UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Sistema de Información de alumnos, tutores y aspirantes del Programa de Posgrado de Ciencias Médicas, Odontológicas y de la Salud

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACÍON

P R E S E N T A :

RODRIGO TONATIHU CORTÉS MONROY

DIRECTOR DE TESIS:

M. en C MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ

2012

Agradecimientos

Esta tesis está dedicada a mis padres Evaristo y Socorro por siempre estar conmigo en las buenas y en las malas apoyándome con su gran cariño, comprensión y amor. Gracias por enseñarme con su ejemplo que nunca hay que darse por vencidos y siempre lograr las metas por lo más difíciles que sean.

Agradezco a mis hermanos Alonso y Placido por la compañía y el apoyo que me brindan. Sé que cuento con ellos siempre. A mi Tío Teodoro por su apoyo incondicional.

Agradezco infinitamente por su comprensión, cariño, amor y sobre todo su apoyo incondicional a mi novia Areli ya que sin ella jamás hubiera logrado todo lo cosechado, muchas gracias te amo.

Gracias a la M. en C María Guadalupe Elena Ibargüengoitia González por haber confiado en mi persona, por la paciencia y la dirección de este trabajo. A mis sinodales por los comentarios, concejos y atenta lectura de este trabajo.

Gracias también a mis amigos y compañeros que me acompañaron en esta aventura, que me apoyaron y me enseñaron el gran valor de la amistad. Adriana, Ángela, Carlos, Daniel, Christian, Luis Enrique, Oswaldo, Sandra y Víctor. De manera especial quiero agradecer a Felipe, José Luis y Rebeca por brindarme su gran apoyo ya que sin ellos no podría a ver logrado todo lo que he cosechado.

A mi hermano Rogelio quiero agradecerle por todos los momentos de enseñanza y aprendizaje, pero sobre todo por esa gran amistad que tenemos.

Por último quisiera agradecer a la Dra. Rosa Elba Leyva Huerta y a mis compañeros de trabajo por su apoyo. Aleithia, Aida, Ana, Guillermo, Rafael y Martha.

Índice:

• Introducción

El presente trabajo se refiere a un problema común que se llega a encontrar en empresas, escuelas e instituciones públicas o privadas en las cuales no manejan un factor de éxito conocido como organización de información. Una buena organización garantiza disponer de información precisa, permitiendo una coordinación y comunicación entre los miembros. Otros beneficios de la organización de información es la reducción de costos y esfuerzo, incremento de la ventaja competitiva, aumento de la calidad y satisfacción de sus miembros.

El objetivo de este trabajo es lograr una buena organización de información y así poder tener un buen uso de ella evitando redundancia de datos. Logrando esto se podrá tener la administración de cualquier organización u empresa con la ayuda de un sistema computacional que realice la tarea de resguardar los datos, mostrarlos y actualizarlos.

La utilización de tecnologías y automatizaciones pueden ser muy buenas opciones pero también frustrantes ya que su aplicación genera gigantescas bases de datos, muchas veces en diferentes formatos, evolucionando como subsistemas independientes y multiplicando los esfuerzos; almacenando volúmenes de información dispersa que se ocultan una a otras y frecuentemente son incapaces de ofrecer una visión consolidada de la información.

Es por ello que la aplicación de tecnologías y automatizaciones debe estar orientada hacia un proceso de cambio dinámico evolutivo, donde lo más importante a tener en cuenta son los resultados que las empresas, escuelas e instituciones públicas o privadas deben alcanzar dentro y fuera de la organización, es decir, las salidas del sistema, ya que las entradas al mismo, deben estar condicionadas a las necesidades cambiantes de la organización.

Las tecnologías que se usen durante el proceso de organización deben ofrecer un acceso a todo tipo de información, puede constituir una limitante para las organizaciones incapaces de organizar y convertir la información requerida, en la solución de los problemas reales y obtener de ella suficientes beneficios, tangibles, accesibles y útiles.

Mi interés por lograr una buena organización es conllevar la importancia de organizar la gran cantidad de información que se genera en empresas, instituciones, etc. favorecida por la utilización de las tecnologías de información para convertirla en conocimiento.

La experiencia demuestra que con frecuencia las empresas, instituciones etc. carecen de la información requerida y/o en el tiempo preciso en que la solicitan.

No son pocas las dificultades que tienen para extraer conocimiento de la colección de datos de sus sistemas de información. Por lo que la creación de sistemas debe tener estrategias para lograr el objetivo del mismo y una de ellas es seguir un enfoque de vida de software el cual es una aplicación sistemática, disciplinada y cuantificable del desarrollo, operación y mantenimiento de software [IEEE Standard glossary of Software Engineering Terminology, std610.12-1990].

La estrategia que usaré en este trabajo es la de los Principios de Ingeniería de Software los cuales ofrecen una serie de métodos y técnicas que ayudan al proceso de desarrollo de software. Estos métodos y técnicas están agrupados por metodologías las cuales son reforzadas por herramientas.

Por la manera en que está construido el sistema, existe la posibilidad de conectarlo con algunos otros sistemas de la Coordinación de Posgrado de la Universidad Nacional Autónoma de México ya que la mayoría de los posgrados utilizan el mismo proceso de control para su información. Un ejemplo puede ser los tutores, ya que un tutor puede estar acreditado en uno o varios programas de la coordinación de posgrado y así poder homogenizar una sola tabla de tutores para todos los programas de la Coordinación de Posgrado.

La finalidad de este trabajo es lograr con éxito la elaboración de un sistema de información con herramientas, metodologías, métodos, técnicas y Principios de Ingeniería de Software que deben aplicarse a través de un proceso. El proceso de software me permitirá llevar a cabo actividades que se necesitan para transformar las necesidades de usuarios que utilizaran el producto de software. Para esto lo compondré por fases como está establecido en la Ingeniería de Software.

Mi aportación para el desarrollo del sistema fue como líder de proyecto en términos generales responsable de garantizar el éxito del proyecto, gestionar el plan de trabajo a fin de asegurar que las tareas se terminaran a tiempo. Diseñe el esquema de la base datos así como la programación en Tapestry 5.0. La instalación y configuración de software en los servidores la realice de acuerdo a las tecnologías que se usaron.

Quiero dar las gracias por el apoyo al Lic. Felipe Navarrete Córdoba en la fase de Hibernate ya que sin su comparación no se podría llevar acabo todo este desarrollo.

Capitulo 1

Se establece el proceso de desarrollo que se va aplicar para la elaboración de la herramienta de software junto con las técnicas y procedimientos que nos permitirán conocer los elementos necesarios para definir un proyecto de software. El proceso de compone de fases las cuales empiezan por especificación de requerimientos, etapa de de diseño, construcción y pruebas.

Capitulo 2

La gestión y los diferentes tipos de modelos de bases de datos han evolucionado desde la aplicación computacional especializada hasta convertirse en parte esencial de los entornos computacionales modernos. Por tanto, el conocimiento acerca de los sistemas de bases de datos se ha convertido en parte imprescindible de la formación computacional. Es por eso que se explicará las herramientas a usar en el sistema a desarrollar.

Capitulo 3

Es importante tener en cuenta la tecnología que se usará, se necesitará de tecnología dinámica y altamente escalable. Es por eso que se explicará en este capítulo cada herramienta a utilizarse y el por qué se decidió usarlas en el sistema a realizar.

Capitulo 4

Se explica la forma en que trabaja en la organización a la que se le desarrolla el sistema todo esto para saber sus antecedentes y poder generar un buen producto de software, también se definirá el problema que se quiere resolver y el objetivo al que se quiere llegar. El sistema comprende varios rubros los cuales se explicarán para el funcionamiento de la organización.

Capitulo 5

En este capítulo se define la estrategia adecuada para el desarrollo del producto de software, la estrategia que se usara será la descrita en cada uno de los capítulos anteriores. De acuerdo a lo especificado se elaborarán casos de uso, requerimientos funcionales y no funcionales, diagrama de clases, diseño de bases de datos, construcción y pruebas.

• Capitulo 1 Proceso de Desarrollo de Software

Un proceso de desarrollo es un conjunto de técnicas y procedimientos que nos permiten conocer los elementos necesarios para definir un proyecto de software. Este proceso se compone de fases las cuales empiezan con la especificación de los requerimientos, después procede a la etapa de diseño, luego a la etapa de construcción y por último la etapa de pruebas. Las fases ayudarán a construir los pasos significativos del proceso de software.
Una herramienta que ayudará a especificar, construir, visualizar y documentar al proyecto de software durante el proceso de desarrollo es UML (Unified Modeling Language). El UML es un lenguaje de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos. El modelado no es más que la construcción de un modelo a partir de especificaciones. [Braude p.145]

1.1 Especificación de requerimientos

Algunos libros citan que la especificación de requerimientos es donde se inicia la construcción propiamente del sistema a realizar. Por lo cual es importante entender, capturar y especificar los requerimientos para tener una descripción clara y no ambigua de lo que será el sistema. [Braude p. 184; Ibargüengoitia p 51]. Por otro lado también los requerimientos especifican un proceso de descubrimiento y refinamiento del sistema.

Para construir cualquier sistema es necesario contar con la cooperación del cliente, entender cuál es el problema que se desea resolver y cuáles son sus necesidades reales. Para esto se necesita establecer un vocabulario común.

Tanto el que desarrolla el software del sistema como el cliente tiene un papel activo en la especificación de requerimientos, el cliente intenta formular su concepto de la combinación y comportamiento de los programas a detalle. El que desarrolla el software actúa como interrogador, consultor y refina las necesidades del cliente.

Una característica de los requerimientos es que cambian constantemente por muchas razones: cambian las necesidades del cliente, cambia de parecer, cambia la tecnología, etc. Para esto es necesario establecer un proceso de negociación constante entre el cliente y desarrollador donde ambos aprenden y entienden el objetivo del software en el transcurso del desarrollo.

El proceso de especificación de requerimientos plantea la asignación de software a nivel de sistema y el diseño de programas. Facilita al desarrollador el especificar la integración y comportamiento de los programas, identifica con otros elementos del sistema y establecer las ligaduras de diseño que debe cumplir el programa.

Algunos autores consideran el significado de Especificación de Requerimientos como el construir algo, primero debe entenderse lo que se debe ser ese "algo". El proceso de entender y documentar este algo se le llama "Análisis de Requerimientos". No confundir los significados de especificación de

requerimientos que se describió anteriormente y análisis de requerimientos. El análisis consiste en examinar los requerimientos en su consistencia, completitud y ambigüedad y se clasifican en base a las necesidades de los clientes y usuarios.

La captura de requerimientos termina cuando se obtiene la especificación de los requerimientos que es el documento donde se resume lo que el cliente necesita y que servirá de base a los desarrolladores para analizar esos requerimientos, validarlos, administrarlos y generar el software adecuado. Todo requerimiento debe poderse validar o comprobar que el software lo cumpla. Administrar los requerimientos significa que se lleva un control de los cambios que van sufriendo dichos requerimientos.

Algunos libros de software hacen referencia a los requerimientos funcionales y no funcionales los cuales se consideran muy importantes para la creación de un proyecto.

Los requerimientos funcionales son las entradas y salidas de datos, los cálculos, las funciones o casos de usos, estos especificarán los servicios que deben proporcionar el sistema. Para algunos autores los requerimientos funcionales especifican los servicios que debe proporcionar al sistema. [Ibargüengoitia p 54].

Para especificar los requerimientos funcionales se usarán los casos de uso ya que estos dicen mucho de lo que el sistema intenta hacer. Con frecuencia los requerimientos se expresan de manera natural como una interacción entre el sistema y el usuario. Un caso de uso se identifica primero por su nombre y por el tipo de usuario del sistema. Los casos de uso los usaremos como hilos conductores de todo el proceso de desarrollo, ya que es más fácil identificar los requerimientos y a partir de ellos diseñar, implementar y probar el software. También permite rastrear los requerimientos a través de todo el proceso de desarrollo hasta el producto terminado.

Los requerimientos no funcionales son los requisitos que especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de su comportamiento específico. Por lo que se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar.

Entre los requerimientos no funcionales están:

- **Las necesidades de la interfaz externa: tipo de usuario, hardware, software, comunicaciones, facilidad de uso por los usuarios.**
- **Los atributos del software: eficiencia, disponibilidad, seguridad, conversión, portabilidad y mantenimiento.**
- **Restricciones del diseño: formatos de archivo, lenguajes, estándares, compatibilidad.**
- **Otros: bases de datos, instalación, etc.**

Para elaborar un formato de revisión se puede comprender mejor considerando algunas de las siguientes preguntas:

- **¿Los objetivos y fines establecidos para el programa son consistentes con los objetivos y fines del sistema?**
- **¿Se han descrito todas las interfaces importantes de todos los elementos del sistema?**

- **¿Se ha definido el flujo, contenido y estructura de la información de forma adecuada al dominio del problema?**
- **¿Son los diagramas claros?; ¿Puede cada uno de ellos utilizarse sin el texto suplementario?**
- **¿Permanecen las funciones principales dentro del ámbito y se describen adecuadamente cada una de ellas?**
- **¿Cuál es el riesgo tecnológico del desarrollo?**
- **¿Se han considerado requerimientos alternativos?**
- **¿Se han establecido con detalle criterios de validación?; ¿Son adecuados para describir un buen sistema?**
- **¿Existen inconsistencias, omisiones o redundancias?**
- **¿Se ha estado en un contacto continuo con el cliente?**
- **¿Ha revisado el usuario el manual de usuario o el prototipo?**
- **¿Cómo han afectado las estimaciones del Plan del Proyecto Software?**

Una vez que se ha completado la revisión, se señala como terminada la Especificación de Requerimientos tanto por el cliente como por el desarrollador.

1.2 Diseño

Algunos autores describirían la fase de diseño como el primer paso en la fase de construcción de cualquier producto o sistema de ingeniería. Puede ser definido como: "… el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física" [Braude p. 237; Ibargüengoitia p. 71; Pressman p. 304]

Uno de los objetivos del diseñador es producir un modelo o representación del sistema. El proceso por el cual se desarrolla el modelo combina: intuición y criterios basándose en la experiencia de construir sistemas similares, un conjunto de principios y/o heurísticas que guían la forma en la que se desarrolla el modelo, un conjunto de criterios que facilitan discernir sobre la calidad y un proceso de iteración que conduce finalmente a una representación del diseño final.

Algo muy importante de la fase de diseño es la parte de la construcción de software en que se cambia el enfoque de las necesidades del cliente, a las necesidades del desarrollador y que servirá de base para la construcción del software propiamente dicha. En esta fase se decide el ambiente de implementación que se usará.

Una especificación del diseño es describir las partes o componentes principales de que se compondrán el software, cómo interactúan y cómo se integra en el producto terminado. Ya elaborado se hace un diseño detallado de los componentes tomando en cuenta el ambiente en que se codificará.

Para evaluar diseños alternativos se necesitan principios de diseño y así cumplir con lo que se espera del software. Para realizarlo debemos evaluar las alternativas de acuerdo con los requerimientos no funcionales.

Es muy importante tener una arquitectura de software ya que es donde se definen los componentes que formarán el software.

El Modelo Vista Controlador es un patrón de arquitectura de software que separa la lógica del sistema de la interfaz de usuario, facilita la evolución por separado de ambos aspectos, incrementa reutilización y flexibilidad. Las vistas y controladores suelen estar muy relacionados, los controladores tratan los eventos que se producen en la interfaz gráfica (vista).

Su flujo de control es:

- **El usuario realiza una acción en la interfaz.**
- **El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo.**
- **El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario.**
- **Se genera una nueva vista. La vista toma los datos del modelo (el modelo no tiene conocimiento directo de la vista).**
- **La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo.**

Definida la arquitectura del producto de software, se representa con diagrama de paquetes. Un paquete es un mecanismo general de UML para organizar elementos en grupos, "se usa para modelar elementos que forman un todo" [Booch p. 170], en la representación de la arquitectura representan un componente o funcionalidad del sistema. Un paquete se representa por un "folder" con su nombre. Los elementos de cada paquete tienen una alta cohesión y estar poco acoplados.

En el caso del diseño orientado a objetos los elementos de los paquetes serán clases. Se construyen dos vistas de video del diseño:

- **La vista estática formada por diagramas de clase que representan las clases importantes de la aplicación con sus responsabilidades (atributos y operaciones), sus relaciones que se modelan en uno o varios diagramas de clases.**

- **La parte dinámica que muestra la interacción de esas clases en los casos de uso y se modelan con diagramas de secuencia.**

Para el diseño de la base de datos es esencial la identificación de las consultas e interfaces, se requiere la especificación del flujo, estructura y asociatividad de la información y debe desarrollarse un documento formal de los requerimientos.

1.3 Construcción

En esta etapa se desarrolla el código del software. El resultado de este proceso es un producto listo para que los usuarios lo puedan operar y consiste en un software integrado en las plataformas adecuadas, los materiales para soporte del usuario y una descripción de la versión actual (versión

"beta") para esto se describe las actividades del sistema, como construcción del código, diseño detallado, programación, planear y aplicar pruebas unitarias.

Las pruebas unitarias permiten aislar cada parte del programa y mostrar que cada parte individual es correcta. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

1. **Fomentan el cambio: Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.**
2. **Simplifica la integración: Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.**
3. **Documenta el código: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.**
4. **Separación de la interfaz y la implementación: Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando objetos mock (mock object) para simular el comportamiento de objetos complejos.**
5. **Los errores están más acotados y son más fáciles de localizar: dado que tenemos pruebas unitarias que pueden desenmascararlos.**

Para probar cada unidad lógica (clases) se elaboran pruebas unitarias, esto para que la calidad de las clases sea buena. Un caso de prueba de un método consiste en definir un conjunto representativo de valores para los parámetros del método y el valor del resultado esperado. Para cada método se puede definir uno o más casos de prueba según la complejidad del método.

Para tener las pruebas unitarias de cada clase que se está construyendo se elabora un plan de pruebas unitarias. Algunos autores definen el plan de pruebas unitarias como: las clases que probarán, los métodos y los casos de prueba para cada método. Una forma de especificar este plan con éxito es haciendo una tabla con:

- **La clase que se probará.**
- **Método a probar.**
- **Los casos de prueba con los conjuntos de valores para los parámetros para cada método.**
- **El resultado esperado de cada caso de prueba.**

Se pueden realizar varios casos de prueba pero para determinar que un requisito es completamente satisfactorio se emplean técnicas para definir los casos de pruebas las cuales son:

- **Pruebas de caja blanca**

Se toman en cuenta la estructura del código de un método y se busca que durante las pruebas cada instrucción se ejecute al menos una vez.

- **Pruebas de caja negra**

Esta prueba revisa que la unidad cumpla con la funcionalidad esperada sin considerar el detalle del código. Para definir los casos de prueba, se establecen los posibles resultados esperados de la unidad y se identifican los conjuntos de valores de los parámetros, para que se generen estos resultados.

Estas dos técnicas darán seguridad al momento de comprobar si se tiene todos los componentes completos y que funcionan bien.

1.4 Pruebas

La importancia de las pruebas de integración de software y sus implicaciones con la calidad del software no se debe subestimar por lo cual se especifica las actividades necesarias para la fase de pruebas e integración y cumplan con los requerimientos antes mencionados.

Cuál es el objetivo de las pruebas:

- **La prueba es un proceso de ejecución de un programa con la intención de descubrir un error**
- **Un proceso caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.**
- **Una prueba tiene éxito si descubre un error no detectado hasta entonces.**

Uno de los objetivos es tener un diseño de pruebas que sistemáticamente saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y esfuerzo. Se debe llevar a cabo el plan de integración planteado y desarrollado en el diseño y complementarlo de acuerdo a la implementación para así realizar las pruebas.

Debemos realizar un flujo de información para las pruebas el cual podemos seguir con el siguiente esquema.

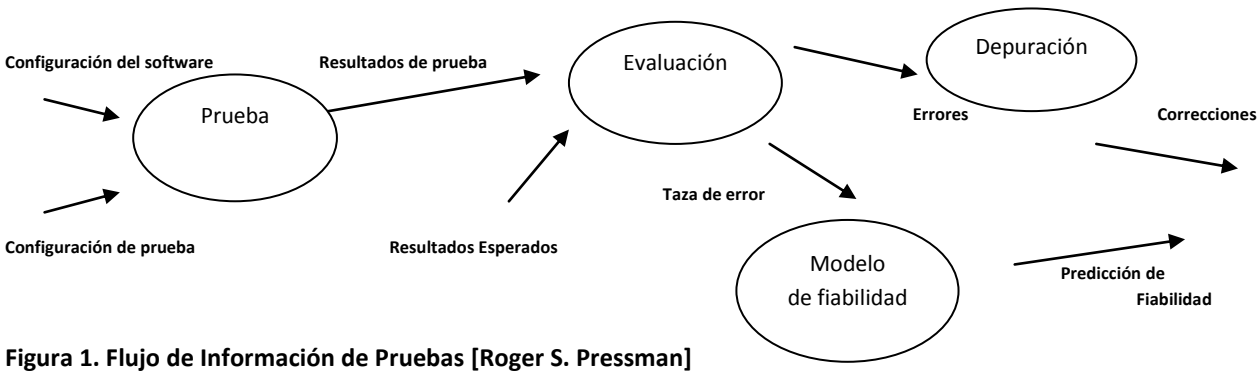


Figura 1. Flujo de Información de Pruebas [Roger S. Pressman]

Describe dos clases de entrada al proceso de prueba 1) una configuración del software que incluye la especificación de requerimientos del software, la especificación del diseño y el código fuente; 2) una configuración de prueba que incluye un plan y procedimientos de prueba, casos de prueba y resultados esperados.

Para la integración se verifica que se tengan todos los componentes y que juntos funcionen bien desde el nivel inferior hasta el superior.

Es importante decir que no existe una estrategia adecuada para todos los sistemas, decidir cual usar es decisión del equipo o desarrollador, el proyecto y el nivel de diseño efectuado.

Teniendo todo integrado se realizan las pruebas del sistema, se tratará de comprobar que el sistema cumple con todos los requerimientos establecidos tanto los funcionales como los no funcionales. Es muy difícil probar al cien por ciento todos los aspectos del sistema, se debe planear qué se le dará prioridad.

Para probar el sistema se usarán los siguientes aspectos definiendo casos de prueba para:

- **La instalación del sistema**
- **El arranque del sistema**
- **La usabilidad del sistema con la participación de usuarios reales**
- **Se debe incluir pruebas de los otros requerimientos no funcionales como:**
  - **Rendimiento.**
  - **Robustez-recuperación de errores de hardware.**
  - **Tiempo de respuesta.**
  - **Capacidad de carga (memoria, accesos).**
  - **Estimación de número de defectos por día.**
  - **Estimación de tiempo de corrección de defectos.**
  - **Estimación de tiempo total de prueba de sistema y corrección**

Al efectuar las pruebas, se corrigen los defectos encontrados.

- # Capitulo 2 Planteamiento del Problema

2.1 Antecedentes o contexto

Programa de Maestría y Doctorado de Ciencias Médicas, Odontológicas y de la Salud.

El Programa de Maestría y Doctorado de Ciencias Médicas, Odontológicas y de la Salud tiene como objetivo formar recursos humanos de alto nivel de preparación para el ejercicio académico, profesional o ambos; aptos para fungir como los líderes de la medicina y la odontología mexicanas; capaces de contribuir a su mejoramiento, generado, preservado, aplicando y transmitiendo conocimientos avanzados en los campos de la medicina y la odontología; comprometidos socialmente con la solución de los problemas nacionales, especialmente los relacionados con la salud, desde la perspectiva del conocimiento universal; competentes en el contexto de la globalización económica y social; capaces de trabajar en equipo y en redes académicas y sociales; y poseedores de valores humanos y espíritu de servicio.

Están involucrados 7 campos del Conocimiento:
Ciencias Médicas
Ciencias Odontológicas Clínicas
Ciencias odontológicas básicas
Ciencias de la Salud
Ciencias Sociomédicas
Humanidades en Salud
Investigación clínica experimental en salud

Cada campo cuanta con sus campos de estudio principal y sus entidades académicas.
Campos de Estudio Principal

Antropología en Salud
Bioética
Biología Bucal
Biomateriales Dentales
Bioquímica Clínica
Ciencias Médicas
Cirugía Oral y Maxilofacial
Educación en Ciencias de la Salud
Educación Médica
Educación Odontológica
Endodoncia
Epidemiologia Clínica
Farmacia Clínica
Farmacología Clínica
Historias de las Ciencias de la Salud
Gestión y Políticas en Salud
Materiales Dentales
Odontología Comunitaria
Odontopediatria
Ortodoncia
Patología Bucal
Periodoncia e Implantología
Prótesis Maxilofacial
Salud en el Trabajo

Salud Mental Pública
Salud Pública Bucal
Sistemas de Salud
ENTIDADES ACADÉMICAS PARTICIPANTES:

Facultad de Filosofía y Letras

Facultad de Medicina

Facultad de Odontología

Facultad de Psicología

Facultad de Química

Facultad de Estudios Superiores Iztacala

Instituto Nacional de Perinatología "Isidro Espinosa de los Reyes"

Instituto Nacional de Cardiología "Ignacio Chávez"

Instituto Nacional de Pediatría

Instituto Nacional de Rehabilitación

Facultad de Estudios Superiores Zaragoza

Instituto de Investigaciones Biomédicas

Instituto Nacional de Ciencias Médicas y Nutrición "Salvador Zubirán"

Instituto Nacional de Neurología y Neurocirugía "Manuel Velasco Suárez"

Instituto Nacional de Psiquiatría "Ramón de la Fuente Muñiz"

Hospital Infantil de México, "Federico Gómez

Hospital General de México

Instituto Mexicano del Seguro Social

Instituto de Oftalmología "Fundación Conde de Valenciana IAP"

2.2 Problema y Objetivo

Debido a la gran diversidad de información que se tiene dentro del programa del Posgrado de Ciencias Médicas, Odontológicas y de la Salud, es necesaria la automatización a través de una aplicación Web que administre todos los datos de los aspirantes y alumnos así como de tutores dentro del programa.

El objetivo general es la implementación de un sistema que facilite a los administradores y capturistas el registro de alumnos, aspirantes y tutores que se encuentran dentro del programa de posgrado a fin de mantener los datos íntegros y almacenados correctamente, para evitar redundancia de datos y tener un control de las personas que tienen acceso a esta información a través de la administración del sistema.

Actualmente la Coordinación de Posgrado de Ciencias Médicas, Odontológicas y de la Salud no cuenta con un sistema. Los datos que se obtienen del registro se guardan en archivos Excel que se manejan como base de datos, las búsquedas, inscripciones y demás tareas se hacen en el mismo archivo Excel, lo que conlleva a problemas como falta de información, datos sucios o redundantes, lentitud en el proceso, etc.

Este sistema lo utilizarán las personas encargadas del registro de aspirantes, alumnos y tutores que dan clase o son asesores de tesis y que también fungen como sinodales. Al mantener toda la

información de la Coordinación integra se lograra que todo el proceso que se realiza en un tiempo a largo plazo se realice a corto plazo como son: evaluaciones, estadísticas, escritos, búsquedas, almacenamiento y actualización de datos.

Para las construcción de los módulos del sistema, se seguirá con el proceso aprendido en el curso de Ingeniería de Software y se mostraran todos los documentos que se van generando, de forma que los módulos quedaran bien documentados a fin de que se puedan complementar con otros módulos útiles a la Coordinación de Posgrado de Ciencias Médicas, Odontológicas y de la Salud.

2.3 Módulos

El sistema comprende varios rubros como son la inscripción de aspirantes, búsquedas de aspirantes, alumnos y tutores, apoyo a los tutores al seguimiento de sus alumnos y la administración de usuarios y de todo el sistema. Todo estos rubros son importantes para el funcionamiento del programa por lo cual es importante implementar estas funcionalidades para simplificar las tareas de los usuarios. Los usuarios tendrán restricciones de uso en todos los módulos.

Módulo 1 Inscripción de Aspirantes

Dentro del programa se pide un registro de inscripción de los Aspirantes para seguir el proceso desde que piden ingresar al programa, esto se hará con un cuestionario donde se piden datos personales, socioeconómicos, antecedentes académicos, antecedentes laborales, campo de conocimiento, entidad académica, campo de estudio principal, nivel de estudio, tutor y en que semestre quiere ingresar. Se generará un reporte con todos sus datos en caso de que se quieran consultar. También se guardaran sus datos en la base de datos para actualizarlos cuando sea aceptado como alumno. Todo esto sirve para las estadísticas del Programa.

Registrado el aspirante puede realizar las etapas de selección que son pagos, exámenes de admisión y la publicación de resultados de exámenes. El aspirante al tener su carta de aceptación del programa personal encargado del sistema cambia al aspirante a alumno llevándose automáticamente todos sus datos.

Módulo 2 Búsquedas alumnos, aspirantes y tutores

Las búsquedas son un factor importante dentro del sistema ya que si ellas no podríamos consultar los datos de alumnos, aspirantes y tutores individualmente o masivamente.

- **Datos Personales**
- **Profesional**
- **Estudios**
- **Contacto**

Se podrá editar y eliminar toda la información.Toda esta información se almacenará de Exceles[1] a una base de datos, el proceso será minucioso ya que se hará limpieza de datos y así se logrará la carga de datos exitosamente.

[1] Los exceles los proporcionara la Coordinación

Búsqueda de Aspirantes

El sistema nos permitirá buscar por semestre, estado, grado y campo de conocimiento donde todo esto nos ahorra tiempo ya que nos mostrara los nombres de todos los aspirantes registrados, todo esto según la combinación que hagamos con los campos de las búsquedas. Esta parte servirá para que los usuarios encargados de almacenamiento y actualización de datos puedan realizar la conversión de aspirante a alumno.

Búsqueda de Alumnos

La información se encuentra de manera que los usuarios pueden consultar individual y masivamente con ayuda de la combinación de campos que tendrá el sistema.

Búsqueda de Tutores

Lo tutores se podrán consultar individual y masivamente con la combinación de campos que tiene el sistema.

Módulo 3 Administración de usuarios y del sistema

Es muy importante este modulo debido a que se efectuara toda la administración del sistema así como sus restricciones para los usuarios. Habrá niveles de usuarios Administrador, capturista y tutor.

Usuario de administración

Las tareas que el administrador podrá realizar en el sistema son agregar y eliminar usuarios; consultar, eliminar y registrar datos de usuarios, tutores, alumnos y aspirantes.

Usuario capturista

Podrá consultar, editar datos de aspirantes, alumnos y tutores, también podrá eliminar datos siempre y cuando se le den los permisos.

Usuario tutor

Podrá consultar solo sus datos como son datos personales, estudios, tutorías, jurados de candidatura y grado y comités tutoriales. No podrá eliminar ningún dato, solo notificara al administrador algún cambio en dichos datos.

Las consultas que se pueden ejecutar se realizarán utilizando diferentes criterios que al usuario le hará más eficiente dicha búsqueda.

• Capitulo 3 Concepto de Bases de Datos

La tarea de creación de aplicaciones de bases de datos es una labor compleja, implica varias fases, como el diseño del esquema de la base de datos, el diseño de los programas que tienen acceso a los datos y los puedan actualizar y el diseño del esquema se seguridad para controlar el acceso de datos

3.1 Sistema Gestor de Bases de Datos

Un Sistema Gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos (base de datos) contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

Algunas de las características deseables en un SGBD son:

- **Control de la redundancia: La redundancia de datos tiene varios efectos negativos (duplicar el trabajo al actualizar, desperdiciar espacio en disco, puede provocar inconsistencia de datos) algunas veces es deseable por cuestiones de rendimiento.**

- **Restricción de los accesos no autorizados: cada usuario ha de tener unos permisos de acceso y autorización.**

- **Cumplimiento de las restricciones de integridad: el SGBD ha de ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad.**

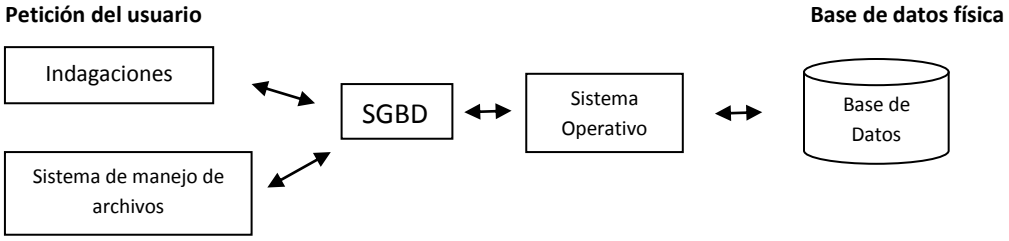


Figura 2. Modelo del SGBD

Se muestra en la Figura como el SGBD está como interface entre la base de datos física y las peticiones del usuario. El SGBD interpreta las peticiones de entrada/salida del usuario y las manda al sistema operativo para la transferencia de datos entre la unidad de memoria secundaria y la memoria principal.

3.2 Modelos de Bases de Datos

Bajo la estructura de las bases de datos se encuentra el modelo de datos, el cual es una colección de herramientas conceptuales para describir los datos, sus relaciones, su semántica y las restricciones de

consistencia. Los modelos de datos ofrecen un modo de describir el diseño de las bases de datos en los niveles físico, lógico y de vista.

La estructura en que se basa el modelo de datos es:

- **Estáticas: Son las propiedades invariantes en el tiempo. Quedan especificadas en el modelo de datos por estructuras.**

- **Dinámicas: Son las propiedades que varían con el tiempo. En el modelo de datos son las operaciones**

3.3 Bases de datos relacionales

El modelo relacional es hoy en día el principal modelo de datos para las aplicaciones comerciales de procesamiento de datos. Ha conseguido esa posición destacada debido a su simplicidad, lo cual facilita el trabajo del programador en comparación con otros modelos. Consisten en un conjunto de tablas, a cada una de las cuales se le asigna un nombre exclusivo, cada fila de la tabla representa una relación entre un conjunto de valores. De manera informal, cada tabla es un conjunto de tales relaciones, hay una fuerte correspondencia entre el concepto de tabla y el concepto matemático de relación, del que toma su nombre el modelo de datos relacional

- **Modelo Relacional**

Usa una colección de tablas para representar tanto los datos como sus relaciones. Cada tabla tiene varias columnas y cada columna contiene un nombre único dentro de la tabla. El modelo relacional es un ejemplo de un modelo basado en registros. Los modelos basados en registros se denominan así porque la base de datos se estructura en registros de formato fijo de varios tipos. Cada tabla contiene registros de un tipo dado. Cada tipo de registro define un número fijo de campos o atributos. Las columnas de la tabla se corresponden con los atributos del tipo de registro.

¿Cuál es el objetivo de una base relacional?

Es la generación de un conjunto de esquemas de relación que permita almacenar la información sin redundancias innecesarias, pero que también permita recuperarla fácilmente.

- **Modelo Entidad-relación**

Se basa en una percepción del mundo real que consiste en una colección de objetos básicos, denominados entidades y de las relaciones entre ellos. Una entidad es una "cosa" u "objeto" del mundo real que es distinguible de otros objetos. Se desarrollo para facilitar el diseño de bases de datos permitiendo la especificación de un esquema de la empresa o cliente que representa la estructura lógica global de la base de datos. El modelo de datos E-R es uno de los diferentes modelos de datos semántico; el aspecto semántico del modelo radica en la representación del significado de los

datos. El modelo E-R resulta muy útil para relacionar los significados e interacciones de las empresas o clientes reales con el esquema conceptual. Debido a esta utilidad, muchas herramientas de diseño de bases de datos se basan en los conceptos del modelo E-R. También emplea tres conceptos básicos: los conjuntos de entidades, los conjuntos de relaciones y los atributos.

3.4 Normalización de Bases de Datos

El proceso de normalización se encarga de seguir una serie de pasos o normas, de tal forma que es la estructura óptima para su implementación, gestión y explotación desde diferentes futuras aplicaciones. Una tabla se dice cuando que está en una forma normal cuando satisface un conjunto de restricciones impuestas por dicha norma.

La normalización se basa en que los datos son independientes de las aplicaciones que los gestionan, y su objetivo es obtener el mayor número de tablas posibles, dejando en cada una de ellas los atributos imprescindibles para representar a la entidad (objeto), o a la relación entre entidades a la que hace referencia la tabla mediante la conexión de sus claves.

Las ventajas que se obtiene tras la normalización de datos para su eficaz gestión es:

- **Facilidad de uso: Los datos están agrupados en tablas que identifican claramente un objeto o una relación.**

- **Flexibilidad: La información que necesitan los usuarios se pueden obtener de las tablas relacionales o relaciones mediante las operaciones del algebra relacional. Por ejemplo, uniendo tablas, seleccionando sus valores y proyectándolos.**

- **Precisión: las interrelaciones entre las tablas consiguen mantener información diferente relacionada con toda exactitud.**

- **Seguridad: Los controles de acceso para consultar o actualizar información (a nivel de tablas o a nivel de atributos) son mucho más sencillos de implementar.**

- **Facilidad de implementación: Las tablas se almacenan físicamente como archivos planos.**

- **Independencia de datos: Los programas no están ligados a las estructuras, con lo que se consigue aumentar la base de datos añadiendo nuevos atributos o nuevas tablas sin que afecten a los programas que las usan.**

- **Claridad: La representación de la información es clara y sencilla para el usuario; son tablas simples.**

- **Facilidad de gestión: Los lenguajes manipulan la información de forma sencilla al estar los datos basados en algebra y cálculo relacional.**

- **Mínima redundancia: La información no estará duplicada innecesariamente dentro de las estructuras.**

- **Máximo rendimiento de las aplicaciones: Sólo se trata aquella información que se va servir de utilidad a cada aplicación.**

El proceso de normalización lo realiza el analista tras sucesivas reuniones que mantienen con el usuario y toda la información que se recibe se debe documentar. [Lucas Gómez pag. 86 y 87]

Para tener una normalización con éxito debemos seguir las Formas Normales (FN), las cuales son 1FN, 2FN, 3FN y posteriormente unas anomalías detectas forzaron a crear una forma normal más completa que la 3FN, son definidas como 4FN Y 5FN.

Primera Forma Normal (1FN).

Una tabla para cumplir la 1FN si y solo si un atributo 'solo debe de mantener valores elementales o únicos'.

Segunda Forma Normal (2FN).

Una tabla para cumplir la 2FN debe encontrarse en 1FN y que todo atributo secundario dependa totalmente de la clave completa y por tanto no de una parte de ella. Es decir la clave principal de la tabla está formada por un único atributo.

Tercera Forma Normal (3FN).

Una tabla para cumplir la 3FN debe encontrase en 2FN y que no existan atributos no primarios que son transitivamente dependientes de casa posible clave de la tabla. Esto quiere decir que un atributo secundario solo se debe conocer a través de la clave principal o claves secundarias de la tabla y no por medio de otro atributo.

Forma Normal de BOYCE-CODD (FNBC)

Una tabla esta en FNBC si y solo si las únicas dependencias funcionales elementales con aquellas en las que la clave principal y las claves secundarias determinan un atributo. Esto quiere decir que si la clave está formada por un solo atributo, la tabla esta está en FNBC.

3.5 SQL

Los sistemas de bases de datos necesitan un lenguaje de consultas cómodo para el usuario. Structured Query Language (SQL) es el lenguaje de consultas distribuido comercialmente de más influencia.

Con SQL es posible definir la estructura de datos, modificar los datos de la base de datos y especificar restricciones de seguridad. Tiene varios componentes como:

Lenguaje de definición de datos (LDD): Proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificación de los esquemas de relación.

Lenguaje interactivo de manipulación de datos (LMD): Incluye un lenguaje de consultas basado tanto en la algebra relacional, como en el cálculo relacional de tuplas. También contiene comandos para insertar, borrar y modificar tuplas.

Restricciones de Integridad: El LDD incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos. Las actualizaciones que violan las restricciones de integridad se rechazan.

Controlador de transacciones: Incluye comandos para especificar el comienzo y el final de las transacciones.

SQL incorporando y SQL dinámico: Definen como se pueden incorporar instrucciones de SQL en lenguajes de programación de propósito general como C, C++, Java, PL/I.

Autorización: El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

SQL es uno de los éxitos de la industria de la computación y nos servirá como lenguaje de consulta ya que las bases de datos relacionales están formadas por un conjunto de relaciones, a cada una de las cuales se le asigna un nombre único. Permite el uso de valores nulos para indicar que el valor es desconocido o no existe. También permite al usuario especificar los atributos que pueden contener valores nulos.

• Capitulo 4 Tecnologías a usar

Es muy importante tener en cuenta la tecnología que se usará ya que un sistema robusto como el que se desarrollará se necesitará de tecnología dinámica y altamente escalable. También es importante decir que se usa software libre ya que son tecnologías de competitividad, baratas, con mayor calidad y su adquisición es sencilla.

En busca de un software apto para el sistema se encontró con frameworks y manejadores de base de datos muy interesantes la mayoría resolvía las necesidades de desarrollo. La decisión fue difícil pero se decidió por Tapestry 5, Hibernate, Spring, Apache tomcat, Postgresql, Ajax y JavaScript ya que cumplen con las expectativas que se quieren lograr con la creación del sistema que se desarrollara.

Para el hardware se especificó que para este sistema se necesitaría un servidor Web con alta capacidad de disco duro y memoria RAM para la instalación del sistema operativo Linux. Se consideró que Linux puede cumplir con las necesidades del software elegido.

4.1 AJAX

Para crear aplicaciones interactivas se necesita técnicas de desarrollo web, es por eso que se eligió Ajax ya que es una tecnología asíncrona, esto quiere decir que se ejecuta en el navegador mientras se mantienen en comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en los sistemas.

La ventaja de Ajax es que no es una tecnología en sí mismo, si no varias tecnologías independientes que se unen de formas nuevas y sorprendentes. Como por ejemplo incorpora:

- **XHTML y CSS, para crear una presentación basada en estándares.**
- **DOM, para la interacción y manipulación dinámica de la presentación.**
- **XML, XSLT y JSON, para el intercambio y la manipulación de información.**
- **XMLHttpRequest, para el intercambio asíncrono de información.**
- **JavaScript, para unir todas las demás tecnologías.**

En las aplicaciones web tradicionales, las acciones del usuario en la página (dar clic en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:

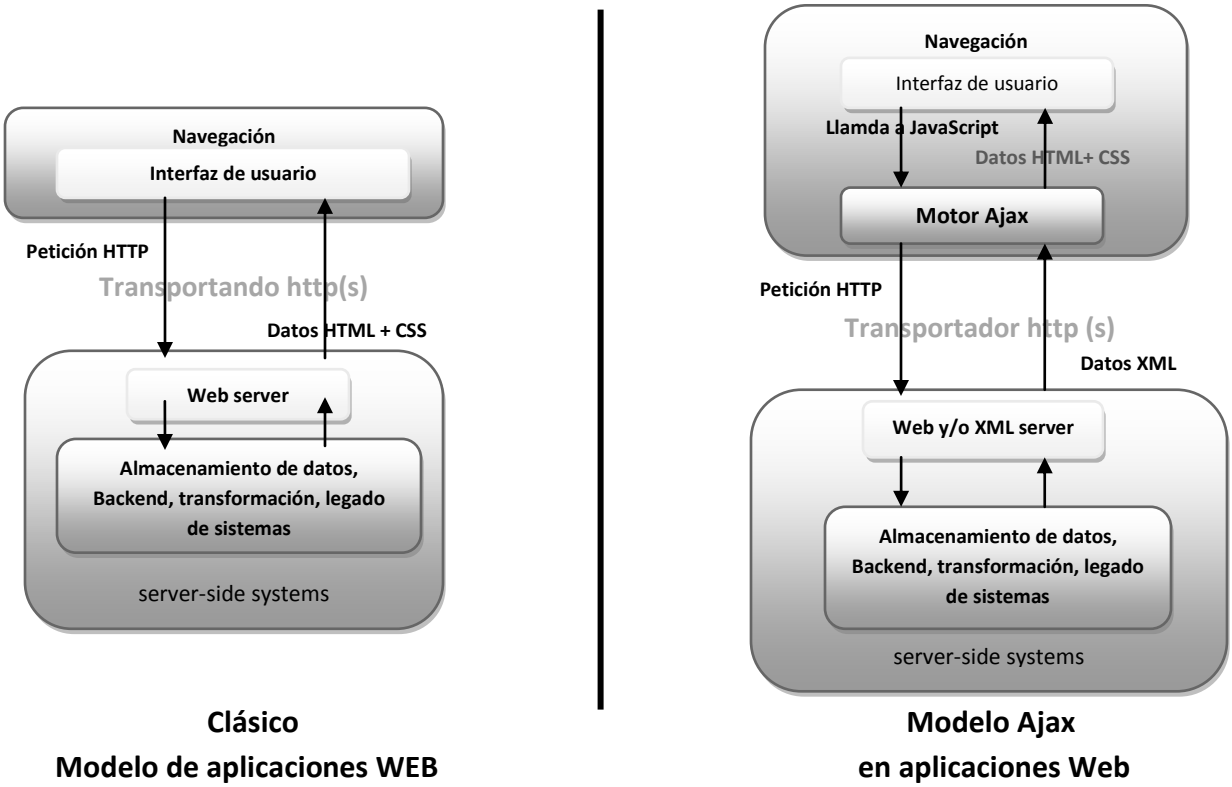


Figura 3. Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX. (Imagen original creada por Adaptive Path)

4.2 POSTGRESQL

PostgreSQL es un potente sistema manejar de bases de datos de código abierto, relacional-objeto, que ejecuta procedimientos almacenados en más de una docena de lenguajes de programación, como Java, Perl, Python, Ruby, Tcl, C / C + +, y su propia PL / pgSQL, que es similar a la de Oracle PL / SQL.

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

PostgreSQL tiene algunos componentes generales importantes en un sistema.

Aplicación cliente: Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP ó sockets locales.

Demonio postmaster: Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes

Ficheros de configuración: Los 3 ficheros principales de configuración utilizados por PostgreSQL, postgresql.conf, pg_hba.conf y pg_ident.conf

Procesos hijos postgres: Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes

PostgreSQL share buffer cache: Memoria compartida usada por PostgreSQL para almacenar datos en caché.

Write-Ahead Log (WAL): Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO)

Kernel disk buffer cache: Caché de disco del sistema operativo

Disco: Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione

Sus características más importantes son:

- **Atomicidad (Indivisible) es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.**
- **Consistencia es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.**
- **Aislamiento es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.**
- **Durabilidad es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.**
- **Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, etc.**
- **Drivers: Odbc, Jdbc, .Net, etc.**
- **Soporte de todas las características de una base de datos profesional (triggers, store procedures-funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.)**

- **Soporte de protocolo de comunicación encriptado por SSL**
- **Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.**

4.3 TAPESTRY

Tapestry es un framework de código abierto para la creación de aplicaciones web dinámicas, robustas altamente escalables en Java, lo que permite asumir la responsabilidad de las preocupaciones clave, tales como la construcción de URL, el envío, almacenamiento de estado persistente en el servidor, la validación de entrada del usuario, la localización / internacionalización, y reportes de excepción.

Tapestry ofrece las herramientas especificas para la construcción como crear tareas típicas de cualquier sistema como la visualización de resultados en páginas y la validación de formularios, de una manera conjunta (por componentes) y de una forma muy simple. Tapestry es orientado a componentes es decir, páginas web donde cada una de ellas se construirá de componentes que puedan generar diversos eventos ante los que la pagina puede reaccionar automáticamente. Algunos ejemplos, son páginas interactivas que recuerdan los valores introducidos por el usuario en los campos de entrada también cuando se hace un clic sobre un enlace o un botón, puede reaccionar a esa acción ejecutando un método que actúa como manejador del evento producido por ese componente.

Otro apoyo, solución, es la creación de plantillas HTML usando HTML plano, y la combinación de las plantillas con pequeñas cantidades de código Java. Se crea la aplicación en términos de objetos de java, y los métodos y propiedades de los objetos en concreto y no en términos de direcciones URL con los parámetros de consulta. Lo más importante es que se desarrolla orientado a objetos fieles a las aplicaciones web en Java, así como la integración a la perfección con otros frameworks por ejemplo HIbernate.

Otra ventaja que tiene Tapestry es de dar solución a los problemas fundamentales del desarrollo con Java que se tiene con los JSP, que no respetan el Modelo Vista Controlador, al permitirnos introducir cualquier código Java dentro de la vista de las páginas y la cantidad de configuración en XML que se tienen que generar.

Tapestry usa un gestor de proyectos llamado Maven. El objetivo de Maven es permitir comprender el estado completo del esfuerzo de desarrollo en el menor período de tiempo. Para alcanzar este objetivo hay varias áreas de preocupación en que Maven intenta resolver:

- **Hacer el proceso de construcción fácil.**
- **Proporcionar un sistema de construcción uniforme.**
- **Proporcionar información sobre el proyecto de calidad.**
- **Proporcionar pautas para el mejor desarrollo de las prácticas.**
- **Permitir la migración transparente a nuevas características.**

Algunas características principales por las que Tapestry es un framework a utilizar son:

- **Requiere un mínimo de configuración en XML.**

- Las páginas se dividen en dos archivos: Uno con el HTML y otro con la clase en Java que contiene toda la funcionalidad.

- Las páginas en Java con POJOs (Plain Old Java Object), no es necesario heredar de ninguna clase ni implementar ninguna interfaz.

- El esquema de funcionamiento se basa en eventos, es decir, una liga o una forma pueden generar uno o varios eventos distintos que se pueden interceptar en la página en Java.

- Las páginas pueden tener varias propiedades que apunten a componentes comunes de la aplicación; el manejador de inyección de dependencias de Tapestry se encarga de configurar la página, basado en las anotaciones que se pongan a las variables de la clase en Java.

- Permite por una parte tener una abstracción del ciclo de petición-respuesta típico de un sistema web pero por otra parte permite tener un control de bajo nivel tan refinado como sea necesario.

- Cualquier página puede ser a su vez un componente y ser incluido dentro de otras páginas o componentes; puede recibir parámetros del componente que lo incluya.

- Tiene buena integración con AJAX, permite respuestas parciales en las páginas y componentes, maneja datos en formato JSON, incluye la librería de JavaScript Prototype.

- Durante el desarrollo se pueden modificar las clases Java y la siguiente invocación va a recargar la clase sin necesidad de reiniciar la aplicación o recargarla por completo, acelerando así la velocidad del desarrollo.

Otra ventaja es que se puede usar una aplicación de Tapestry donde se puede configurar un servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad como Tomcat que será configurado en el IDE a utilizar.

Tapestry nos da una manera de tener un orden en cada página ya que tiene asociada una plantilla y una clase Java donde ambas deben de tener el mismo nombre. El objetivo de la clase java es recibir los eventos relativos a la página y el objetivo de la plantilla es generar un markup (XML) de la página. Este markup tiene una extensión .tml (Tapestry Markup Language), cuando se genera presenta cada componente de la plantilla mostrando en una página todo el código correspondiente.

4.4 SPRING

Permite la manipulación de objetos en tiempo real (de manera similar a Expression Language en JSP/JSF), además de otras interesantes características como la invocación de métodos.
Usar spring es muy práctico ya que algunos Integrated Development Environment (IDE) ya tienen SpringSource Tool Suite que permiten autocompletado de expresiones además de otras características

realmente útiles para trabajar con cualquier lenguaje. En caso de usar maven en los proyectos con Spring solo es necesario solo agregar una dependencia.

Puede ser usado:

- **De forma nativa en una aplicación Spring (insertado en archivos XML o anotaciones)**

- **A través de un parser, creando explícitamente toda la infraestructura necesaria para interpretarlo**

Spring puede ayudar al flujo Web ya que tiene un modulo en el cual es modelo vista controlador que permite definir los controladores con un dominio especifico de lenguaje. Este lenguaje está diseñado para la interacción del usuario que requiere de varias peticiones en el servidor para completar, o puede ser invocado desde diferentes contextos.

4.5 HIBERNATE

Hibernate permite desarrollar clases persistentes para las clases de los lenguajes orientados a objetos incluyendo conceptos como la herencia, polimorfismo, asociación, composición y el marco de las colecciones de Java. También permite trabajar con bases de datos relacionales donde no requiere de tablas especiales o campos y genera gran parte del SQL en el momento de inicialización del sistema y no en el tiempo de ejecución.

Es una herramienta que realiza mapeos entre el mundo orientado a objetos y el mundo de entidad relación de las bases de datos en entornos Java. El termino correcto es ORM (object/relational/mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa. Proporciona capacidades para la obtención y almacenamiento de datos de la base de datos que nos reduce el tiempo de desarrollo.

HIbernate funciona asociando a cada tabla de la base datos un Plain Old Java Object (POJO). Un POJO es similar a un Java Bean, con propiedades accesibles mediante métodos setter y getter, como por ejemplo:

```
package net.sf.hibernate.examples.quickstart;

public class Cat {

private String id;
private String name;
private char sex;
private float weight;

public Cat() {
}

public String getId() {
return id;
}

private void setId(String id) {
this.id = id;
}

public String getName() {
return name;
}
```

```
public void setName(String name) {
this.name = name;
}

public char getSex() {
return sex;
}

public void setSex(char sex) {
this.sex = sex;
}

public float getWeight() {
return weight;
}

public void setWeight(float weight) {
this.weight = weight;
}

}
```

Figura 5. Método setter y getter

Ofrece un rendimiento superior sobre el JDBC de decodificación y es altamente personalizable y extensible. Soporta inicialización perezosa, muchas estrategias para búsquedas, y el bloqueo optimista con versiones automáticas y selladas de tiempo.

Para configurar Hibernate es necesario crear un XML con todas las propiedades que queremos que contenga. Un ejemplo de este archivo XML está en db.posgrado, como se muestra en la figura 6, que es una biblioteca o paquete que se importa desde posgrado-web.

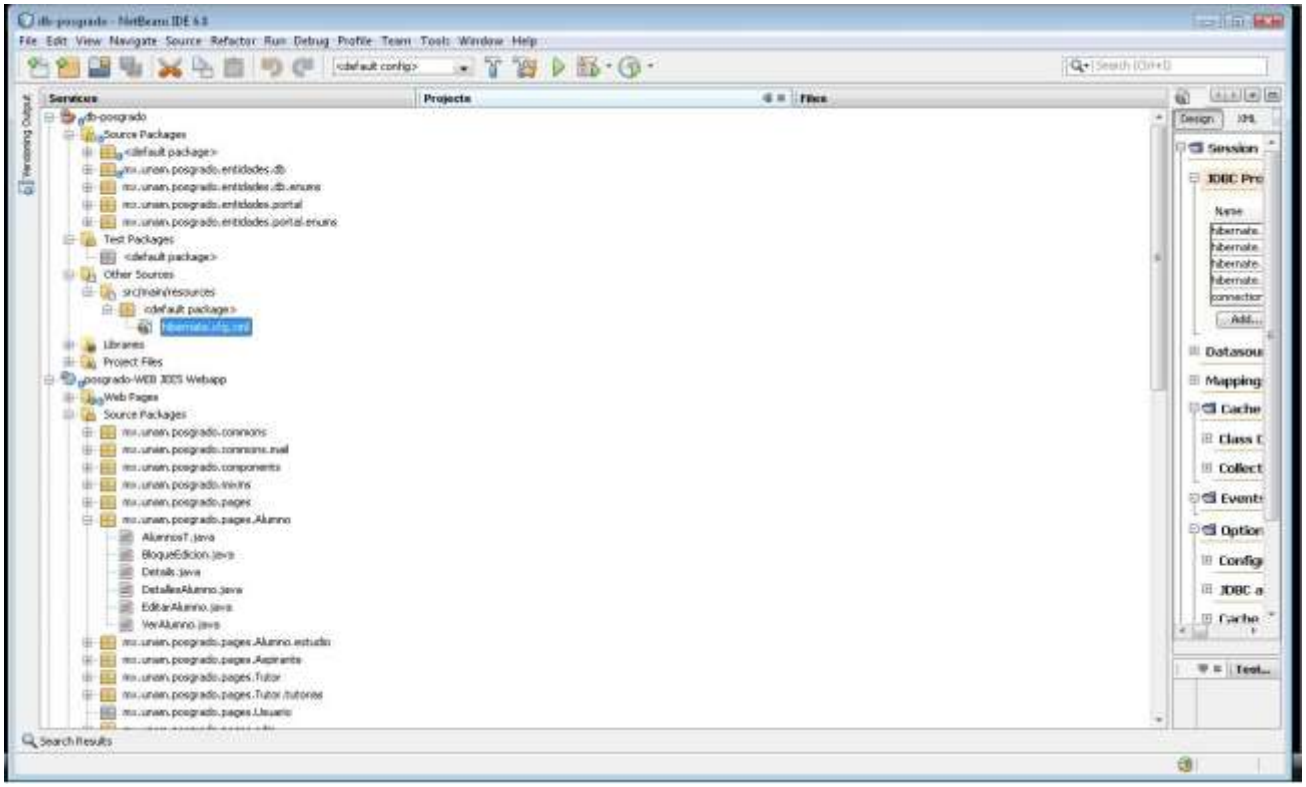


Figura 6. Archivo Hibernate.cfg.xml

Las propiedades son:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/Posgrado</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.connection.password">cmoscmos</property>
<!-- JDBC connection pool (use the built-in) -->
<property name="connection.pool_size">1</property>
<!-- Enable Hibernate's automatic session context management -->
<property name="current_session_context_class">managed</property>
<!-- Disable the second-level cache-->
<property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>
<!-- Echo all executed SQL to stdout -->
<property name="show_sql">false</property>
<property name="hibernate.format_sql">false</property>
<property name="hibernate.use_sql_comments">true</property>
<property name="hibernate.transaction.auto_close_session">false</property>
<property name="hibernate.hbm2ddl.auto">validate</property>
<property name="hibernate.default_batch_fetch_size">200</property>
<property name="hibernate.max_fetch_depth">0</property>
<property name="hibernate.bytecode.use_reflection_optimizer">true</property>
```

La conexión con la Base de datos es muy importante, es por eso que las propiedades deben ser bien configuradas, como se observa en las propiedades, ya que sin ellas no habrá una conexión exitosa.

Hibernate necesita saber cómo leer y almacenar objetos de una clase persistente. Se muestra el archivo donde Hibernate hace el mapeo. El mapeo indica a Hibérnate qué tabla en la base de datos tiene que ser accedida, y qué columnas en dicha tabla deben usarse.

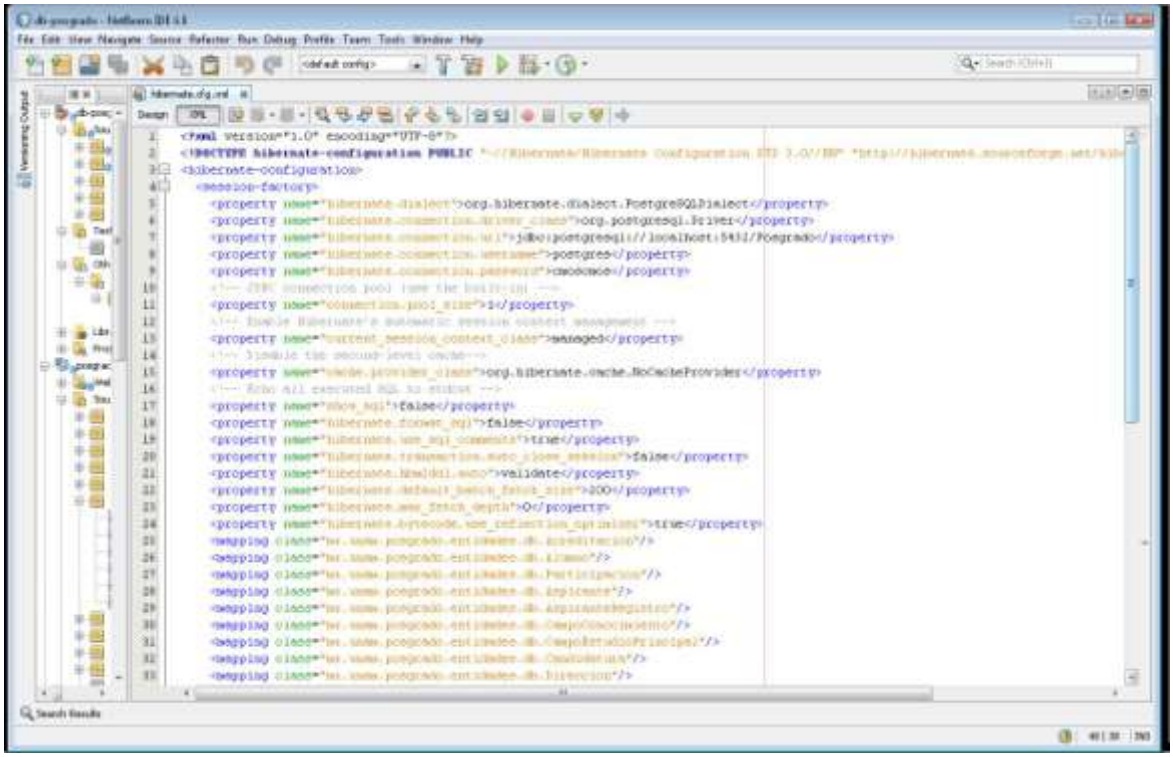


Figura 7. Mapeo

Capitulo 5 Desarrollo

Debido a la gran diversidad de información que se tiene dentro del programa de posgrado de Ciencias Médicas, Odontológicas y de la Salud, es necesaria la automatización a través de una aplicación Web que administre una base de datos con todos los datos del aspirante, alumno así como de tutores dentro del programa.

El objetivo general es la implementación de un sistema que facilite a los administradores y capturistas el registro de alumnos, aspirantes y tutores que se encuentran dentro del programa de posgrado a fin de mantener los datos íntegros y almacenados correctamente, para evitar redundancia de datos y tener un control de las personas que tienen acceso a esta información a través de la administración del sistema.

La realización de todo el desarrollo del sistema se llevo a cabo en 8 meses, se siguió el esquema de los módulos junto con la creación de la base datos. La realización se hizo gracias al proceso de Ingeniería de Software.

El primer paso que se llevo a cabo fue la migración de datos ya que todo se tenía en formato Excel y casa uno de ellos tenían que validarse ya que los datos tenían errores ortográficos, se repetían por lo que eran muy redundante la información.

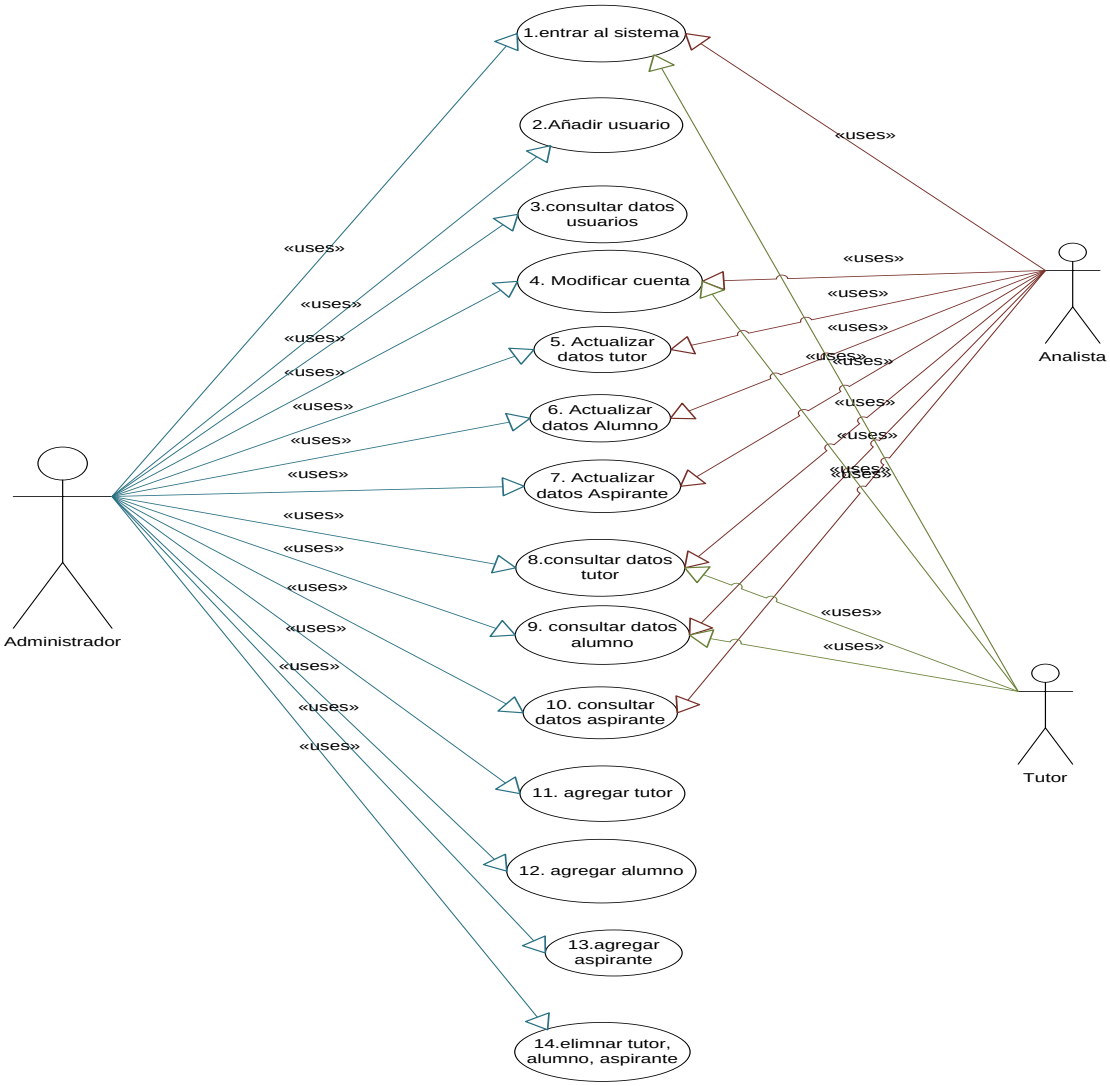
Elabore el esquema de la base de datos de acuerdo a las especificaciones que se hicieron con los encargados del proceso de aspirantes-alumnos todo esto para tener una base de datos eficiente, útil y competente para toda la información robusta que está por manejarse.

El desarrollo del código fue estructurado ya que se iban haciendo las vistas junto con los que van hacer los encargados de usar el sistema. Al realizarlo con ellos tendría unas vistas 100 % manejables por los usuarios y así poder evitar tener vistas inservibles.

Teniendo en cuenta que las pruebas se realizaban en un solo equipo de cómputo, es necesario realizarlo en un servidor que tenga la capacidad de alojamiento de una base de datos robusta y que con el paso del tiempo crecerá. El servidor contara con un procesador xenón de 4 núcleos, memoria RAM de 8 G, un dos discos duro de 1 Tb, uno de los discos duro será para respaldar la información (replicación).

5.1 Especificación de requerimientos

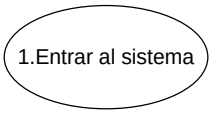
De acuerdo a lo especificado se elaboró un diagrama de casos de uso general que servirá de base para analizar, validar, administrar y generar los requerimientos.



Requerimientos funcionales

A continuación se detallan algunos casos de uso para aclarar la funcionalidad de cada uno y poder hacer el diseño del sistema.

- **Caso de Uso: 1, Entrar al sistema**



Actor: Usuario (Administrador, analista o tutor)
Descripción: El usuario inicia sesión para entrar al menú principal
Precondición: El usuario debe estar registrado

Flujo

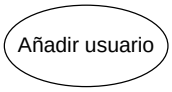
Actor		Sistema		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	**El usuario accede al sistema mediante un navegador**	**2**	**El sistema muestra la pantalla de inicio de sesión para que ingresen los datos**	
3	**El usuario ingresa su nombre de usuario y contraseña**	**4**	**El sistema valida los datos y manda al usuario a la pantalla de menú principal**	**E1**

Excepciones

ID	Nombre	Acción
E1	**Datos inválidos**	**Se muestra un mensaje para que el usuario corrija sus datos y vuelva a ingresarlos**

Pos condición: El sistema muestra la pantalla de bienvenida con menú principal

- **Caso de Uso: 2, Añadir usuario**



Actor: Administrador
Descripción: El administrador puede añadir más usuarios
Precondición: El usuario debe tener el rol de administrador

Flujo

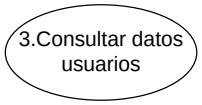
Actor		Sistema		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	**El usuario debe elegir la opción de añadir usuario**	**2**	**El sistema muestra al usuario el formulario con los datos requeridos para registrar un nuevo usuario**	
3	**El usuario llena el formulario**	**4**	**El sistema valida los datos y los guarda en la base de datos**	**E1**

Excepciones

ID	Nombre	Acción
E1	**Datos inválidos**	**Se muestra un mensaje para que el usuario corrija los datos que ingreso**

Pos condiciones: El sistema regresa al menú principal y se ha agregado un usuario

- **Caso de Uso: 3, Consultar datos Usuarios**



Actor: Administrador
Descripción: El administrador puede consultar los datos personales de los usuarios
Precondición: El usuario debe tener el rol de administrador

Flujo

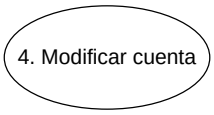
Actor		Sistema		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	**El usuario elige la opción de ver usuarios**	**2**	**El sistema muestra una lista con todos los usuarios registrados**	
3	**El usuario selecciona al usuario del que quiere saber sus datos de registro**	**4**	**El sistema muestra una pantalla con los datos que se registraron en el sistema cuando se dio de alta al usuario**	

Excepciones

ID	Nombre	Acción

Pos condición: El usuario regresa a la pantalla donde está la lista de usuarios

- **Caso de Uso: 4, Modificar cuenta**



Actor: Usuario (Administrador, analista o tutor)
Descripción: El usuario puede modificar sus datos de registro, si es administrados puede modificar otros usuarios
Precondición: El usuario debe estar registrado

Flujo

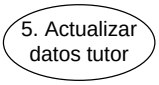
Actor		Sistema		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	**El usuario selecciona la opción modificar cuenta**	**2**	**El sistema muestra un formulario con los datos del usuario que están registrados para que los pueda modificar**	
3	**El usuario modifica los datos que registro**	**4**	**El sistema valida los datos y si son correctos los actualiza en la BD**	**E1**

Excepciones

ID	Nombre	Acción
E1	**Datos inválidos**	**Se muestra un mensaje para que el usuario corrija sus datos y vuelva a ingresarlos**

Pos condición: El usuario regresa al menú principal

Caso de Uso: 5, Actualizar datos tutor



Actor: Usuario (Administrador, analista)
Descripción: El usuario puede modificar los datos con los que se dio de alta un tutor
Precondición: El usuario debe ser administrador o analista, los datos de tutor

Flujo

Actor		Sistema		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	**El usuario elige la opción tutor**	**2**	**El sistema muestra la pantalla donde el usuario puede buscar al tutor que quiere actualizar**	
3	**El usuario busca al tutor que quiere modificar**	**4**	**El sistema muestra la lista de tutores que coinciden con la búsqueda**	
5	**El usuario selecciona la opción actualizar datos en la lista de tutores**	**6**	**El sistema muestra en un formulario los datos registrados del tutor**	

Excepciones

ID	Nombre	Acción
E1	**El usuario no conoce el nombre completo del tutor**	**Se muestran todos los tutores**

Requerimientos no funcionales.

Los requerimientos no funcionales son un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Por lo que se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar.

- **El sistema debe ser web y la administración debe realizarse desde un navegador.**
- **El sistema debe de operar de manera independiente del navegador que se utilice.**
- **El sistema debe tener interfaces graficas de administración y de operación en idioma español y en ambiente 100 % web, para permitir su utilización a través de exploradores o navegadores de internet.**
- **La información de los catálogos y formularios que correspondan a listas de selección deberá ser administrable.**
- **El sistema deberá proveer mecanismos para generar respaldos periódicamente de toda la información que se mantiene en el sistema. Los respaldos deben ser responsabilidad del administrador del sistema quien deberá crearlos y almacenarlos.**
- **Contar con herramientas y medios necesarios para su administración, la realización de búsquedas y la posibilidad de realizar consultas de índole general.**

5.2 Diseño

Uno de los objetivos del diseño es producir un modelo arquitectónico o representación del sistema, el Modelo Vista Controlador me ayudara a generar la arquitectura correcta que separa la lógica de la interfaz de usuario y del almacenamiento de los datos.

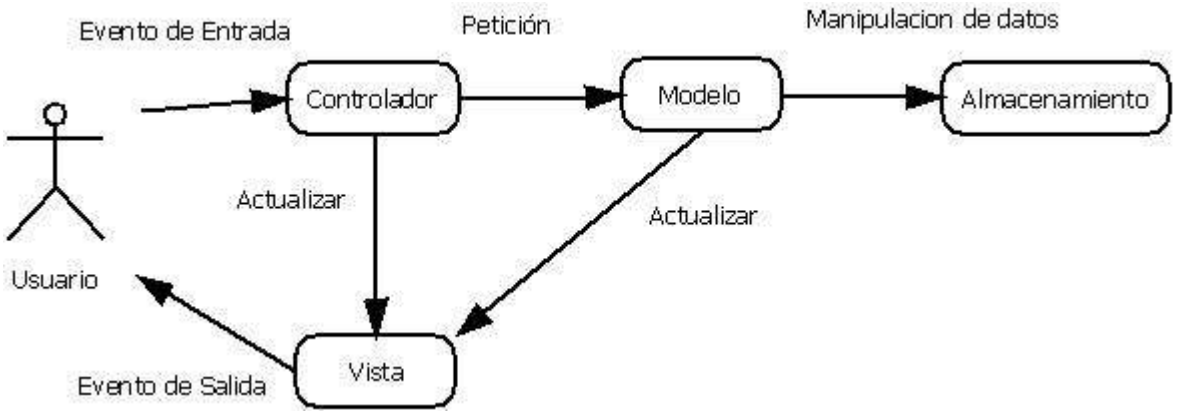


Fig. 4 Modelo Vista Controlador

Los paquetes de Modelo, Vista y Controlador nos ayudarán a organizar elementos en grupos y así poder ver la funcionalidad del sistema.

Consulta de Datos Usuario, Tutor, Alumno, Aspirante.

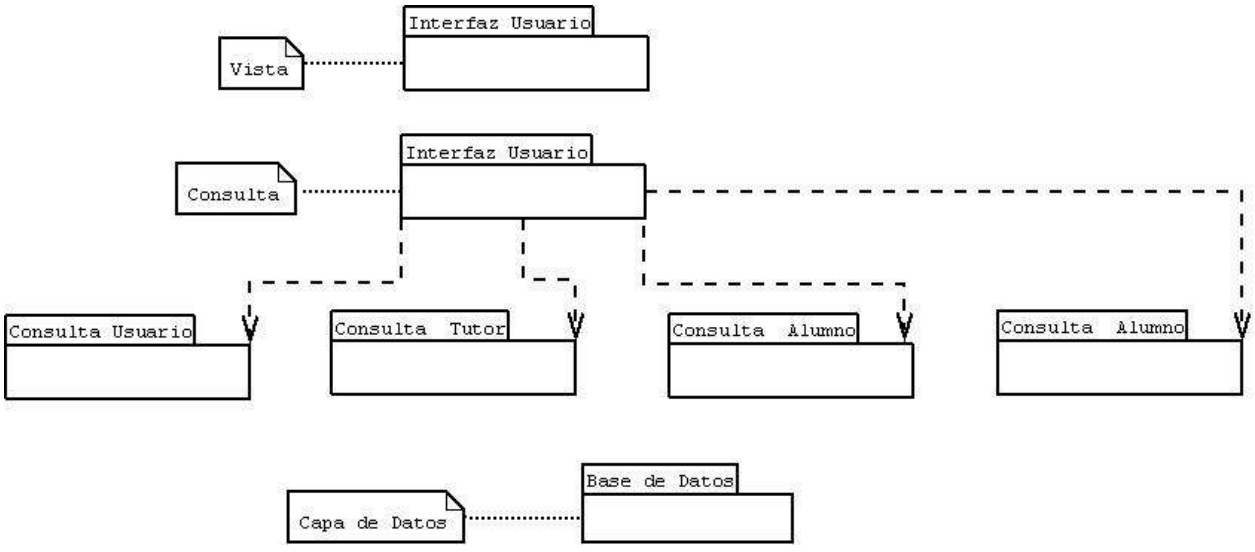


Fig. 5 Consulta de datos

Los usuarios administradores, capturista y tutor podrán consultar datos de acuerdo a los permisos de cada uno que son de lectura, escritura y/o lectura y escritura.

Paquete Actualización Usuario, Tutor, Alumno, Aspirante

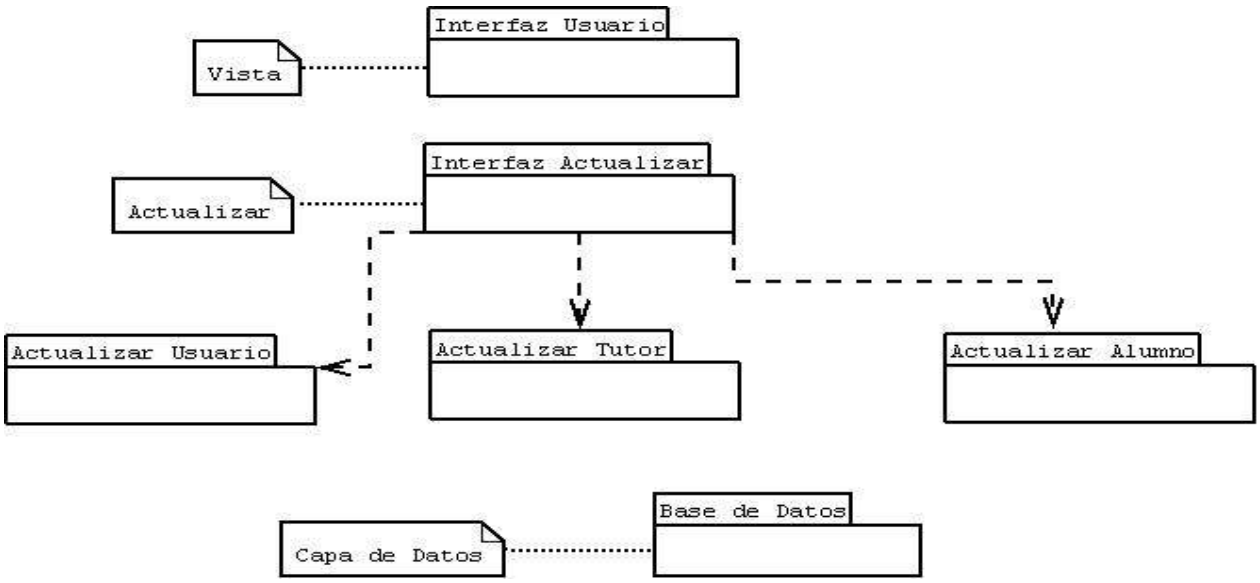


Fig. 6 Actualizar datos

Dependiendo los permisos de usuario se podrá actualizar los datos de usuarios, tutor, alumno y aspirante.

Paquete Agregar Usuario, Tutor, Alumno, Aspirante.

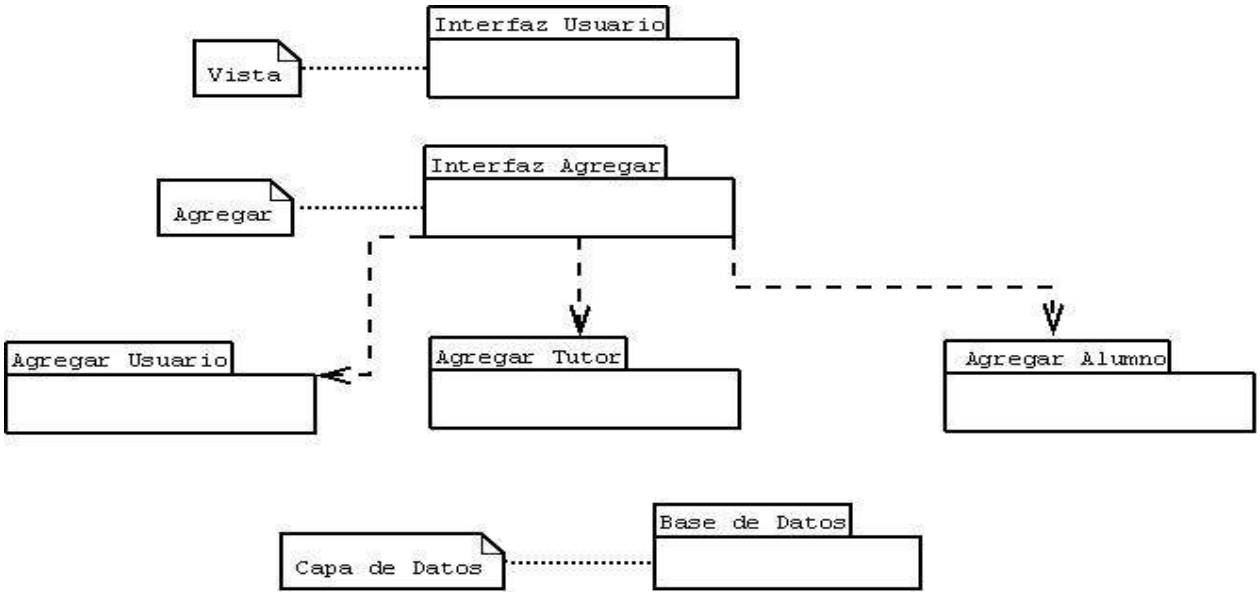


Fig. 7 Agregar Datos

Dependiendo lo permisos se podrá agregar un usuario, tutor y alumno.

Paquete Eliminar Usuario, Tutor, Alumno, Aspirante.

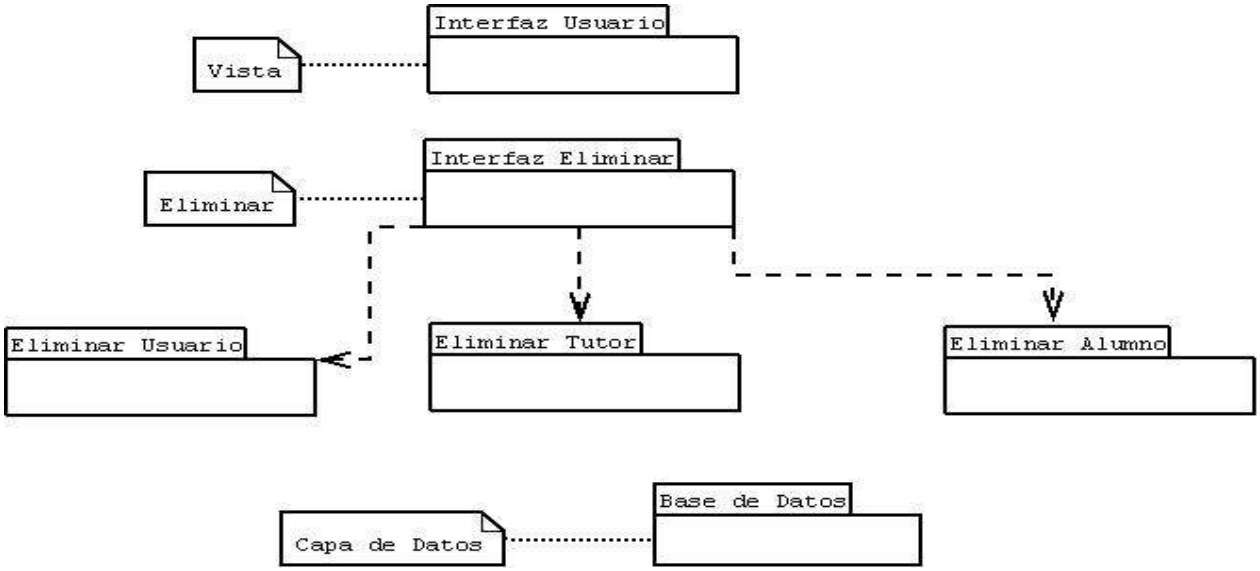
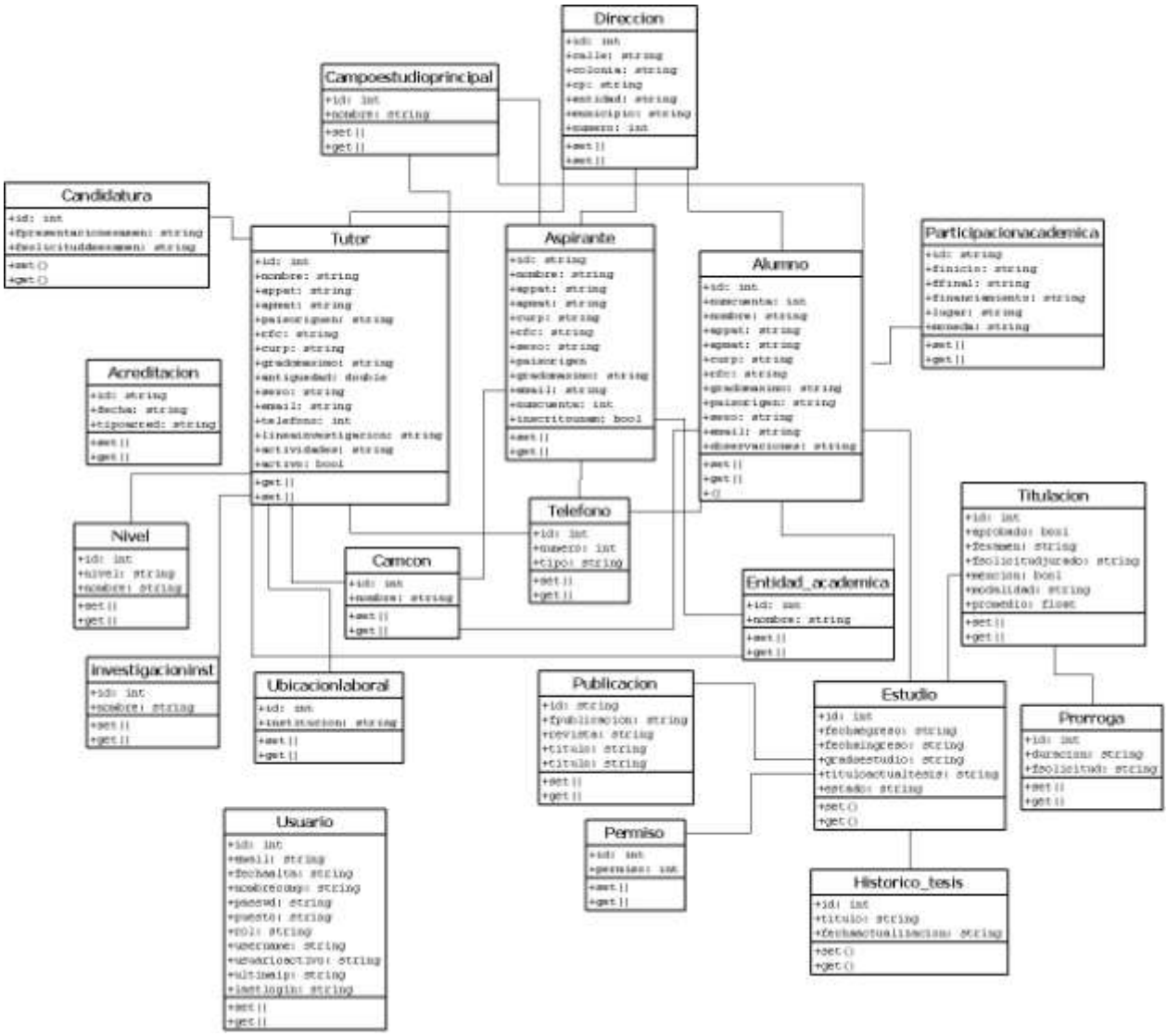


Fig. 8 Eliminar Usuarios

Dependiendo lo permisos se podrá eliminar un usuario, tutor y alumno.

Diagrama de clases



Se muestra un diagrama de clases de la capa lógica de la aplicación para la información de alumnos, tutores y aspirantes al programa de posgrado. Las clases generales tutor, aspirante y alumno pueden tener más de una dirección, teléfono, campo de conocimiento, campo de estudio principal y entidad académica, por lo que no se utilizaron como atributos y si como clases. También se desea guardar lo relacionado a las acreditaciones, ubicación laboral, nivel y candidaturas de un tutor. Un alumno puede tener más de un estudio, ya sea, maestría o doctorado lo que se plantearon como clases, a cada estudio se le relacionara con las publicaciones, prorrogas, permisos y los datos relacionados a la titulación del alumno.

Toda la información de la base de datos está representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas. Por ejemplo, la tabla Tutor contiene campos que representan la información que se posee de los tutores del programa, a su vez se puede observar que está relacionada con las tablas Acreditación y Nivel, la multiplicidad de la relación Tutor Acreditación es uno a uno, porque a pesar de que un tutor se puede re acreditar en repetidas ocasiones no se genera una nueva tupla dentro de la tabla Acreditación sino que se solamente se actualizan los datos correspondientes. Por medio del ID del tutor.

Algunos factores que pueden afectar el desempeño de la base de datos es el volumen de datos que se maneja dentro del sistema, por ejemplo en el PMDCOMS se registran 200 aspirantes cada 6 meses, a los cuales se les pide cierta información que se guarda en la base de datos, esto hace que crezca el tamaño cada 6 meses sin olvidar las actualizaciones que se hace a los alumnos ya registrados.

Registros de volumen de Base de datos

Año	Tamaño
2010	2.0 M
2011	10.0 M

Para poder tener éxito en el diseño de la base datos se tomo en cuenta los conceptos de descomposición sin pérdida y normalizaciones.

Al descomponer las tablas se debe de tener en cuenta que deberá usarse proyecciones que al reunir las tablas resultantes, se puede obtener la tabla original. Para preservar toda la información en la relación original usamos descomposición sin perdida.

Una descomposición {R1, R2,…, Rn} de una relación R es llamada descomposición sin pérdida para R si la reunión natural de R1, R2,…Rn produce exactamente la relación R.

No todas las descomposiciones son sin perdida, porque existen proyecciones que al reunirse no se obtiene exactamente la relación original. Por ejemplo.

Usuario

Usuario	puesto	rol
Mauricio	Tutor	Líder
Mauricio	Profesor	Analista
Mauricio	Tutor	Analista
Raúl	Tutor	Analista

La tabla muestra que puesto tiene cada usuario y que rol va tener al registrarse al sistema.

Al descomponer la tabla por proyección en dos tablas se obtiene lo siguiente:

Tabla 1	
Usuario	**puesto**
Mauricio	Tutor
Mauricio	Profesor
Raúl	Tutor

Tabla 2	
puesto	**rol**
Tutor	Líder
Profesor	Analista
Tutor	Analista

Sin embargo, cuando se reúnen las 2 tablas, se obtiene tuplas adicionales que no aparecían en la tabla original.

Usuario	puesto	rol
Mauricio	Tutor	Líder
Mauricio	Tutor	Analista
Mauricio	Profesor	Analista
Raúl	Tutor	Líder (espúrea)
Raúl	Tutor	Analista

Como podemos ver tenemos una tupla falsa (espúrea) creada por la proyección y reunión. Para poder hacer correctamente la descomposición sin pérdida hay que verificar que para cada par de relaciones que se reúne al conjunto de atributos comunes sea en una llave de una de las relaciones. Esto se hizo colocando atributos funcionalmente dependientes en una relación con sus determinantes y manteniendo las determinantes por sí mismas en una relación.

Para el proceso de normalización se siguió la 1ra, 2da y 3ra forma normal en todas las tablas, de tal forma que la estructura de la base de datos quedara óptima para su implementación, gestión y explotación desde un punto de vista lógico.

La 3FN se aplico a los atributos que no formaron parte de la clave primaria, pero hay que tener en cuenta que para esos atributos se validan dos cosas.

1) **Que sean mutuamente independientes.**

Un conjunto de atributos es mutuamente independiente si ninguno de ellos depende funcionalmente de cualquier combinación de los otros, esto es que cada atributo puede ser actualizado independientemente del resto.

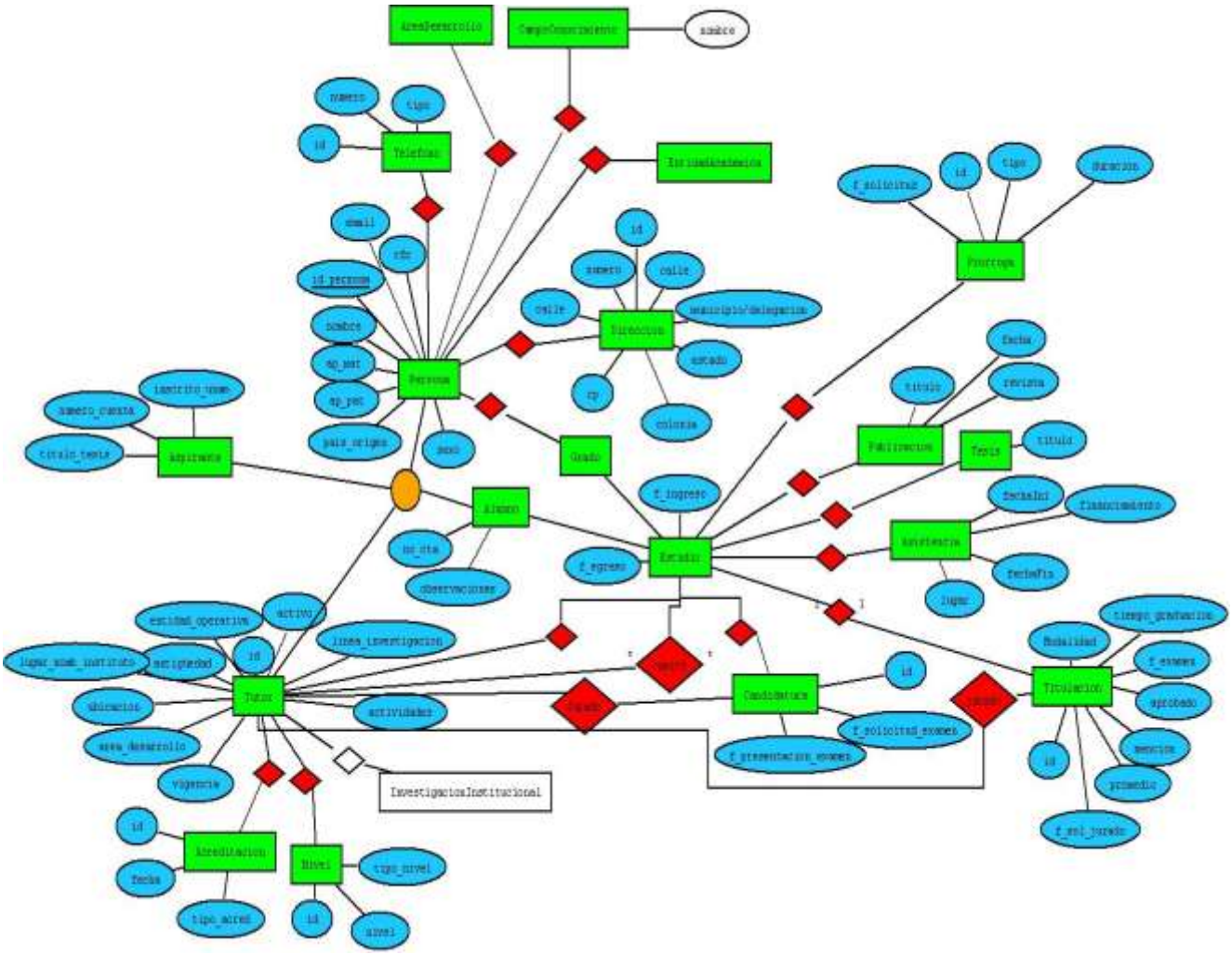
2) **Que sean dependientes irreductiblemente sobre la clave primaria.**

Cualquier atributo es dependiente de la clave primaria pero sin transitividad.

Como por ejemplo

Id ⟶ RFC y RFC ⟶ edad En este caso edad depende de id pero transitivamente y esto no debe de suceder.

Diseño de la Base de Datos



Toda la información de la base de datos está representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas. Por ejemplo, la tabla Tutor contiene campos que representan la información que se posee de los tutores del programa, a su vez se puede observar que está relacionada con las tablas Acreditación y Nivel, la multiplicidad de la relación Tutor Acreditación es uno a uno, porque a pesar de que un tutor se puede re acreditar en repetidas ocasiones no se genera una nueva tupla dentro de la tabla Acreditación sino que se solamente se actualizan los datos correspondientes. Por medio del ID del tutor.

5.3 Construcción

Para la construcción del software del sistema fue necesario revisar los modelos creados en la fase de diseño (casos de uso, diagrama de clases, paquetes, etc.) y contar con los componentes necesarios para la prueba del mismo.

Las herramientas para la creación del código son Tapestry 5, Hibernate, Spring, Ajax, y posrgresql, cada una se utilizo de acuerdo a lo siguiente:

Tapestry 5 al ser un código abierto para la creación de aplicaciones web dinámicas, robustas altamente escalables en Java, permitió crear todo el código ordenadamente ya que asocia una plantilla xhtml (xml + html) y una clase java donde ambas deben de tener el mismo nombre. El objetivo de la clase java es recibir los eventos relativos a la página y el objetivo de la platilla es generar un markup (xml), este markup tiene una extensión .tml y cuando se genera presenta cada componente de la plantilla mostrando en una página todo el código correspondiente.

Algunas otras características importantes que ayudaron a la formación del código fueron que en cada invocación recargaba las clases modificadas sin necesidad de reiniciar el sistema o recargarlo por completo esto ahorro tiempo en el desarrollo. Al tener buena integración con Ajax permite respuestas parciales en las páginas y componentes, manejando datos en formato JSON y así elaborar con mejor calidad las páginas. Ajax elimina la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor, el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

Para la creación de la base de datos, postgresql juega un papel muy importante puesto que es el encargado degestionarla. Al tener la base de datos cargada con todos los datos de tutores, alumnos y aspirantes debemos poder encontrar la manera de manipular todos esos datos y Tapestry al contar con la integración con hibérnate permite realizar métodos eficaces de consulta manejando un esquema de orientación a objetos para accesos a la base de datos y permite a Tapestry realizar más fácilmente las tareas de la interfaz grafica.

A continuación explicare el funcionamiento de todas las herramientas con un ejemplo del sistema:

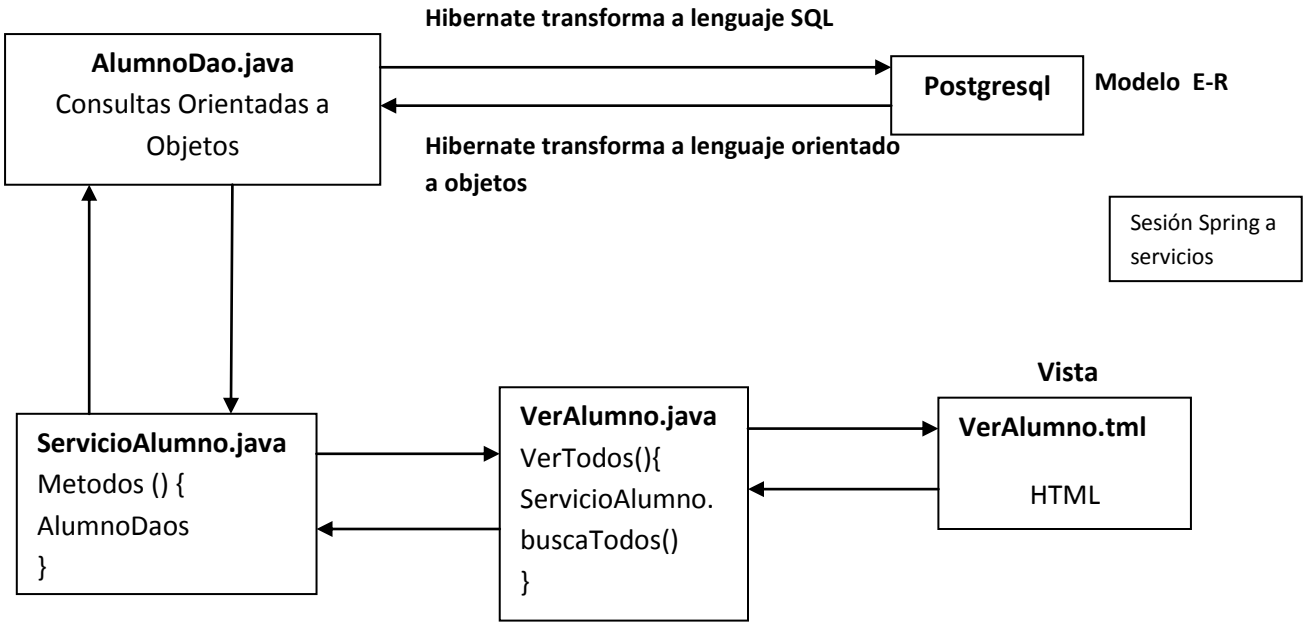
AlumnoDao.java: En este servicio se encuentran los métodos de consulta las cuales son objetos que transforma Hibernate a lenguaje SQL donde se van a la base de datos y se procesa la consulta regresándola en modelo entidad-relación y donde nuevamente Hibernate transforma el modelo entidad-relación en objeto. Aquí se tiene siempre la sesión con Hibernate.

ServicioAlumno.java: Este servicio hace referencia a AlumnoDao.java donde verifica nuevamente los métodos que AlumnoDao.java está llamando, también se encuentran las peticiones que se va puede hacer el usuario.

VerAlumno.java: este controlador recibe las peticiones de ServicioAlumno.java. Se encuentran variables temporales de almacenamiento, usados por tapestry en loops o en formularios. También se encuentran los objetos los cuales tienen métodos que van a mostrarse en la vista.

VerAlumno.tml: Esta es la vista en la cual se van a mostrar todos los datos que se solicitaron. Esta vista se encuentra codificada en xhtml la cual tapestry hace la función de transforma a html.

En todo este proceso Spring hace la función de mantener la sesión del usuario haciendo que solo se tenga una sola y así hacer más eficiente al sistema en cuestión de tiempo y memoria.



En la siguiente figura se muestran la estructura general del sistema con que cuenta el sistema y se ilustran los 3 roles de usuarios existentes.

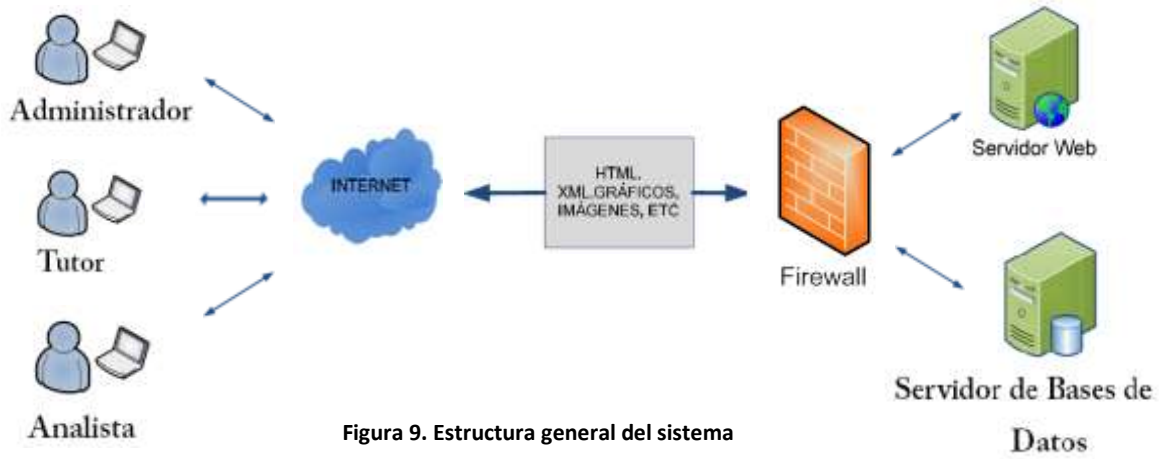


Figura 9. Estructura general del sistema

La base de datos radica en un servidor dentro de una zona segura y ofrecer sus servicios a través de un servidor de aplicaciones con acceso a través de Internet.

Se detallo la composición de versiones de los productos elaborados, ya que en algunos casos las primeras versiones no eran muy exitosas por lo cual se tenían que mejorar. En esta actividad se estudia el alcance de cada versión ya que todo fue planteado por el cliente o usuario.

La siguiente lista muestra los tiempos empleados en cada producto elaborado durante todo el desarrollo del sistema.

Listado		
Producto	**Tiempo dedicado**	**Tamaño estimado**
Requerimientos	**4 semanas**	**14**
Diagrama de clases	**2 semanas**	**21**
Casos de Uso	**2 semanas**	**13**
Paquetes	**3 semanas**	**4**
Diseño de Bases de datos	**5 semanas**	**19 tablas**
Diseño de Interfaces	**8 semanas**	**40**
Codificación	**8 semanas**	**24 .tml y 20 .java**
Prueba de Módulos	**1 semana**	**3**
Prueba de Sistema	**1 semana**	**1**

El tiempo total dedicado para el desarrollo del sistema fue de 34 semanas

5.4 Pruebas

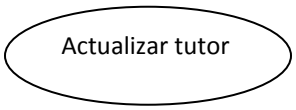
La importancia de las pruebas de integración de software y sus implicaciones con la calidad del software no se debe sobrevalorar por lo cual se especifica las actividades necesarias para la fase.

Al arrancar los servicios del servidor el proceso de ejecución del sistema no mando ningún error de configuración ya que se especificaron cada uno de los servicios.

El proceso de los casos de prueba ayudó a encontrar fallas en el funcionamiento de cada una de las páginas, como por ejemplo:

Actualizar datos tutor

- **Caso de Prueba: 1, Actualizar datos tutor**



Actor: Usuario (Administrador, analista)
Descripción: El usuario busca al tutor a actualizar.
Precondición: El usuario debe estar registrado y tener los permisos de edición.

Flujo

Actor		Sistema		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	**El usuario accede al sistema mediante un navegador**	**2**	**El sistema muestra la pantalla de inicio de sesión para que ingresen los datos**	

3	**El usuario ingresa su nombre de usuario y contraseña**	**4**	**El sistema valida los datos y manda al usuario a la pantalla de menú principal**	**E1**
5	**El usuario hace una búsqueda en Tutores**	**6**	**Se hace la petición mediante la consulta que busca en la base de datos**	**E2**

Excepciones

ID	Nombre	Acción
E1	**Datos inválidos**	**Se muestra un mensaje para que el usuario corrija sus datos y vuelva a ingresarlos**
E3	**Resultados encontrados 0 no hay tutores**	**Se muestra un mensaje avisando que no hay resultados encontrados.**

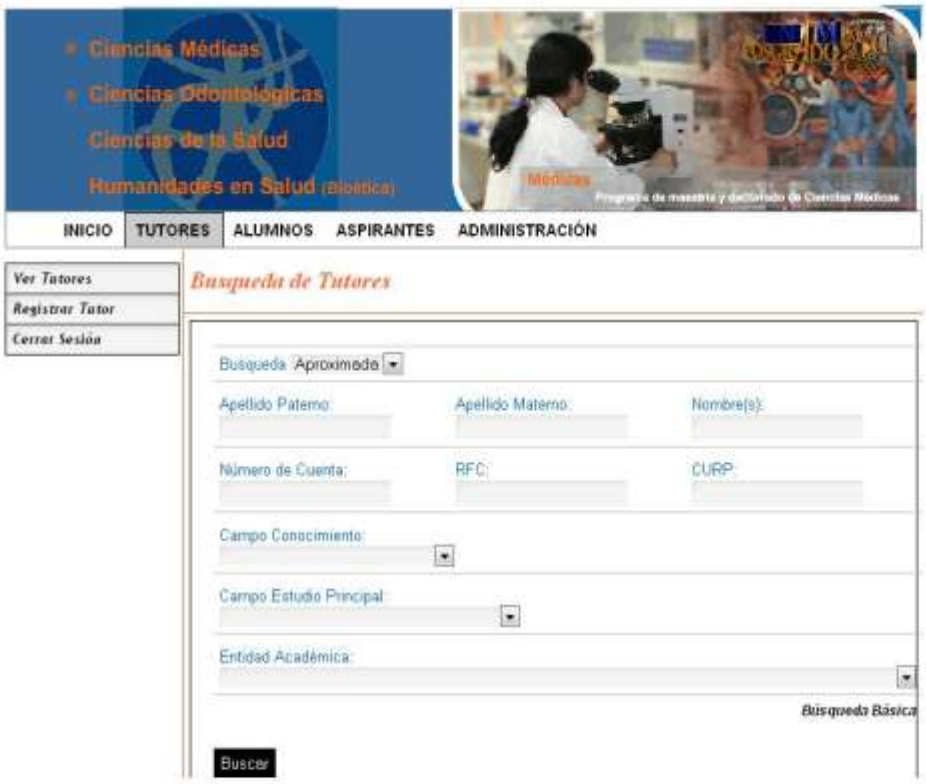
1. Inicio de Sesión

Se muestra la pantalla inicial en la cual nos pedirá el usuario y contraseña

El alerta muestra que el dato de la clave es nulo, quiere decir que esta fuera del dominio de datos validos, por lo tanto es invalido, por lo cual le pide que vuelva a escribir el dato pero correcto. En caso de no saber la contraseña se mostrará una pantalla con las instrucciones para recuperarla.

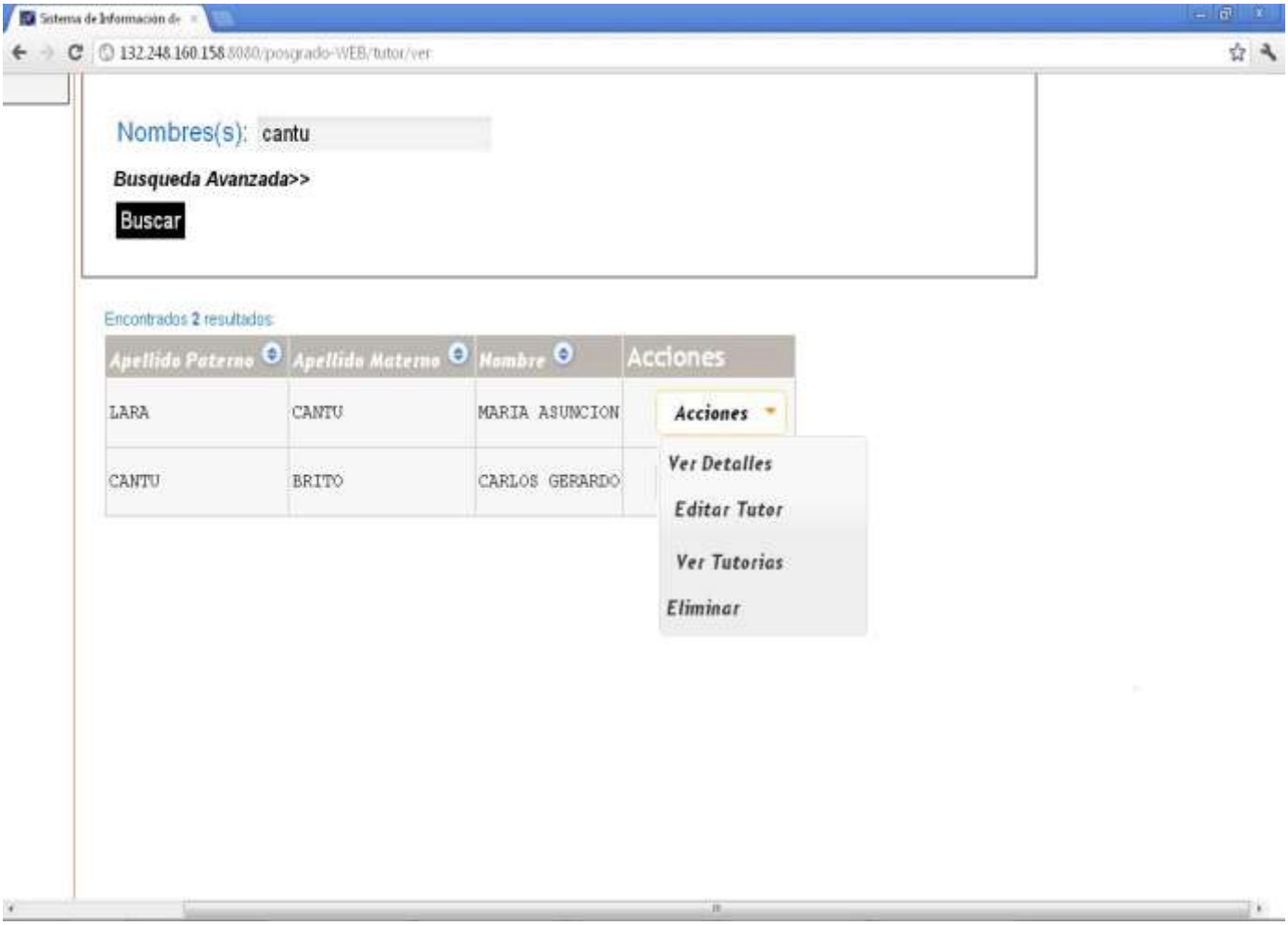
Para poder recuperar la contraseña nos pide los datos del usuario: Usuario y correo electrónico al cual se le enviara su contraseña.

EL usuario puede realizar búsquedas básicas donde solo pone el nombre del tuto. También existe el caso en el cual el usuario quiera realizar búsquedas especificas, para esto se usaran las búsquedas avanzadas.



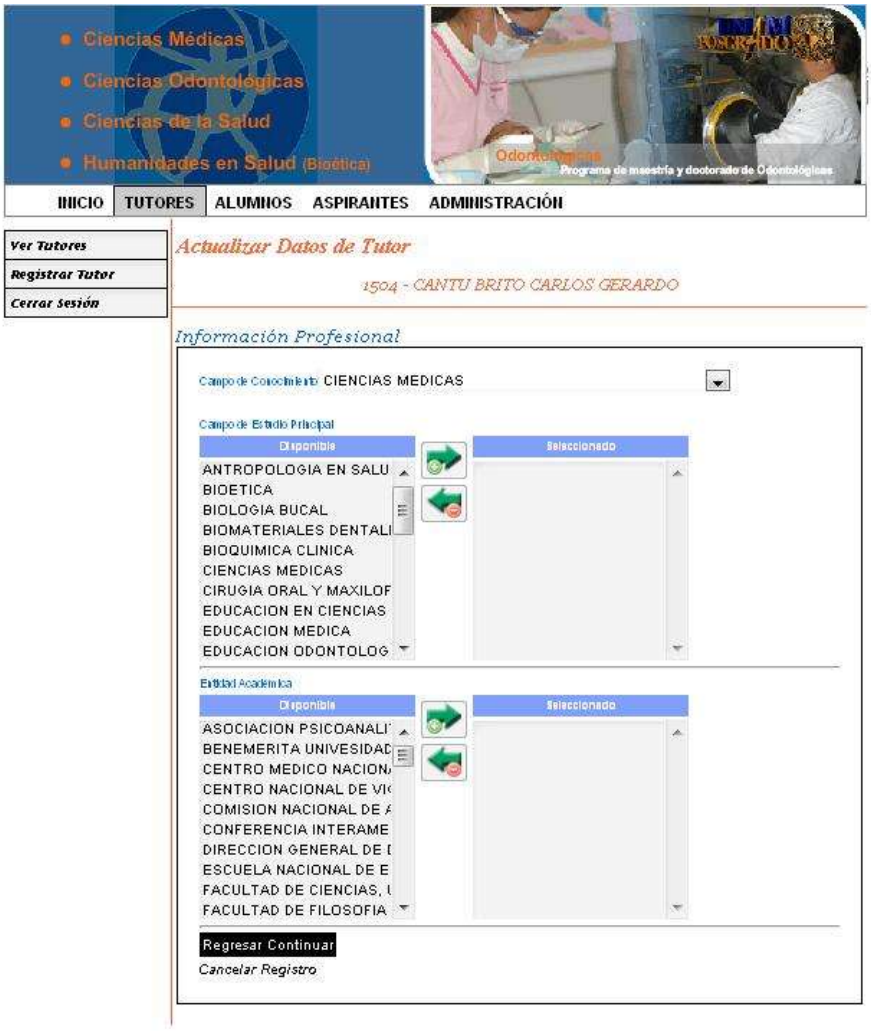
La búsqueda avanzada puede hacerse mediante campos específicos de un tutor como:

- **Apellido Paterno**
- **Apellido Materno**
- **Nombre(s)**
- **Número de Cuenta**
- **RFC**
- **CURP**
- **Campo de Conocimiento**
- **Campo de Estudio Principal**
- **Entidad Académica**
-



Hay que tener en cuenta que esta prueba es con un usuario administrador, el cual puede eliminar cualquier tutor, alumno y aspirante. La opción de acciones nos permite realizar diferentes procesos: Ver Detalles, Editar Tutor, Ver Tutorias y Eliminar. En este caso se hará la prueba con Editar Tutor el cual nos permite actualizar los datos del tutor.

Los datos generales del tutor son parte de la actualización los cuales constan de: Nombre, Apellido Paterno, Apellido Materno, Grado, CURP, RFC, Grado Máximo de Estudio, País de Origen, Actividades, Activo, Antigüedad, Línea de Investigación, Lugar Nombre Inst, Ubicación, Vigencia.



La información Profesional de cada tutor se compone por campo de conocimiento, campo de estudio principal y entidad académica a la que pertenece.

Al ingresar a Contacto del tutor podremos guardar su dirección: Calle, Número, Colonia, Código Postal, Entidad, Municipio/Delegación, el correo electrónico y teléfonos: casa, oficina, celular, fax.

Al terminar de actualizar los datos del Tutor, se guarda y se envía una vista de actualización satisfactoria.

El objetivo del diseño de casos de prueba es derivar un conjunto de pruebas que tenga la mayor posibilidad de descubrir los defectos en el software desarrollado.

Al realizar este caso de pruebas (Actualizar Tutor) se encontró que la conexión con el servidor Web fue nula por lo cual me di cuenta que la configuración no estaba correcta. Un error muy común era que las consultas que se hacían no regresaban los datos correctos ya que dicha consulta se hacía sobre objetos que no existían.

En general las pruebas de integración de módulos y validaciones del sistema se centraron en la verificación funcional de cada módulo. Las pruebas de validación demostraron que los requerimientos del software fueron bien definidos.

En este caso en todas las pruebas que se hicieron al sistema (casos de prueba e integración) se encontraron errores. Gracias al diseño de caminos de manejo de errores se logro llevar a cabo una serie de pruebas que simularon la presencia de datos (código, consultas, base de datos) en mal estado o de otros posibles errores en la interfaz del software.

5.5 Alcance del Sistema

Al desarrollar el sistema se verifico que se contara con las instalaciones necesarias para la ejecución de las distintas fases que se iban a elaborar. Se pidió que se tuviera una persona experta en todo el proceso de trabajo que se hace en el Posgrado de Ciencias Médicas, Odontológicas y de Ciencias de la Salud ya que su participación me iba ayudar a entender los requerimientos que se habían establecido.

El alcance que se tuvo al desarrollar este sistema Web fue:

- **Almacenamiento de información que se tenía desde 1998 hasta el 2008 de alumnos y tutores.**
- **Registro de Aspirantes desde el 2009 .**
- **Consulta de información de alumnos del semestre 1998-2 a 2012-2.**
- **Consulta de tutores acreditados desde los el año 2000 al 2012.**

Al contar con todos estos beneficios se puede manejar mejor la información de los alumnos y tutores que se encuentran dentro del Posgrado de Ciencias Médicas, Odontológicas y de Ciencias de la Salud ya que antes se contaba con una gran pérdida de información que se tenía en papel así como de el tiempo excesivo que se requería para realizar los diferentes procesos antes mencionados.

El sistema está creado para poder agregar más módulos que pueda necesitar el Posgrado de Ciencias Médicas, Odontológicas y de Ciencias de la Salud.

• Conclusión

De acuerdo a los reportes que se tenían antes de desarrollar el sistema el proceso de inscripción de un aspirante, este se tenía que realizar físicamente el proceso en las oficinas de la Coordinación de Posgrado, ahora la realización de este trámite es vía WEB, logrando obtener su historial desde que son aspirantes hasta que se gradúan, lo cual conlleva un ahorro de tiempo y trabajo para el programa de maestría y doctorado.

Al desarrollar el sistema se logro que el proceso de operación de la información del Programa de Posgrado de Ciencias Médicas, Odontológicas y de la Salud (PCMOS) se redujera al reunir toda la información generada del 1998 al 2012 de alumnos, tutores y aspirantes en una base de dato dando la opción de poder consultar, almacenar, editar y eliminar datos del programa (dependiendo de los permisos de cada usuario) con una interfaz grafica amigable.

Las tecnologías empleadas para la elaboración del sistema son capaces de lograr la estabilidad del mismo, así como lograr que cualquier programador pueda utilizar de nuevo el código, inclusive extenderlo para el beneficio del PCMOS. El sistema se encuentra operando desde el mes de febrero del 2010 hasta la fecha con buenos resultados.

BIBLIOGRAFIA

- PRESSMAN, Roger S. Ingeniería del Software, Un enfoque pragmático. McGraw-Hill. 1999. Pp. 618.

- LUCAS GÓMEZ, Ángel, ROMERO GARCÍA, Paloma, FRAILE DOTES, María Victoria, ARGENTE DEL CASTILLO, José y ANTONIO ALFARO, Antonio. Diseño y Gestión de Sistemas de Bases de Datos. Paraninfo. 1993. pp. 472.

- BRAUDE, Eric J. Ingeniería de Software, Una perspectiva orientada a objetos. Alfaomega. 2003. pp.539.

- IBARGÜENGOITIA GONZÁLEZ, Guadalupe, OKTABA, Hanna. Ingeniería de Software Pragmática. Facultad de Ciencias UNAM. 2006. pp. 89.

- LEWIS SHIP, Howard M. Tapestry in Action. Manning. 2009. pp. 553.

- GARCÍA MOLINA, Héctor, ULLMAN, Jeffrey D., WIDOM, Jennifer. DATA BASE SYSTEM: The Complete Book. Prentice Hall. 2002.1119.

- SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S. Data Base System Concepts. McGraw-Hill. 2002. Cuarta edición. pp. 787.

- WHITTEN, Bentley. ANALISIS DE SISTEMAS DISEÑOS Y METODOS. McGraw-Hill. 2008. pp. 569.

- BAUER, Christian, KING, Gavin. Hibernate In Action. HANNING. 2004. pp. 408.

- BAUER, Christian, KING, Gavin. Java Persistente whith Hibernate. HANNING. 2006. pp. 880.

- HAROLD, Kerzner. Project Management a Systems approach to planning, scheduling and controlling. Hardcover. 2009. pp. 1120