARCHIVO NO ENCONTRADO FAVOR DE VERIFICAR");
                opcn = -4;
            } catch (java.io.IOException ioe) {
                ioe.printStackTrace();
            }
            opcn = 6;
            log = new File(logPath + "activaciones.log");
        } else if (op.equals("VERIFICACION")) {
            opcn = 5;
            log = new File(logPath + "checkVerificacion.log");
        } else if(op.equals("ALUMNOS")){
                    opcn = 2;
                    cntdr = Integer.valueOf(arg);
                    //System.out.println(" BP "+cntdr);
                    log = new File(logPath + "checkAlumnos.log");
        }else {
            opcn = 0;
            log = new File(logPath + "checkFail.log");
        }
        System.out.println("Users check system ON.....");
        System.out.println("log file : " + log.getName() + " path " + log.getPath());
    }

    private Administrador getDefaultAdmin() {
        Administrador admin = new Administrador();
        admin.setCuentaAdministrador("proxy-check");
        admin.setEmail("ar-bidi@dgb.unam.mx");
        admin.setNombre("Acceso Remoto BiDi-UNAM");
        admin.setApellidoPaterno("");
        admin.setApellidoMaterno("");
        return admin;
    }

    private void init() throws Exception {
        ctx = new InitialContext();
        remoteSession = (Usuarios) ctx.lookup("UsuariosBean/remote");
        alumnosSession = (Alumnos) ctx.lookup("AlumnosBean/remote");
        empleadosSession = (Emp) ctx.lookup("EmpBean/remote");
        admSession = (Administradores) ctx.lookup("AdministradoresBean/remote");

        log.createNewFile();
        logWriter = new FileWriter(log);
        logWriter.write(new Date(System.currentTimeMillis()).toString() + "\n");
        endProc = beginProc = 0;
        alumVigente = alumNoVigente = alumOtro = emplVigente = emplNoVigente = emplOtro = 0;
        cuentaCiclos++;
    }

    public void perform(Date date, long l) {
        try {
            beginTime = System.currentTimeMillis();
            System.out.println("Users check system initializing.....");
            init();
```

```
      listCargos = remoteSession.ListaCargos();
      Date now = new Date();
      Date end = new Date();

      if (opcn != 0) {
         if (opcn == 1) {
            checkAlumnos(0);
            checkEmpleados(0);
         } else if (opcn == 2) {

            checkAlumnos(cntdr);

         } else if (opcn == 3) {

            checkEmpleados(0);

         } else if (opcn == 4) {

            repetirCnt = 0;
            insertUsuario(0);

         } else if (opcn == 6) {

            activaUsuarios(0);
         }
         endTime = System.currentTimeMillis();
         System.out.println("Check proccess total time : " + (endTime - beginTime) / 1000 + " sec.");
         System.out.println("Ending check system process.....");
         logWriter.write("Check proccess total time : " + (endTime - beginTime) / 1000 + " sec." + "\n");
         logWriter.flush();
      } else {
         System.out.println(AVISO);
         logWriter.write(AVISO);
         logWriter.flush();
      }

   } catch (Exception e) {
      //logWriter.write(e.toString());
      e.printStackTrace();
   }
}

public void checkEmpleados(int index) {
   try {
      empleadoUsers = empleadosSession.dameEmpleados();
      if (index == 0) {
         emplIter = empleadoUsers.listIterator();
         beginProc1 = System.currentTimeMillis();
         empI = 0;
      } else {
         emplIter = empleadoUsers.listIterator(index);
         empI = index;
      }

      System.out.println("numero de empleados " + empleadoUsers.size());
      logWriter.write("numero de empleados " + empleadoUsers.size() + "\n");
      Empleado emp = new Empleado();
      Usuario user = new Usuario();
      Cargo job = new Cargo();
      Date end = new Date();
```

```
                mx.unam.dgpe.bd.servicios.EmpleadoService dgpeSystem = new mx.unam.dgpe.bd.servicios.EmpleadoService();
                mx.unam.dgpe.bd.servicios.Empleado port;// = dgpeSystem.getEmpleadoPort();
                String depId;
                System.out.println("index at : " + index);
                logWriter.write("index at : " + index);

                for(empI=0;empI<21;empI++){
                   found = false;
                   port = dgpeSystem.getEmpleadoPort();
                   emp = (Empleado) emplIter.next();
                   if (emp.getDependencia().endsWith("E")) {
                      depId = emp.getDependencia();
                      depId = depId.substring(1, depId.length() - 1);
                   } else {
                      depId = emp.getDependencia();
                   }
                   ListIterator cargoIter = listCargos.listIterator();
                   System.out.print(empI + " - empleado " + emp.getNumEmp());
                   logWriter.write(empI + " - empleado " + emp.getNumEmp() + "\n");
                   user = remoteSession.dameUsuario(emp.getNumEmp());
                   if (user != null) {
                      empleadoInf = port.vigente(emp.getRFC());
                      System.out.println("status " + empleadoInf.getStatus());
                      logWriter.write("status " + empleadoInf.getStatus() + "\n");
                      if (empleadoInf.getStatus().startsWith("Vigente") && (empleadoInf.getTipoEmpleado().equals("Emeritos") ||
empleadoInf.getTipoEmpleado().equals("Asignatura") || empleadoInf.getTipoEmpleado().equals("Académicos"))) {

                         job = admSession.dameCargo(empleadoInf.getTipoEmpleado());
                         System.out.println("job : " + job.getNombre());
                         if (emp.getTipo() == null || !emp.getTipo().equals(empleadoInf.getTipoEmpleado())) {
                            emp.setTipo(String.valueOf(job.getId()));//empleadoInf.getTipoEmpleado());
                         }
                         if (depId == null || !depId.equals(empleadoInf.getCveDep()) || !emp.getDependencia().endsWith("E")) {
                            emp.setDependencia(empleadoInf.getCveDep() + "E");
                         }
                         if (emp.getCategoria() == null || !emp.getCategoria().equals(empleadoInf.getCategoria())) {
                            emp.setCategoria(empleadoInf.getCategoria());
                         }
                         if (user.getCargo() == null || !user.getCargo().equals(empleadoInf.getTipoEmpleado())) {
                            user.setCargo(String.valueOf(job.getId()));
                         }
                         if (depId == null || !depId.equals(empleadoInf.getCveDep()) || !user.getDependencia().endsWith("E")) {
                            user.setDependencia(empleadoInf.getCveDep() + "E");
                         }
                         if (user.getFechaNacimiento() == null) {
                            user.setFechaNacimiento(java.sql.Date.valueOf("0000-00-00"));
                         }
                         end.setTime(job.getFechaFin().getTime());
                         empleadosSession.insertEmpleado(emp);
                         remoteSession.actualizaVigencia(user, end);
                         emplVigente++;
                      }
                   }
                   empI++;
                }
                endProc1 = System.currentTimeMillis();
                System.out.println("Empleados' check proccess time : " + (endProc1 - beginProc1) / 1000 + " sec.");
                logWriter.write("Empleados users updated " + emplVigente + "\n");
                logWriter.write("Empleados users not updated " + (empleadoUsers.size() - emplVigente) + "\n");
                logWriter.write("Empleados' check proccess time : " + (endProc1 - beginProc1) / 1000 + " sec." + "\n");

             } catch (javax.xml.ws.soap.SOAPFaultException sfe) {
```

```
        checkEmpleados(empI);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

protected void checkAlumnos(int pntr) {
    try {
        beginProc = System.currentTimeMillis();
        alumnoUsers = alumnosSession.dameAlumnos();
        System.out.println("numero de alumnos " + alumnoUsers.size());
        logWriter.write("numero de alumnos " + alumnoUsers.size() + "\n");
        Alumno alum = new Alumno();
        Usuario user = new Usuario();
        Cargo job = new Cargo();
        Date end = new Date();
        wsDgae dgaeSystem = new wsDgae();
        ListIterator alumIter;
        if(pntr == 0) {
          alumIter = alumnoUsers.listIterator();
        } else {
                alumIter = alumnoUsers.listIterator(pntr);
        }
        alumI = 0;
        while(alumIter.hasNext()){
            found = false;
            ListIterator cargoIter = listCargos.listIterator();
            alum = (Alumno) alumIter.next();
            user = remoteSession.dameUsuario(alum.getCuentaUsuario());
            System.out.print(alumI + " - cuenta " + alum.getCuentaUsuario() + " ");
            logWriter.write(alumI + " - cuenta " + alum.getCuentaUsuario() + "\n");
            if (user != null) {
                try{
               alumnoInf = dgaeSystem.getProperties(alum.getCuentaUsuario());
                } catch(java.lang.NullPointerException npe) {
                        npe.printStackTrace();
                        dgaeSystem = new wsDgae();
                        alumnoInf = dgaeSystem.getProperties(alum.getCuentaUsuario());
                }
               System.out.println("Mensaje: " + alumnoInf.getProperty("mensaje"));
               logWriter.write("Mensaje: " + alumnoInf.getProperty("mensaje") + "\n");
               if (alumnoInf.getProperty("mensaje").equals("El Alumno tiene vigencia")) {
                  alumVigente++;
                  String grd = "";
                  if (alumnoInf.getProperty("grado").equals("L")) {
                     grd = "Estudiante de Licenciatura";
                  } else if (alumnoInf.getProperty("grado").equals("M")) {
                     grd = "Estudiante de Maestria";
                  } else if (alumnoInf.getProperty("grado").equals("D")) {
                     grd = "Estudiante de Doctorado";
                  } else if (alumnoInf.getProperty("grado").equals("B")) {
                     grd = "Estudiante de Bachillerato";
                  } else if (alumnoInf.getProperty("grado").equals("I")) {
                     grd = "Estudiante de Iniciacion universitaria";
                  } else {
                     grd = "Estudiante de Especialidad";
                  }
                  System.out.println("BPgrado...." + grd+" "+alumnoInf.getProperty("grado"));

                  job = admSession.dameCargo(grd);
                  System.out.println("job : " + job.getNombre());
                  if (alum.getGrado() == null || !alum.getGrado().equals(alumnoInf.getProperty("grado"))) {
```

```
                        alum.setGrado(String.valueOf(job.getId()));
                    }
                    if (alum.getDependencia() == null || !alum.getDependencia().equals(alumnoInf.getProperty("clave"))) {
                        alum.setDependencia(alumnoInf.getProperty("clave"));
                    }
                    if (alum.getVigencia() == null || !alum.getVigencia().equals(alumnoInf.getProperty("vigencia"))) {
                        alum.setVigencia(alumnoInf.getProperty("vigencia"));
                    }
                    if (user.getCargo() == null || !user.getCargo().equals(grd)) {
                        user.setCargo(String.valueOf(job.getId()));
                    }
                    if (user.getDependencia() == null || !user.getDependencia().equals(alumnoInf.getProperty("clave"))) {
                        user.setDependencia(alumnoInf.getProperty("clave") + "A");
                    }
                    if (user.getFechaNacimiento() == null) {
                        user.setFechaNacimiento(java.sql.Date.valueOf("0002-11-30"));
                    }
                            //alum.setEmail(alumnoInf.getProperty("correo"));
                    end.setTime(job.getFechaFin().getTime());
                    alumnosSession.actualizaAlumno(alum);
                    remoteSession.actualizaVigencia(user, end);
                  } else {
                    alumNoVigente++;
                  }
                }
                alumI++;
              }
              endProc = System.currentTimeMillis();
              logWriter.write("Alumnos users not updated " + (alumnoUsers.size() - alumVigente) + "\n");
              logWriter.write("Alumnos users updated " + alumVigente + "\n");
              logWriter.write("Alumnos check process time : " + (endProc - beginProc) / 1000 + " sec." + "\n");

            } catch (Exception e) {
              e.printStackTrace();
            //logWriter.write(e.printStackTrace());
            }
          }
          private void insertUsuario(int indice) {
            try {
              Date hoy = new Date();
              Date end = new Date();
              String usr;
              String psswd;
              ListIterator<String> ctaIter = ctaList.listIterator(indice);
              Alumno alumno = new Alumno();
              Empleado empleado;
              Usuario usuario = new Usuario();
              Cargo job = new Cargo();
              wsDgae dgaeSystem = new wsDgae();
              mx.unam.dgpe.bd.servicios.EmpleadoService dgpeSystem = new mx.unam.dgpe.bd.servicios.EmpleadoService();
              mx.unam.dgpe.bd.servicios.Empleado port = dgpeSystem.getEmpleadoPort();

              for (cntdr = indice; cntdr < ctaList.size(); cntdr++) {
                usr = (String) ctaIter.next();
                System.out.println(cntdr + " usuario : " + usr);
                if ((alumnoInf = dgaeSystem.getProperties(usr)).getProperty("vigencia") != null &&
Integer.valueOf(alumnoInf.getProperty("vigencia")) > 0) {
                  String grd = "";
                  if (alumnoInf.getProperty("grado").equals("L")) {
                    grd = "Licenciatura";
                  } else if (alumnoInf.getProperty("grado").equals("M")) {
                    grd = "Maestria";
```

```
            } else if (alumnoInf.getProperty("grado").equals("D")) {
               grd = "Doctorado";
            } else if (alumnoInf.getProperty("grado").equals("B")) {
               grd = "Bachillerato";
            } else if (alumnoInf.getProperty("grado").equals("I")) {
               grd = "Iniciacion universitaria";
            } else {
               grd = "Especialidad";
            }
            System.out.println("dependencia ...." + alumnoInf.getProperty("clave"));
            job = admSession.dameCargo(grd);

            if ((usuario = remoteSession.dameUsuario(usr)) == null) {
               psswd = remoteSession.buildPassword(4, 3, 1);
               usuario.setCuentaUsuario(usr);
               usuario.setPalabraSecreta(psswd);
               usuario.setNombre(alumnoInf.getProperty("nombres"));
               usuario.setApellidoPaterno(alumnoInf.getProperty("apellido1"));
               usuario.setApellidoMaterno(alumnoInf.getProperty("apellido2"));
               usuario.setCargo(grd);
               usuario.setEmail(alumno.getEmail());
               usuario.setDependencia(alumnoInf.getProperty("clave") + "A");
               usuario.setFechaNacimiento(java.sql.Date.valueOf("0000-00-00"));
               usuario.setFechaCreacion(hoy);
            }

            usuario.setFechaModificacion(hoy);
            end.setTime(job.getFechaFin().getTime());
            Administrador adm = getDefaultAdmin();
            adm.setDependencia(alumnoInf.getProperty("clave") + "A");
            remoteSession.actualizaUsuarioConBitacora(usuario, adm, hoy, end);

            if ((alumno = alumnosSession.dameAlumno(usr)) == null) {
               alumno = new Alumno();
               alumno.setCuentaUsuario(usr);
               alumno.setNombre(alumnoInf.getProperty("nombres"));
               alumno.setApellidoP(alumnoInf.getProperty("apellido1"));
               alumno.setApellidoM(alumnoInf.getProperty("apellido2"));
               alumno.setSexo(alumnoInf.getProperty("sexo"));
               alumno.setGrado(grd);
               alumno.setDependencia(alumnoInf.getProperty("clave") + "A");
               alumno.setVigencia(alumnoInf.getProperty("vigencia"));
               alumno.setEmail(alumnoInf.getProperty("correo"));
            }

            usuario = remoteSession.dameUsuario(usr);
            alumno.setPalabraSecreta(usuario.getPalabraSecreta());
            alumnosSession.insertAlumno(alumno, hoy, end);

         }
      }
   } catch (NullPointerException npe) {
      repetirCnt++;
      try {
         if (repetirCnt < 3) {
            System.out.println("reseteando.....");
            insertUsuario(cntdr);
         } else {
            logWriter.write("ultima posicion indice : " + cntdr + "\n");
            logWriter.write(DGAEWSFALLA + "\n");
            System.out.println(DGAEWSFALLA);
            return;
```

```
                }
            } catch (java.io.IOException ioe) {
                ioe.printStackTrace();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void activaUsuarios(int indice) {
        try {
            Date hoy = new Date();
            Date end = new Date();
            String usr;
            String psswd;
            ListIterator<String> ctaIter;
            List btList = Collections.emptyList();
            ListIterator btIter;
            Alumno alumno = new Alumno();
            Empleado empleado;
            Usuario usuario = new Usuario();
            Cargo job = null;
            Bitacora registroBt = new Bitacora();
            Administrador adm = getDefaultAdmin();
            String cuenta;
            String usrdep = new String();
            String nomCargo;
            alumVigente = 0;
            alumOtro = 0;

            for (ctaIter = ctaList.listIterator(indice); ctaIter.hasNext();) {

                cuenta = (String) ctaIter.next();
                System.out.println(ctaIter.previousIndex() + " usuario " + cuenta);
                logWriter.write(ctaIter.previousIndex() + " usuario " + cuenta + "\n");
                if ((usuario = remoteSession.dameUsuario(cuenta)) != null || !(btList =
remoteSession.dameRegistroBitacoraD(cuenta)).isEmpty()) {
                    if (usuario != null) {
                        if ((job = admSession.dameCargo(usuario.getCargo())) == null) {
                            job = admSession.dameCargo((admSession.dameCargos())[0]);
                        }
                        System.out.println("Usuario en tabla.");
                        logWriter.write("Usuario en tabla.\n");
                        usrdep = depEquivalente( usuario.getDependencia() );
                        nomCargo = gradoEquivalente( job.getNombre() );
                    } else {
                        System.out.println("Usuario en bitacora");
                        logWriter.write("Usuario en bitacora\n");
                        usuario = new Usuario();
                        btIter = btList.listIterator();
                        while (job == null) {
                            registroBt = (Bitacora) btIter.next();
                            job = admSession.dameCargo(registroBt.getCargo());
                            if (!btIter.hasNext()) {
                                job = admSession.dameCargo((admSession.dameCargos())[0]);
                            }
                        }
                        System.out.println("nombre " + registroBt.getNombre() + " cuenta " + registroBt.getCuentaUsuario());
                        /*System.out.println(registroBt.getCuentaUsuario());
                        System.out.println(registroBt.getDependencia());
                        System.out.println(registroBt.getEmail());
                        System.out.println(registroBt.getCargo());*/
```

```
                usuario.setCuentaUsuario(registroBt.getCuentaUsuario());
                usuario.setPalabraSecreta(registroBt.getPalabraSecreta());
                usuario.setNombre(registroBt.getNombre());
                usuario.setApellidoPaterno(registroBt.getApellidoPaterno());
                usuario.setApellidoMaterno(registroBt.getApellidoMaterno());
                usuario.setCargo(registroBt.getCargo());
                usuario.setEmail(registroBt.getEmail());
                usuario.setFechaNacimiento(registroBt.getFechaNacimiento());
                usuario.setFechaCreacion(hoy);
                btList = null;
                usrdep = depEquivalente( registroBt.getDependencia() );
                nomCargo = gradoEquivalente( registroBt.getCargo() );
              }
              usuario.setFechaModificacion(hoy);
                    usuario.setCargo(nomCargo);
              end.setTime(job.getFechaFin().getTime());
              adm.setDependencia(usrdep);
              remoteSession.actualizaUsuarioConBitacora(usuario, adm, hoy, end);
              job = null;
              alumVigente++;

            } else {
              System.out.println("Usuario no encontrado en la BD.");
              logWriter.write("Usuario no encontrado en la BD.\n");
              alumOtro++;
            }
          }

          logWriter.write("Usuarios reactivados : " + alumVigente + "\n");
          logWriter.write("Usuarios no reactivados : " + alumOtro + "\n");
        } catch (Exception e) {
          e.printStackTrace();
        }
      }

      public String gradoEquivalente(String grado) {
        String grd = grado;
        if (grado.equals("L")) {
          grd = "Licenciatura";
        } else if (grado.equals("M")) {
          grd = "Maestria";
        } else if (grado.equals("D")) {
          grd = "Doctorado";
        } else if (grado.equals("B")) {
          grd = "Bachillerato";
        } else if (grado.equals("I")) {
          grd = "Iniciacion universitaria";
        } else if (grado.equals("E")) {
          grd = "Especialidad";
        }

        return grd;
      }
      public String depEquivalente(String dp){
      try{
        Integer.valueOf(dp);
        return dp+"A";
      } catch (NumberFormatException nfe){
        return dp;
      }
      }
    }
```

Capítulo 5
Implementación.

5.1 Sistema administrador de bases de datos MySQL.

SQL son las siglas en inglés del Lenguaje de Consulta Estructurado, diseñado para definir, administrar y consultar bases de datos relacionales. Las bases relacionales son utilizadas ampliamente hoy en día en muchos campos de aplicación. Descrito por Edgar F. Codd en la década de 1970 en su artículo "A Relational Model of Data for Large Shared Data Banks". Se considera un lenguaje declarativo aunque también incluye elementos de tipo procedural.

Mysql es uno de los manejadores de bases de datos más utilizado en el mundo, es de código abierto y software libre lo que en gran parte ha ayudado a su difusión. Creado por la compañía sueca Mysql AB en 1995 por Michael Widenius, David Axmark y Allan Larsson, que después fue adquirida en 2008 por Sun Microsystems y pertenece actualmente a Oracle después que esta adquirió a Sun Microsystems en 2010.

Entre los atributos que hacen de Mysql tan popular para su uso en aplicaciones web están:

- Soporta codificación Unicode.
- Soporta varios tipos de motores de registro (storage engine), como InnoDB.
- Soporta conexiones seguras tipo SSL.
- Soporta vistas actualizables.
- Soporta varios tipos de codificación de caracteres, muy importante cuando se trata de idiomas como el español.
- Indexación.
- Soporta disparadores y condicionales.

A continuación se presentan las tablas creadas a partir del diseño utilizando la herramienta Mysql Workbench:

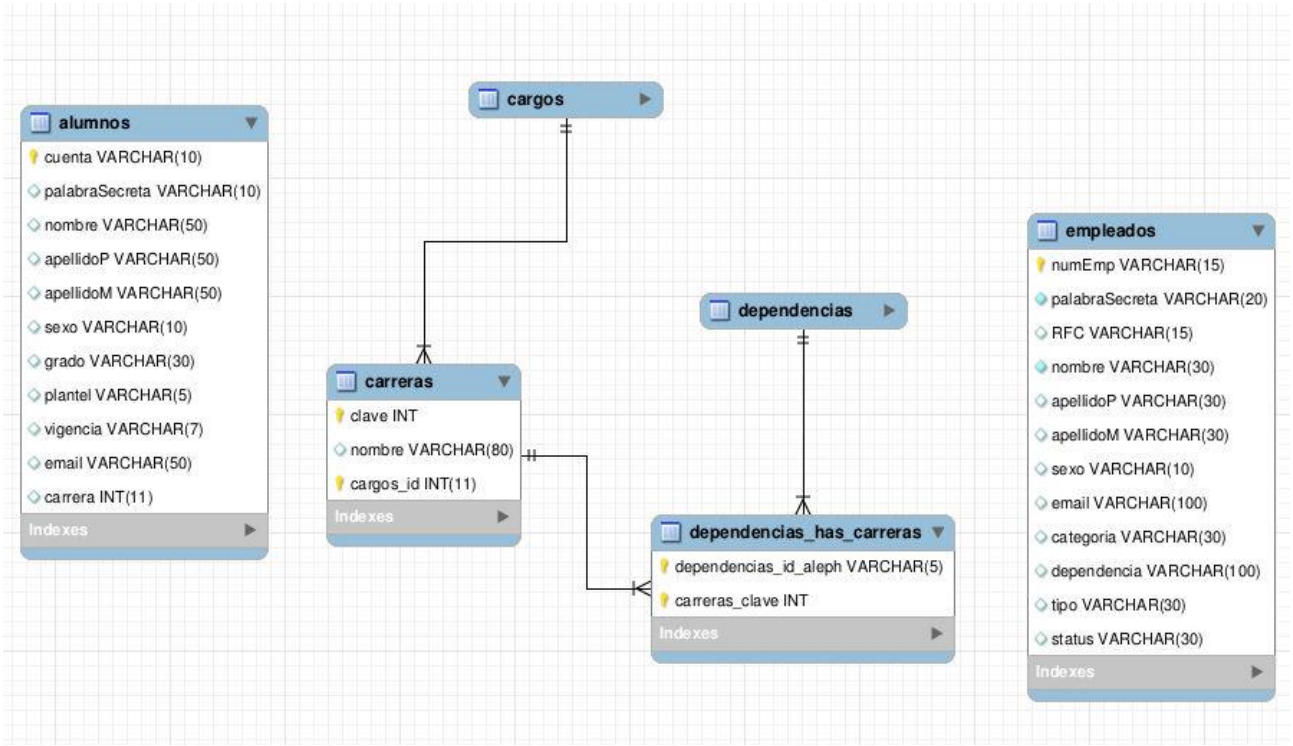

Figura 5.1. Tablas de alumnos, empleados y carreras.

Para poder almacenar la información que la DGAE proporciona de los alumnos se creó la tabla de alumnos y dado que es necesario asociar las carreras con las dependencias se crea la tabla de carreras que contiene la clave de carrera y el nombre de la carrera, además de la tabla dependencias_has_cargos que asocia a dependencias con carreras. Se crea la tabla de empleados para guardad la información que la DGP proporciona de los académicos.

La tabla de empleados registra la información de los académicos que la DGP envía. En la Figura 5.2 se muestra la integración con las bases mencionadas.

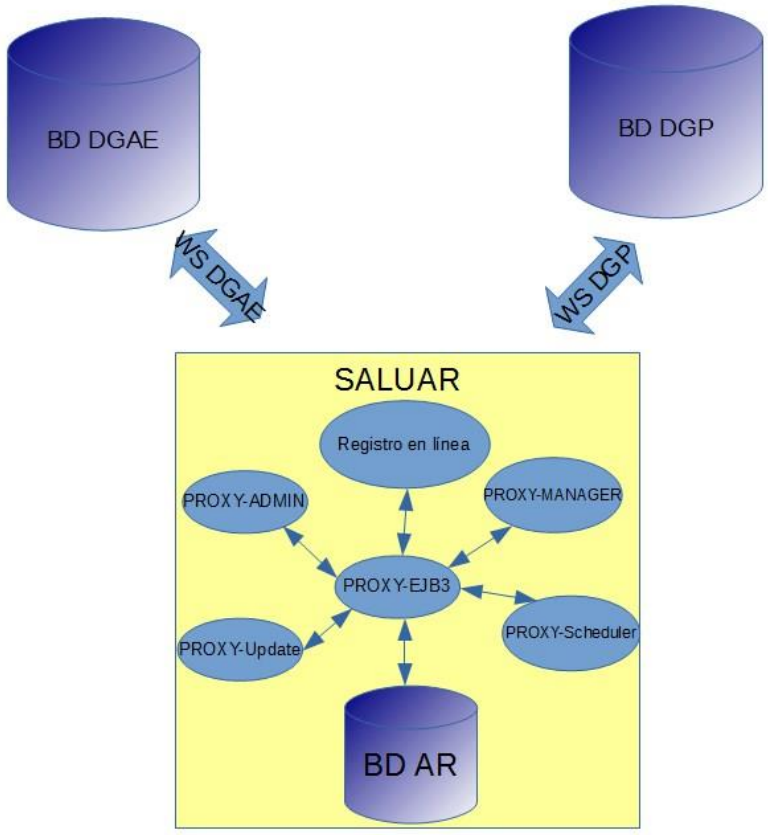

Figura 5.2. Diagrama de integración de las bases DGAE, DGP y de Acceso Remoto.

5.2 Servidor de aplicaciones Jboss.

El servidor de aplicaciones web Jboss (del inglés Jboss Aplicación Server) fue desarrollado en 1999 por la compañía Jboss creada por Marc Fleury. Inicialmente se llamó EJB-OSS (del inglés Entreprise Java Bean Open Source Software) que implementaba en API EJB desarrollado por Sun Microsystems. Cambió su nombre a Jboss cuando Sun Microsystems pidió al jefe del proyecto que desistiera de usar la marca registrada EJB en el nombre, así surgió Jboss. En la actualidad Jboss es desarrollado por Red Hat y se renombro a Jboss AS como WildFly.

Jboss está basado en la tecnología Java e implementa la plataforma Java Enterprise Edition. Es software libre y de código abierto, aunque posee una versión empresarial comercial. Algunas de sus características son:

- Puede ser implementado en varias plataformas Windows, Linux, BSD, etc.
- Integración con Hibernate.
- Soporta Java Transaction API(JTA)
- Soporta Java Naming and Directory Interface(JNDI)
- Certificado como soporte total del API Java EE 6 por parte de Oracle.
- Despliegue distribuido.
- Clustering
- Soporta JDBC
- Java Server Faces
- Java Server Pages
- Java Servlet

En el caso del servicio de Acceso Remoto y el SALUAR ya estaban implementados sobre el servidor de aplicaciones Jboss. Por sus características antes descritas se decidió continuar usándolo como servidor de aplicaciones.

5.3 Arquitectura Modelo-Vista-Controlador (MVC).

Un patrón de diseño describe un problema recurrente y su solución en un contexto definido, haciendo uso de la misma solución *n* número de veces sin hacerlo de la misma forma.

La división lógica que ofrece la arquitectura MVC del sistema permite que sea fácilmente modificable, con baja cohesión y alto acoplamiento, además permite reutilizar el código generado en diferentes tareas que el sistema deba realizar.

Para desarrollar el sistema de registro al acceso remoto en línea se utilizó el framework

Apache Struts versión 1.3, que implementa en su API la gran mayoría de la arquitectura MVC.

Para las vistas se utilizaron Java Servlet Pages(JSP), ya que están basadas en código HTML además de proporcionar las facilidades de utilizar clases Java. También se utilizaron hojas de estilo (CSS), el framework Jquery de Javascript y AJAX para las llamadas asíncronas al sistema.

El controlador lo provee el mismo API de Apache Struts a través de sus clases: *Action*, que es la clase que se encarga de recibir la petición del usuario a través de las vistas, seleccionar la acción que debe realizar el modelo y en base a los resultados que entrega seleccionar la respuesta que será enviada a las vistas en JSP.

El modelo lo proveen clases tipo EJB, a través del API de Java EE. Estas clases son de tipo POJO(Plain Old Java Object) y se utilizan para representar las tablas del modelo de datos, además de otras útiles que representan vistas o consultas compuestas al mismo modelo descrito en el capítulo anterior.

5.4 Pruebas.

Las pruebas al sistema permiten verificar si lo que se diseñó en los anteriores pasos cumple con el análisis de requisitos y va de lo más pequeño y avanza hacia lo grande, el sistema en su conjunto. Su objetivo es proporcionar información objetiva e independiente sobre la calidad del producto software.

Las pruebas son un subconjunto de prácticas que pertenecen al área de Verificación y Validación (VyV). La Verificación tiene por objetivo aseguran que se construya el software correctamente y la Validación que se construya el software correcto.

Existen varios tipos de pruebas según la literatura y la arquitectura de software que se elija, pero son 2 las más mencionadas: pruebas unitarias y pruebas de integración.

5.4.1 Pruebas unitarias.

Las pruebas unitarias se concentran en cada clase como unidad para verificar su correcta funcionalidad sin tomar en cuenta su relación con otras partes del sistema. Las tablas de la 5.1 a la 5.9 muestran las pruebas unitarias hechas al diseño.

Tabla 5.1 conexión a la base de datos.

Resultado esperado	Validación y conexión correcta a la base de datos con el fin de hacer consultas, insertar nuevos datos, borrar y actualizar datos almacenados.
Prerrequisitos	Nombre de la base de datos. Usuario de la base de datos. Contraseña. Puerto de comunicación. Driver JDBC para MySQL.
Datos de prueba	Datos de prueba de usuarios, tanto de alumnos como de académicos, para insertar, actualizar, borrar o consultar. Así como verificar datos de usuario: cuenta, contraseña y puerto con MySQL, cierre de conexión.
Resultados	Los datos de usuario de prueba mostraron integridad y se realizó conexión con MySQL y cierre de sesión.
Análisis de resultados	Los datos de prueba se manipularon de forma correcta en las funciones consultar, insertar, actualizar y borrar. El usuario y contraseña de MySQL se validar de forma correcta, así como el cierre correcto de la sesión.

Tabla 5.2 conexión al web service de DGAE.

Resultado esperado	Envío de validación de datos de alumno y generación de llave codificada, recepción de datos de forma correcta.
Prerrequisitos	Cuenta de alumno. Nivel académico del alumno. Clave de plantel del alumno. Clave de carrera del alumno. Llave codificada en SHA1.

Datos de prueba	Datos de alumnos tanto válidos para la DGAE como de verificación de no existencia. Usuario y contraseña proporcionadas por la DGAE para la generación de llave codificada.
Resultados	La llave se generó y envío junto con los datos de prueba, el web service DGAE respondió con los datos del alumno como nombre, apellido paterno, apellido materno, vigencia, semestre de inscripción, email, sexo, mensaje en el caso de datos válidos. En el caso de datos de verificaron de no existencia el web service DGAE respondió con un mensaje de "Verifique sus datos" o "Nùmero de cuenta no encontrado".
Análisis de resultados	La llave se generó de forma correcta, los datos y llave se enviaron de forma correcta al web service DGAE. La respuesta del web service DGAE fue recibida de forma correcta en ambos casos, cuando los datos son válidos para la DGAE y cuando los datos son verificación de no existencia. Se recibió las respuestas esperadas.

Tabla 5.3 conexión al web service de DGP.

Resultado esperado	Envío de validación de datos de académico y verificación de certificado, recepción de datos de forma correcta.
Prerrequisitos	RFC con homoclave del académico. Certificado.
Datos de prueba	Datos de académico tanto válidos para la DGP como de verificación de no existencia. Certificado proporcionado por la DGP.
Resultados	Los datos del académico y el certificado se enviaron como petición al web service DGP, a su vez el web service DGP respondió con los datos del académico: nombre completo, numero de empleado, tipo de académico, entidad laboral, clave de tipo de académico, estatus, mensaje, etc.; en el caso de que el RFC con homoclave es válido. En caso de datos de verificación de no existencia el web service DGP respondió sólo con el mensaje "RFC no válido" o "RFC

	no encontrado”
Análisis de resultados	Los datos y el certificado se enviaron de forma correcta y la respuesta del web service DGP se recibió también de forma correcta. Las respuesta del web service DGP fueron las esperadas en el caso de datos válidos y en el caso de datos de verificación de no existencia.

Tabla 5.4 insertar un alumno a la base de datos.

Resultado esperado	Insertar correctamente los datos de prueba en las tablas de alumnos.
Prerrequisitos	Cuenta Contraseña Nombre Apellido paterno Apellido materno Nivel académico Correo electrónico Clave de plantel Clave de carrera Vigencia. Sexo.
Datos de prueba	Datos de usuarios a insertar en la tabla de alumnos. Se usaron datos válidos y datos no validos como números de cuenta con letras o caracteres especiales.
Resultados	Los datos se insertaron en la tabla de alumnos en el caso de datos válidos. En el caso de datos que no correspondían al dominio, MySQL reportó un error de dominio.
Análisis de resultados	Los datos de prueba se insertaron correctamente en la tabla de alumnos en el caso de datos válidos. En caso de datos no validos el manejador reportó los errores como es

	de esperarse.

Tabla 5.5 insertar un académico a la base de datos.

Resultado esperado	Insertar correctamente los datos de prueba en las tablas de empleados.
Prerrequisitos	RFC Numero de empleado Contraseña Nombre Apellido paterno Apellido materno Correo electrónico Sexo Dependencia Tipo Categoría Estatus
Datos de prueba	Datos de usuarios a insertar en la tabla de empleados. Se usaron datos válidos y datos no validos como números de empleado con letras o caracteres especiales.
Resultados	Los datos se insertaron en la tabla de empleados en el caso de datos válidos. En el caso de datos que no correspondían al dominio, MySQL reportó un error de dominio.
Análisis de resultados	Los datos de prueba se insertaron correctamente en la tabla de empleados en el caso de datos válidos. En caso de datos no validos el manejador reportó los errores como es de esperarse.

Tabla 5.6 consultar información de un usuario de Acceso Remoto.

Resultado esperado	Consultar información de la base de datos, consultas tanto atómicas como de subconjuntos de los usuarios registrados. Resultados en tablas temporales.
Prerrequisitos	Uno o varias cuentas de usuario, que comúnmente son cuentas de alumnos o números de empleados.
Datos de prueba	Cuentas de usuarios de Acceso Remoto válidas y no válidas.
Resultados	MySQL regresa, en forma de tablas, las consultas en el caso de cuentas de usuarios válidas, en el caso de cuentas no válidas las tablas están vacías.
Análisis de resultados	Las tablas que regresa MySQL son correctas en el caso de cuentas de usuarios válidas y también cuando las cuentas no son válidas. Las respuestas en tablas son correctas.

Tabla 5.7 actualizar información de un usuario de Acceso Remoto.

Resultado esperado	Actualizar información de los usuarios en la base de datos de forma atómica.
Prerrequisitos	Cuenta Contraseña Nombre Apellido paterno Apellido materno email Nivel académico Dependencia Fecha de cumpleaños

	Fecha de registro Fecha de modificación
Datos de prueba	Datos válidos y no válidos para actualizar la tabla de usuarios.
Resultados	Los datos se actualizaron en la tabla de usuarios en el caso de datos válidos. En el caso de datos que no correspondían al dominio de las columnas, MySQL reportó un error de dominio.
Análisis de resultados	Los datos de prueba se insertaron correctamente en la tabla de usuarios en el caso de datos válidos. En caso de datos no válidos el manejador reportó los errores como es de esperarse.

Tabla 5.8 actualizar vigencia de un usuario de Acceso Remoto.

Resultado esperado	Actualizar información de fecha de fin de vigencia en la base de datos de forma atómica.
Prerrequisitos	Cuenta Fecha de fin de vigencia
Datos de prueba	Datos válidos y no válidos para actualizar la tabla de eliminados.
Resultados	Los datos se actualizaron en la tabla de eliminados en el caso de datos válidos. En el caso de datos que no correspondían al dominio de las columnas, MySQL reportó un error de dominio con excepción de los datos que no presentan ningún valor para la fecha de fin de vigencia. En este caso MySQL utiliza el formato “0002-11-30”.
Análisis de resultados	Los datos de prueba se insertaron correctamente en la tabla de eliminados en el caso de datos válidos. En caso de datos no válidos el manejador reportó los errores como es de esperarse. Para el caso de que la fecha no presenta ningún valor el sistema debe notificar.

Tabla 5.9 enviar correo electrónico.

Resultado esperado	Enviar exitosamente un correo electrónico a un usuario de Acceso Remoto.
Prerrequisitos	Correo electrónico. Cuenta Mensaje.
Datos de prueba	Correo electrónico. Válido y no válido.
Resultados	El correo electrónico se envía en el caso de que exista, de otra forma se reporta una excepción
Análisis de resultados	Los correos electrónicos se enviaron correctamente en caso de existencia, de otra forma se reporta la excepción como se espera.

5.4.2 Pruebas de integración.

Las pruebas de integración verifican, al unir todas las partes, el correcto funcionamiento del sistema haciendo énfasis en la colaboración entre unidades. Estas pruebas están basadas en los casos de uso analizados en el Capítulo 3. Las tablas de la 5.10 a la 5.14 muestran las pruebas y los resultados obtenidos.

Tabla 5.10 registrar un alumno al Acceso Remoto.

Resultado esperado	Registrar a un alumno como nuevo usuario de Acceso Remoto, utilizando la información que el web service DGAE proporciona. Enviar notificación y contraseña al

	interesado vía correo electrónico.
Prerrequisitos	Alumno regular inscrito en alguno de los niveles educativos que ofrece la UNAM. Conexión con el web service DGAE. Conexión con la base de datos.
Datos de prueba	Número de cuenta de alumno, nivel de estudios, clave del plantel y clave de carrera, llave codificada.
Resultados	Se valida la información enviada por la DGAE, si el alumno es vigente se registra su información en las tablas usuarios y alumnos. Además de registrar su fecha de fin de vigencia en la tabla de eliminados y registrar su notificación en la tabla de agregados.
Análisis de resultados	El registro del alumno se realiza de manera exitosa, con excepción de los casos en que se utiliza la letra 'ñ' con lo que se tuvo que codificar en UTF-8 la información que la DGAE envía.

Tabla 5.11 registrar un académico al Acceso Remoto.

Resultado esperado	Registrar a un académico como nuevo usuario de Acceso Remoto, utilizando la información que el web service DGP proporciona. Enviar notificación y contraseña al interesado vía correo electrónico.
Prerrequisitos	Académico (de tipo académicos, asignatura y eméritos) en nómina de la UNAM. Conexión con el web service DGP. Conexión con la base de datos.
Datos de prueba	RFC con homoclave del académico y llave codificada.
Resultados	Se valida la información enviada por la DGP, si el académico es empleado de la UNAM se registra su información en las tablas usuarios y empleados. Además de registrar su fecha de fin de vigencia en la tabla de

	eliminados y registrar su notificación en la tabla de agregados.
Análisis de resultados	El registro del académico se realiza de manera exitosa, con excepción de los casos en que se utiliza la letra 'ñ' con lo que se tuvo que codificar en UTF-8 la información que la DGP envía.

Tabla 5.12 cambiar la contraseña de un usuario de Acceso Remoto.

Resultado esperado	Actualizar la contraseña del usuario que lo solicite, según las características que debe presentar dicha contraseña.
Prerrequisitos	Estar registrado como usuario en Acceso Remoto. Número de cuenta o número de trabajador o cuenta asignada por administrador local. Contraseña Conexión a la base de datos.
Datos de prueba	Número de cuenta de Acceso Remoto y contraseña actual.
Resultados	Se valida al usuario por medio de su cuenta y contraseña. El usuario decide cambiar su contraseña siguiendo el patrón: al menos una letra mayúscula, al menos 2 caracteres numéricos, al menos 2 caracteres especiales(#,$,(,),*,/,+,-,&,@,?,¿,=), la contraseña no debe ser menor de 8 caracteres. Se registra la nueva contraseña.
Análisis de resultados	La contraseña se registra de manera exitosa. La validación de la contraseña también se realiza correctamente.

Tabla 5.13 recuperar contraseña de un usuario de Acceso Remoto.

Resultado esperado	El usuario solicita su contraseña y el sistema la envía vía correo electrónico.
Prerrequisitos	Estar registrado como usuario de Acceso Remoto.

	Número de cuenta o número de trabajador o cuenta asignada por administrador local. Conexión a la base de datos.
Datos de prueba	Número de cuenta de Acceso Remoto.
Resultados	El usuario de Acceso Remoto proporciona su número de cuenta de Acceso Remoto, si está vigente en el servicio de Acceso Remoto el sistema le responde que su contraseña será enviada a su correo electrónico en un periodo de 10 a 15 minutos. De lo contrario le notifica que no se encuentra su número de cuenta de Acceso Remoto.
Análisis de resultados	Se hace la recuperación de la información del usuario de Acceso Remoto de manera correcta, en caso de existir, y se envía al correo electrónico registrado. Si no existe el usuario en la base de datos se notifica correctamente.

Tabla 5.14 actualizar vigencia de los usuarios de Acceso Remoto.

Resultado esperado	El personal responsable del Acceso Remoto en BiDi actualiza las vigencias de los usuarios (alumnos o académicos) que aun pertenecen a la comunidad universitaria, el proceso se lleva a cabo 15 días antes de la fecha de fin de vigencia establecida por los mismos responsables.
Prerrequisitos	Lista de alumnos registrados por medio del sistema de registro en línea con el formato: cuenta, nivel, clave de plantel, clave de carrera. Lista de académicos registrados por medio del sistema de registro en línea que incluye el RFC con homoclave de los académicos. Conexión con la base de datos. Conexión al web service DGAE. Conexión al web service DGP.
Datos de prueba	Lista de usuarios aún vigentes en el Acceso Remoto, próximos a terminar su vigencia con el formato

	especificado según sean alumnos o académicos.
Resultados	La actualización de vigencia se comienza generando los listados e invocando primero el web service DGAE para recibir la información y el estatus actual de los alumnos, de esta forma se actualiza la fecha de fin de vigencia, en la tabla de eliminados, y se extiende el uso del servicio de Acceso Remoto. Después se invoca el web service DGP, se recibe la información y el estatus de los académicos registrados en Acceso Remoto y, según su estatus, se actualiza la fecha de fin de vigencia, también en la tabla de eliminados para extender el uso del servicio de Acceso Remoto.
Análisis de resultados	La actualización de la fecha de fin de vigencia se realiza de manera correcta para aquellos alumnos que aún son estudiantes con sus derechos vigentes en la UNAM, también se actualiza su plantel, carrera y semestre actual en la tabla de alumnos de forma correcta. Sin embargo dado que hay registrados cerca de 85,000 alumnos de la UNAM en el servicio de Acceso Remoto la consulta al web service DGAE es muy lenta ya que por cada alumno se debe enviar cuenta, nivel, clave de plantel y clave de carrera y después recibir los datos actualizados. La actualización ha llegado a tardar hasta 8 horas. La actualización de la fecha de fin de vigencia, para académicos que aun laboral en la UNAM, se realiza de manera correcta. También se actualiza su dependencia, categoría y tipo de empleado. Dado que solo hay aproximadamente 15,000 académicos registrados en el servicio de Acceso Remoto la consulta al web service DGP no tarda más de 2 horas.

5.5 Liberación.

A la fecha se han entregado dos versiones del sistema de registro en línea del Acceso Remoto, según han evolucionados los requisitos para validar a los alumnos ya que ellos representan hasta el 90% de los usuarios. En el desarrollo inicial sólo se requería que el alumno ingresara su cuenta de alumno UNAM, para la siguiente versión y por cuestiones de dar mayor seguridad y certeza al validar a los alumnos se requirió que el alumno proporcionara cuenta,

nivel, plantel y carrera. No así con los académicos, el web service DGP no ha cambiado mucho en su funcionamiento, al ser una población más estable e identificarse con su RFC con homoclave que es un dato con bastante seguridad y certeza.

La última versión del sistema es la 2.0 y ya incluye todos los módulos descritos. Se encuentra en producción en los servidores del área de informática del edificio anexo de la DGB.

5.5.1 Requisitos de instalación.

El sistema de registro en línea al Acceso Remoto fue escrito en lenguaje Java, siguiendo el modelo de aplicación web con arquitectura MVC, por lo que su instalación es posible tanto en un servidor con Microsoft Windows Server como en un servidor con sistema operativo tipo UNIX (Open Solaris, SUSE Linux, Debian, Fedora).

Actualmente el sistema de registro en línea al Acceso Remoto está instalado en un servidor con las siguientes características.

- Sistema operativo Fedora Linux versión 16.
- Servidor de base de datos MySQL a partir de la versión 5.1
- Servidor de aplicaciones Red Hat Jboss AS a partir de la versión 4.3 ó actual WildFly. También se pueden utilizar otros servidores de aplicaciones como GlassFish, pero no se probó.
- Java SE a partir de versión 1.6 por cuestiones de conectividad IPV6 con el web service DGAE y el web service DGP.
- Java EE a partir de la versión 1.6.
- Servidor marca DELL PowerEdge T410 con 24GB DDR3 de RAM, procesador Intel Xeon 5660, disco duro SAS de 750 GB.

Conclusiones.

El manejo de la información bibliotecaria y los diferentes procesos a los que se enfrentan los responsables de las bibliotecas es un campo poco explorado por la ingeniería del software, a pesar de tratarse de un área que maneja mucha y muy importante información.

La revolución que supuso el Internet en la década pasada llevó a que la información fuera más fácil de compartir con la gente interesada en ella. Las bibliotecas y los libros han tenido que adaptarse a esta tecnología que aunque no hará desaparecer a las primeras si supone un cambio en cuanto a que es una biblioteca y un libro. Las nuevas generaciones conocen el libro electrónico pero ya no sabrán lo que es una estantería de fichas bibliográficas.

En el año 2009 con el episodio de influenza AH1N1 se desarrolló como solución emergente el sistema de registro en línea al Acceso Remoto. Para ese entonces el servicio se comenzaba a ofrecer a los académicos de la UNAM, y se planeaba ofrecer el servicio a los alumnos de los diferentes niveles educativos desde iniciación universitaria hasta los doctorados.

Desde el año 2009, cuando fue puesto en funcionamiento, al día de hoy el sistema de registro en línea al Acceso Remoto ha ayudado a aproximadamente 160,000 alumnos y profesores a aprovechar los recursos electrónicos que la DGB ofrece. El servicio de Acceso Remoto cuenta con una población flotante de cerca de 100,000 usuarios por semestre. El sistema puede ser usado por cualquier miembro académico de la comunidad universitaria no importando donde se encuentre y siempre y cuando cuente con una conexión a Internet y sea alumno o académico vigente. Los miembros de la comunidad universitaria pueden solicitar en cualquier hora del día su contraseña de Acceso Remoto.

Actualmente esta modalidad de registro es la más utilizada por los alumnos y académicos que desean tener acceso a los recursos electrónicos por su rapidez y facilidad de uso. Del total de los usuarios registrados en Acceso Remoto el 80% de ellos obtuvieron su contraseña por esta modalidad.

Las herramientas elegidas para el desarrollo del sistema de registro en línea al Acceso Remoto permitieron llevar a buen término el proyecto. El proceso unificado (UP) permitió el desarrollo gradual del sistema con la participación de los interesados. La arquitectura MVC permitió

modularizar de forma correcta al sistema para que el caso de actualización sólo se modificara lo necesario y no se afectara todo el proyecto.

El software libre (Java, Apache Struts, Mysql, Jquery, Javascript) permitió que el costo en materia de software para el desarrollo del proyecto fuera mínimo. Además al ser un desarrollo propio de la UNAM éste puede ser ampliado, actualizado o modificado cuantas veces sea necesario según los nuevos requerimientos de la DGB sin incurrir en gastos adicionales en software de terceros.

Las ventajas del sistema de registro en línea al Acceso Remoto son:

- Alta disponibilidad, a cualquier hora todos los días del año.
- Rapidez para llevar a cabo el trámite y que el usuario obtenga su contraseña para acceder a los recursos digitales que la DGB ofrece.
- Capacidad para ofrecer al usuario la posibilidad de recuperar su contraseña en cualquier momento.
- El usuario solo se da de alta una vez, ya que la renovación de vigencia es llevada a cabo por la BiDi de forma automática.
- La arquitectura MVC y el diseño orientado a objetos permiten hacer actualizaciones o modificaciones rápidas.

Las desventajas del sistema de registro en línea al Acceso Remoto son:

- Aunque la contraseña es una buena idea para controlar el acceso, el usuario (alumno o académico) ya maneja varias para diferentes servicios que la UNAM ofrece.
- El framework Apache Struts 1.3 tiene poco soporte.
- La interfaz de usuario a veces no se visualiza bien en dispositivos móviles.
- La actualización automática toma mucho tiempo y puede interrumpir el funcionamiento de otros módulos del registro en línea;

En lo personal me ha dejado mucha satisfacción este proyecto por los alcances que ha tenido y la ayuda que ha prestado y sigue prestando a toda la comunidad universitaria en donde quiera que se encuentre ya sea en la Ciudad de México o en China. El ayudar a acercar la información a aquellos que lo requieren ya sea el estudiante en busca de respuestas a las tareas o proyectos o al investigador que hace ciencia básica o aplicada y que repercuta de forma positiva para el mundo.

APÉNDICES.

A.1 Instalación de JDK 1.7 para Linux.

El JDK tiene dos versiones una para arquitectura de 32 bits y otra para arquitectura de 64 bits. Para la instalación en servidor se recomienda la versión de 64 bits, si se tiene una máquina con más de 8 años de antigüedad se recomienda instalar el JDK de 32 bits.

Descargar de la página de Oracle http://www.oracle.com/technetwork/es/java/javase/downloads/index.html la versión que más convenga según el criterio antes descrito.

Este archivo es un comprimido con los archivos del kit de desarrollo. Para descomprimir se abre una consola de comandos y se ejecuta el siguiente comando:

tar zxvf jdk-8uversion-linux-x64.tar.gz

Este comando descomprimirá todo el contenido en una carpeta con nombre:

/jdk-8uversion-linux-x64

Desde aquí se pueden ejecutar la máquina virtual de java y el compilar javac.

Ahora se deberá configurar las variables de entorno para que el sistema sepa donde se encuentra el directorio de instalación del JDK. Desde la consola de comandos se ejecuta el comando:

vim .bashrc

Con este comando abrimos el archivo de configuración de variables de entorno. Ahora

debemos indicar en el archivo el directorio del JDK. Agregaremos al final del archivo las siguientes líneas:

export JAVA_HOME=/<path_de_directorio_jdk>/jdk<version>

export PATH=$PATH:${JAVA_HOME}/bin

Estas líneas le indican al sistema que hay una variable de entorno llamada "JAVA_HOME", y que debe estar disponible para su llamada; muchas aplicaciones dependientes de java buscarán este nombre de variable, se puede poner cualquier otro pero se recomienda usar éste. Se debe sustituir <path_de_directorio_jdk> por la ruta completa del directorio donde este el JDK.

Ahora se cierra la línea de comandos y se vuelve a abrir. Tecleamos los siguientes comandos:

java -version

javac -version

Debe aparecer en la línea de comandos los datos de la versión de JDK instalada:

java version "<version>"

Java(TM) SE Runtime Environment (build <version>)

Java HotSpot(TM) 64-Bit Server VM (build 24.45-b08, mixed mode)

y con esto ya se puede usar el API del JDK Standart Edition que es la base para cualquier aplicación escrita y ejecutada en java.

```
drwxrwxr-x 3 efrain efrain  4096 mar 31  2011 backup
drwxr-xr-x 2 efrain efrain  4096 ago  8  2013 bin
drwxr-xr-x 2 efrain efrain  4096 ago  8  2013 client
-rw-r--r-- 1 efrain efrain  6134 jul 18  2008 copyright.txt
drwxr-xr-x 7 efrain efrain  4096 jul 18  2008 docs
-rw-r--r-- 1 efrain efrain 60808 jul 18  2008 jar-versions.xml
-rw-r--r-- 1 efrain efrain  8178 jul 18  2008 JBossORG-EULA.txt
-rw-r--r-- 1 efrain efrain 33732 jul 18  2008 lgpl.html
drwxr-xr-x 3 efrain efrain  4096 ago  8  2013 lib
-rw-r--r-- 1 efrain efrain 62697 jul 18  2008 readme.html
drwxr-xr-x 5 efrain efrain  4096 jul 18  2008 server
```

A.2 Instalación de Jboss 4.2.3 para Linux.

Jboss es un software servidor de aplicaciones que ejecuta programas basados en lenguaje java y que son accedidos a través de Internet por un navegador como Mozilla Firefox, Google Chrome, MS Internet Explorer, etc. Provee a las aplicaciones los accesos a puertos de comunicación, acceso a base de datos a través de Hibernate e implementa el API de Apache Struts 1.3.

Para su instalación se debe primero descargar de la página de la comunidad de Red Hat Jboss www.jboss.org . Cabe mencionar que esta versión corresponde al año 2008 y que no es la última pero ha probado ser muy estable. Al día de hoy ha cambiado de nombre, se llama WildFly y está en su versión 8.2.

Una vez descargado hay que descomprimir el contenido del archivo con el siguiente comando:

tar zxvf jboss-4.2.3.GA.tar.gz

Al descomprimir el archivo se generará una nueva carpeta llamada *jboss-4.2.3.GA* y contendrá el siguiente árbol de subdirectorios:

El directorio “/bin” contiene los archivos ejecutables para iniciar y detener el servidor, además de la configuración de ejecución para la máquina virtual de java. El directorio “/lib” contiene las librerías que se necesitan para ejecutar Jboss y las que provee para las aplicaciones que

a su vez ejecuta y administra.

El directorio "/server" es el directorio de trabajo del servidor ahí se encuentran los archivos de configuración del servidor como puertos, ssl, bases de datos, aplicaciones a ejecutar y sus directorios de trabajo, administradores,etc. El directorio "/server" contiene una ruta a una subcarpeta donde se depositan las aplicaciones, la ruta "/server/default/deploy". Aquí se depositan y cuando el servidor arranca aquí es donde busca las aplicaciones y algunas de las configuraciones a ejecutar.

Para arrancar el Jboss se ejecuta el siguiente comando:

/bin/run.sh &

Con esto el servidor arrancará y comenzará a desplegar las aplicaciones, dará aviso por si encuentra algún fallo al desplegarlas, los servicios que se levantan como puertos, conexiones a bases de datos y otros servicios de terceros como el API de Quartz-Scheduler para ejecución de tareas periódicas.

Cuando se requiera detener el servidor se debe ejecutar el comando:

/bin/shutdown -S

Con esto Jboss dará de baja las aplicaciones y servicios.

GLOSARIO.

Acceso Remoto. Se le llama Acceso Remoto al Servicio de Acceso Remoto que es el mecanismo por el cual un usuario registrado puede tener acceso remoto a los recursos digitales que se encuentran en los catálogos de la DGB, esto es cuando el usuario no accede al portal de la DGB por medio de la RedUNAM sino por medio de una conexión privada a Internet como lo son los servicios de las compañías telefónicas o las compañías de comunicación con servicios tripleplay (telefonía, Internet y televisión de paga).

Aplicación web. Aplicación que reside en un servidor central, donde es accedida por muchos usuarios simultáneos desde cualquier parte del mundo a través de un navegador de Internet, su interfaz de usuario normalmente está desarrollada con lenguaje HTML.

Desarrollo web. Método de desarrollo de software para aplicaciones web. Generalmente se utilizan los patrones de diseño basados en capas como el MVC.

Lenguaje Unificado de Modelado (UML). Es un lenguaje de descripción de los diferentes pasos que puede dar el desarrollo del software, desde la descripción de los requisitos con casos de uso, hasta el flujo completo de la información a través del software, así como también las partes que conforman el software.

Metabuscador. Servicio de búsqueda que recupera información de diferentes fuentes o bases de datos, en el caso de la DGB permite recuperar información de sus diferentes bases de datos (TESIUNAM, LIBRUNAM, PERIODICA, SERIUNAM, etc.)

Modelo-Vista-Controlador (MVC). Arquitectura de 3 capaz que separa la interfaz gráfica de usuario del modelo y este del controlador. Ayuda a que el software construido con esta arquitectura sea fácilmente actualizado, mantenido y configurado sin grandes cambios. Algunas API's que siguen esta arquitectura son Apache Struts, Springs, JavaScriptMVC, etc.

Proceso Unificado (UP o RUP por sus siglas en inglés). El proceso unificado es un método de desarrollo de software centrado en los casos de uso, guiado por la arquitectura y es iterativo e incremental, con la participación activa de los usuarios finales.

Recursos digitales. Se define como recurso digital a todo aquel material documental

(revistas, tesis, libros, mapas, artículos científicos, imágenes, videos, bases de datos, etc.) de interés para la comunidad universitaria y que se encuentra disponible en algún formato electrónico y que debe ser accedido a través de una computadora o dispositivo móvil con conexión a Internet para su visualización.

Servicio web. Tipo de aplicación que contiene elementos de seguridad y autenticación automáticos, para consulta de información sensible. Escrita para que otro software la consulte, generalmente no tiene una interfaz gráfica de usuario y sus respuestas están cifradas y devueltas en archivos tipo XML.

Servidor de aplicaciones. Es un tipo especial de software que se encarga de administrar, controlar y desplegar las aplicaciones web, les da salida al Internet y provee los mecanismos por los cuales las aplicaciones web pueden acceder los recursos del servidor donde se alojen como bases de datos, sistema de archivos, puertos de comunicación, etc.

REFERENCIAS.

Fowler Martin, Kendall Scott. (1999). UML gota a gota. Mexico: Addison Wesley.

Jacobson, Ivar; Booch, Grady; Rumbaugh, James. (2004). El proceso unificado de desarrollo de software. México: Pearson Addison Wesley.

Larman, Craig, *UML y Patrones.* (2003).*Una introducción al análisis y diseño orientado a objetos y el Proceso Unificado.* México: Prentice Hall.

Maurizio Gabbrielli, Simone Martini. (2010).*Programming languages: principles and paradigms*, México: Springer.

Pratt, Terrence W. and Marvin V. Zelkowitz. (1996). *Programming Languages: Design and Implementation*, México: Prentice Hall.

Pressman, Roger S, (2005) *Ingeniería de Software. Un enfoque práctico,* México: Mc Graw Hill.

Ruíz Salinas, Silvestre; Villa García Luis Alberto. (2005). *Desarrollo de una aplicación web, que se emplee en el análisis numérico y gráfico del origen y evolución de rocas volcánicas.* México: UNAM.

Saynez Fabian, Julio Cesar; Benavidez Martinez, Alfredo. (2007) *Sistema de administración y control para el programa de tecnología en cómputo de la división de ingeniería eléctrica: portal PROTECO / tesis.* México: *UNAM*.

Schmuller, Joseph. (2001). *Aprendiendo UML en 24 hrs.* México: Ed Prentice Hall.

Silberschatz, Abraham; Korth, Henry F.; S. Sudarshan. (2002). *Fundamentos de bases de datos.* México: MCgraw Hill.

phD Cockburn, Alistair (2008). *Formato para escribir y dibujar casos de uso*. [ONLINE] Available at: http://alistair.cockburn.us/Alistair. [Last Accessed 2013].

Steve Masover (2004). *Model View Controller (MVC) Seminar*. [ONLINE] Available at: https://ist.berkeley.edu/as-ag/pub/pdf/mvc-seminar.pdf. [Last Accessed 2013]

Robert Eckstein (2007). *Java SE Application Design With MVC*. [ONLINE] Available at: http://www.oracle.com/technetwork/articles/javase/index-142890.html. [Last Accessed 2014].

Steve Burbeck Ph.D., (). *Applications Programming in Smalltalk-80(TM): How to use*

Model-View-Controller (MVC). [ONLINE] Available at: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html. [Last Accessed 2014]