UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de Administración del Laboratorio de Microsoft

TESIS

Que para obtener el título de

Ingeniero en computación

P R E S E N T A

José Miguel Díaz López

DIRECTOR DE TESIS

Ing. Jorge Alberto Solano Gálvez

Ciudad Universitaria, Cd. Mx., 2019

Índice de General

1 Introducción

La evolución de la computación ha dado origen a diversos medios para acceder a la información siendo este punto clave tanto en el sector público como en el sector privado.

Para el manejo de la información, aun en nuestros días, se realiza y se almacena en papel o documentos electrónicos, restringiendo el uso y acceso solo a las personas que lo generaron o a las que se les proporciono físicamente o en medios electrónicos, dicho proceso coloca a la información insegura, inconsistente y sin validez.

La necesidad de manipular la información de una forma segura, verídica y correcta, pero sobre todo acceder desde diferentes sitios por varias personas, sin que haya cambio en la integridad de ésta, hace necesario la creación de un sistema

La creación de un medio o sistema de información es un aspecto indispensable y es por ello que se requiere de nuevas técnicas que faciliten la consulta, proceso y manejo de la misma. El internet es el medio de comunicación más utilizado, ya que proporciona información accesible a nivel general y conciso de cualquier tema.

De aquí surge la necesidad de diseñar y desarrollar páginas, sitios o portales web que faciliten la comunicación y despliegue de información de forma fácil y entendible para el usuario, pero sobre todo que la información que se requiere obtener esté organizada, centralizada y sea segura.

El SALM (Sistema de Administración del Laboratorio de Microsoft) fue creado mediante una base de datos que permite al administrador controlar el sistema de préstamos con que cuenta el laboratorio. De igual forma se muestran las clases, cursos, diplomados y proyectos que ofrece el laboratorio, los cuales pueden ser de interés para la comunidad universitaria. También se puede consultar los trabajos de tesis que han sido desarrollados por parte de estudiantes en colaboración con el laboratorio. Ayuda a controlar el inventario de los artículos que se encuentran disponibles, así como el alta y baja de usuarios que tienen derecho a utilizar dichos recursos.

De igual forma el sistema permite que los usuarios puedan consultar desde cualquier dispositivo la lista artículos que tiene disponibles el laboratorio, así como el historial de préstamos de cada uno de ellos.

2 Marco Teórico

2.1. Antecedentes

El laboratorio de Microsoft fue donado por la división Research University Relations de Microsoft y está ubicado en el edificio Valdés Vallejo del conjunto sur de la Facultad de Ingeniería en el campus de Ciudad Universitaria. Es un espacio creado para realizar tareas de investigación y de desarrollo de nuevas tecnologías.

La donación se enmarcó en el Programa Internacional de Microsoft Research University Relations, que ha establecido centros similares en escuelas de Argentina (Universidad de Buenos Aires y Universidad Tecnológica Nacional), Brasil (Universidade de Sao Paulo y Universidade Estudual de Campinas), Chile (Pontificia Universidad Católica de Chile), México (UNAM y Tec de Monterrey) y Puerto Rico (Universidad de Puerto Rico Mayagüez).

En este tipo de convenios que se tiene con Microsoft, destacan tres elementos claves: primero, Microsoft comparte ideas y colabora para resolver los retos más significativos de la computación en nuestro tiempo, esa es la misión de los laboratorios Microsoft Research de Cambridge, Beijín y Redmond con más de 700 investigadores que trabajan sobre 55 diferentes áreas; segundo, la empresa trabaja con la comunidad académica suministrándole recursos para superar las fronteras de la investigación y la escolaridad; y tercero, el éxito de la industria informática depende de la talentosa gente joven que se gradúa cada año de las universidades.

En poco tiempo, estas colaboraciones se extendieron a escuelas de otros países de Europa, China e India. Este proyecto llega a América Latina, en particular en Brasil, Chile, Argentina y México. El centro no sólo fortalece la investigación que se hace en el departamento de computación de la Facultad de Ingeniería sino también la investigación en cómputo que se hace en la Facultad de Ciencias y en el posgrado de computación de la UNAM. La inauguración de este laboratorio demuestra que la investigación científica en computación no está fuera del alcance de las nuevas generaciones.

Microsoft otorgó becas a ciertos alumnos de la Facultad de Ingeniería que actuaron como estudiantes consultores que se encargaron de mantener operando el lugar y brindar asesoría a alumnos y docentes.

La colaboración con Microsoft es de vital importancia ya que contribuye a poner al alcance de los estudiantes los más avanzados implementos tecnológicos para la investigación.

El laboratorio tiene como objetivo exponer a los estudiantes y académicos las últimas tecnologías Microsoft disponibles, incorporando su aplicación en asignaturas relacionadas con las carreras de la Facultad de Ingeniería, desarrollar investigación, desarrollar aplicaciones, así como organizar cursos sobre tecnologías Microsoft.

2.2. Descripción del problema

El laboratorio de Microsoft proporciona diversos servicios a la comunidad universitaria: se imparten clases, cursos y diplomados. Además, el laboratorio ofrece el servicio de préstamo de software, sistemas operativos, libros y hardware, así como algunas otras herramientas del laboratorio. Sin embargo no se cuenta con un software que administre el servicio y que facilite la organización para el control de inventario, este se lleva a cabo en una hoja de Excel, donde se le solicita al usuario proporciona el número de cuenta y nombre de la carrera a la que pertenece indicándole la fecha en que debe de regresar el articulo solicitado, lo que genera un grave problema, ya que los artículos prestados no siempre son devueltos, ocasionando pérdidas al laboratorio.

2.3. Justificación

Si bien el servicio universitario proporcionado actualmente es una actividad que permite tanto al alumno como al profesor tener acceso al uso de tecnología de la información y comunicación de Microsoft para desarrollar distintas actividades, es necesario incluir en dichas estrategias una vinculación más dinámica y eficaz de estas tecnologías al trabajo y al desarrollo docente.

Tomando en cuenta el problema expuesto, se crea esta propuesta para mejorar la administración y organización de los recursos del laboratorio de Microsoft para el apoyo directo en la capacitación y desarrollo tanto del alumno y/o docente de la Facultad de Ingeniería, como del personal que labora en dicho laboratorio.

2.4. Objetivo General

Desarrollar un sistema capaz de administrar el servicio de préstamos de software, sistemas operativos, libros de computación, hardware, así como algunas otras herramientas, llevar el inventario de todos estos recursos, proporcionando mayor integridad en los datos dentro de un sistema centralizado. Lo anterior con el fin de automatizar el sistema con el que se trabaja actualmente y llevar un adecuado control de los recursos con los que cuenta el laboratorio.

2.5. Objetivos Específicos

1. Crear una base de datos con la información de los usuarios que soliciten el servicio, así como el inventario de todo el material que se tenga en el laboratorio.
2. Crear un sistema que permita:

    - Registro de usuarios
    - Registro de software, sistema operativo, libros, hardware y herramientas
    - Registro de préstamos
    - Consulta de disponibilidad de materiales
    - Consulta de usuarios con préstamo o adeudos

- Permitir que los usuarios accedan por medio de internet, para poder revisar la existencia y disponibilidad de los recursos del laboratorio, así como revisar el estado actual de sus préstamos y/o adeudos.

2.6. Alcance

Con este proyecto se propone tener un mejor control en cuanto al servicio de préstamos del laboratorio de Microsoft, con el fin de automatizar el sistema que se tiene actualmente.

Para poder realizar lo anterior se espera desarrollar un software que sea amigable para los usuarios y por lo tanto, los servicios de préstamo sean más eficientes y confiables, siempre pensando en sus necesidades particulares y cubriéndolas de la mejor manera. Aprovechar los recursos que se tienen y que éstos no sean un obstáculo para que el sistema sea incorporado en el Laboratorio de Microsoft de la Facultad de Ingeniería de la UNAM.

Este sistema podrá ser consultado desde cualquier computadora o dispositivo móvil por cualquier usuario, facilitando el acceso a la información que el laboratorio de Microsoft ofrece.

3 Ingeniería del Software

La ingeniería de software es la aplicación de un enfoque sistemático disciplinado y cuantificable al desarrollo, operación y mantenimiento de software y el estudio de estos enfoques, es decir, el estudio de las aplicaciones de la ingeniería al software.

La ingeniería de software aplica diferentes normas y métodos que permiten obtener mejores resultados, en cuanto al desarrollo y uso del software, mediante la aplicación correcta de estos procedimientos se puede llegar a cumplir de manera satisfactoria con los objetivos fundamentales los cuales son:

- Mejorar el diseño de aplicaciones o software adaptándose mejor a las necesidades de las organizaciones.
- Promover mayor calidad al desarrollar aplicaciones complejas.
- Establecer tiempos para el desarrollo de proyectos.
- Aumentar la eficiencia de los sistemas al introducir procesos que permitan medir, mediante normas específicas, la calidad del software desarrollado.
- Una mejor organización de equipos de trabajo, en el área de desarrollo y mantenimiento de software.
- Detectar, a través de pruebas, posibles mejoras para un mejor funcionamiento del software desarrollado.

3.1. Metodología

Es la disciplina que indicará que métodos y técnicas hay que usar en cada fase del ciclo de vida de desarrollo del proyecto.

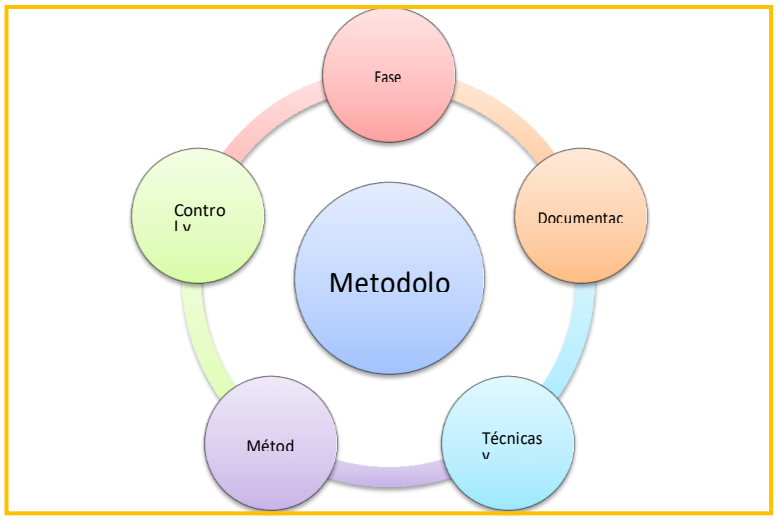


Figura 3.1. Elementos básicos de una metodología.

Los elementos que componen a una metodología son:

1. **Las Fases:** En este punto se marcarán las diferentes actividades que hay que realizar por cada fase.

2. **Los Métodos:** Se tendrá que identificar el modo en el que se realizará el proceso de desarrollo del producto software. Generalmente se suele descomponer los procesos en tareas más pequeñas, en estas tareas se definen los valores que recibirá cada fase así como los que generará y la técnica que se tendrá que usar.

3. **Técnicas y Herramientas**: Indicarán cómo se debería de resolver cada tarea y qué herramientas podríamos usar. Existe diferentes tipos de técnicas, algunas de ellas son:
    - De recopilación de datos: Uso de entrevistas, formularios, etc…
    - Técnicas gráficas: Diagramas, organigrama, diagramas de matrices, etc…
    - Técnicas de modelado: Desarrollos estructurados y orientados a objetos.

4. **Documentación:** Es necesario indicar qué documentación se va a entregar durante todas las fases, esa documentación se debería de realizar de una manera exhaustiva y completa usando todos los valores de entrada y salida que se van generando, esto servirá para recoger los resultados y tomar decisiones de las diferentes situaciones planteadas.

5. **Control y Evaluación:** El control y la evaluación también se debe de realizar a lo largo de todo el ciclo de vida. Consistirá en comprobar y aceptar/denegar todos los resultados que se vayan obteniendo y poder replantear, si es necesario, una nueva planificación de las tareas asignadas, la meta será lograr el objetivo.

3.2. La metodología tradicional

Las metodologías tradicionales son el primer tipo de metodología que surge como guía para garantizar la creación de un producto con un nivel de calidad.

Esta metodología es una disciplina que tiene como base una gestión predictiva, es decir, que parte de unos requisitos iniciales. Con estos requisitos se configurará un plan adecuado usando los recursos y el tiempo necesarios, durante la fase de creación se comprueba si hay desviaciones, si las hay se definen las medidas a tomar y valorar cuáles son las modificaciones que puede experimentar la planificación original.

Por lo tanto, esta metodología define un conjunto de fases secuenciales en las que se indican las operaciones que se van a realizar, el tiempo que van a llevar y cuál será su costo.

Las características más relevantes de esta metodología son:

- Los requisitos son definidos durante todo el proyecto.
- Se basa en los procesos.

- Se supone que el proyecto no va a surgir ningún tipo de cambio por lo tanto no está sujeto a variables.
- Los proyectos suelen estar bien documentados.
- Gestión predictiva.
- El desarrollo se define en fases cuyo conjunto se denomina “ciclo de vida”.
- Documentación exhaustiva de todo el proyecto.
- Se enfocan en obtener el producto en tiempo estimado y con el coste establecido.

3.2.1. Modelo en Cascada

Denominado así por la posición de las fases durante el desarrollo de éstas, que parecen caer en cascada. Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase (Figura. 2.1).

La metodología de desarrollo en cascada es:

1. Análisis de requisitos
2. Diseño del sistema
3. Diseño del programa
4. Codificación
5. Pruebas
6. Implementación del programa
7. Mantenimiento

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo. La palabra cascada sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

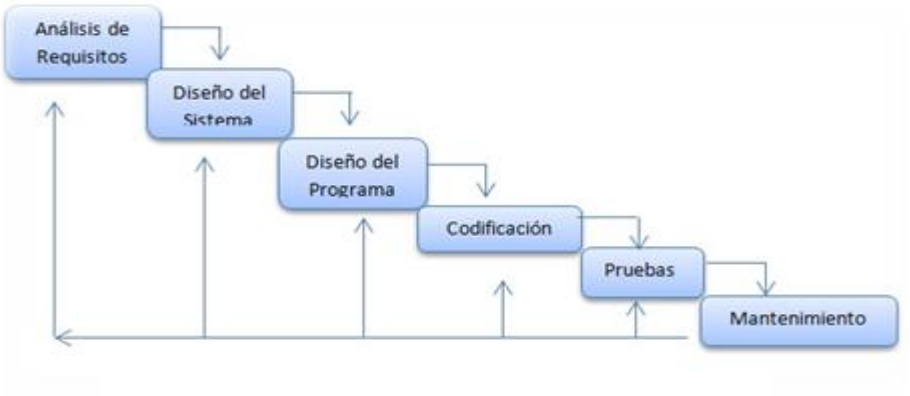


Figura. 3.2 Modelo en Cascada

3.3. Metodología RUP

Otra metodología característica de la metodología tradicional es **RUP (Rational Unified Proces).**

RUP fue desarrollado por *Rational Software* y ahora perteneciente a IBM. Se basa en un marco de procesos de trabajo que pueden ser adaptados por las organizaciones que hagan el desarrollo y por los desarrolladores, seleccionando los elementos más apropiados del proceso.

El proceso Unificado Rational resulta de una combinación de varias metodologías y se vio influenciado por otros métodos como el espiral, fue desarrollado con las mismas técnicas que el equipo de creadores y desarrollo usaba para el diseño del software, usando UML (Unified Modeling Language).

RUP se basa en tres módulos principales que contestan a las preguntas de: quién hace el proceso, qué productos de trabajo se van a realizar, qué documentos y modelos se van a producir y cómo se van a realizar las tareas.

Las fases que forman el ciclo de vida de RUP se dividen en cuatro:

1. **Inicio:** Se establece el objetivo del sistema y se recogen los requisitos del usuario.

2. **Elaboración:** Se busca reducir riesgos y cumplir con la planificación y coste indicado. Se genera una estructura arquitectónica que se puede ejecutar y que servirá de punto de partida para después permitir desarrollar la disciplina de diseñar, implementar y probar.

3. **Fase de construcción:** Partiendo de la arquitectura elaborada en la fase anterior se realizará casi toda la implementación, creando versiones totalmente funcionales para comprobar que satisface las necesidades del usuario.

4. **Transición:** Se comprueba que el software cumple con todas las necesidades y se realizan feedback con el cliente para ajustar el software, dado una de estas fases contiene interacciones necesarias para alcanzar los objetivos del producto y cada fase tiene un objetivo y un hito que indicará que el objetivo se ha alcanzado.

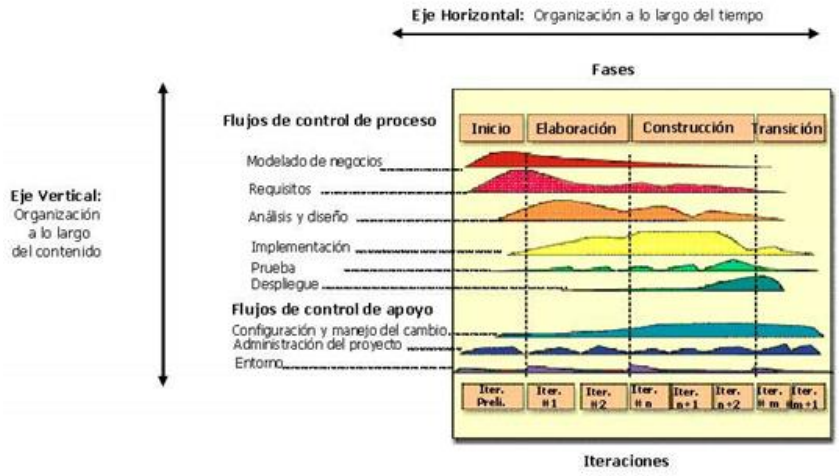


Figura. 3.3 Metodología RUP

Cada una de estas fases se desarrollará mediante un ciclo de interacciones, éstas consisten en hacer un ciclo de vida en cascada reducido, en la que el flujo de trabajo irá variando según la fase en la que se encuentre.

Estas interacciones son llevadas a cabo bajo las disciplinas de:

1. **Disciplina de desarrollo**
   - Requerimientos: Se trasladan las necesidades del negocio a un sistema automatizado.
   - Análisis y diseño: Los requerimientos se trasladan a una arquitectura software.
   - Implementación: Se crea el software adaptándolo a las necesidades.
   - Pruebas: Se comprueba que el software actúa de forma adecuada.

2. **Disciplina de soporte:**
   - Configuración y administración de los cambios.
   - Administración de los horarios y recursos.
   - Ambiente de desarrollo y su administración.

Los roles que se definen en RUP indican las tareas que tiene que hacer cada uno de los miembros del proyecto y el objetivo que se debe de conseguir.

Roles de RUP

DISCIPLINA	ROLES GENERALES	ROLES ESPECÍFICOS
Modelado de negocio	**Analista de procesos de negocio.** Descubrir todos los casos de uso de negocio.	**Diseñador de negocio**. Detallar un conjunto de los casos de uso de negocio.
Requisitos	**Analista de sistemas**. Descubrir todos los casos de uso.	**Especificador de casos de uso**. Detallar un conjunto de los casos de uso.
Análisis y diseño	**Arquitectos de Software**. Toma decisiones tecnológicas de la solución a nivel global.	**Diseñadores**. Detallan el análisis y diseño para un conjunto de casos de uso.
Implementación	**Integrador**. Es el propietario del plan de construcción que muestra cómo se integrarán cada una de las clases, las unas con las otras.	**Desarrollador o programador**. Implementa un conjunto de clases o un conjunto de operaciones de una clase.

Pruebas	**Gestor de las pruebas. Asegura que las pruebas han sido realizadas correctamente.**	
	Analista de pruebas. Selecciona qué se va a probar según lo estimado.	
	Diseñador de pruebas. Decide qué pruebas deberían ser automáticas o manuales, y crea las automáticas.	**Diseñador de pruebas. Implementa las pruebas automáticas de la iteración. Probador. Ejecuta un test específico.**
Despliegue o Implantación	**Gestor de la implantación**. Supervisa la implantación de todas las unidades.	**Artista gráfico, escritor tecnológico y desarrollador de material.** Crean el material necesario para asegurar la correcta implantación.
Gestión del proyecto	**Gestor del proyecto**. Crea los casos de negocio y un plan general, y toma decisiones críticas al respecto de que cosas hacer y cuales no hacer.	**Gestor de proyectos**. Planifica, monitoriza y gestiona los riesgos para una sola iteración.
Entorno	**Ingeniero de procesos**. Es el responsable de los procesos del proyecto.	**Especialista en herramientas**. Crea manuales de uso de herramientas específicas.

Configuración y Mantenimiento	**Gestor de la configuración.** Establece las políticas y planes. **Gestor de control de cambios.** Establece un proceso de control de los cambios.	**Gestor de la configuración**. Crea una unidad de despliegue o implantación, reportes del estado de la configuración, auditorias. **Gestor de control de cambios.** Revisa y gestiona las peticiones de cambios

3.4. Metodologías Ágiles

Las metodologías ágiles surgen como una alternativa a las metodología tradicionales las cuales, tal y como acabamos de ver en los apartados anteriores son demasiado burocráticas y por tanto rígidas para las actuales características del mercado.

Años atrás la evolución de los productos era lenta y se producía siempre en un entorno estable en el que apenas había variaciones.

Hoy en día sin embargo el entorno en el que se mueve el software es demasiado inestable y cambiante por lo que estas metodologías no se adaptan, ya que hay que reducir el tiempo de creación pero sin dejar de todo la calidad del software.

Las metodologías convencionales presentan para este tipo de proyectos los siguientes inconvenientes:

1. Es necesario conocer desde el inicio qué desea el cliente.

2. No se deben de cambiar los requisitos iniciales puesto que a medida que se sigue construyendo el proyecto las modificaciones y la corrección de los errores es más costosa. Además todos los cambios que se produzcan los sufrirá económicamente el cliente.

3. Se establecen mecanismos de control para el proyecto que a veces dan sensación de inflexibilidad a posibles cambios y que de hacerlo incrementaría el coste.

4. Excesiva documentación que a veces incluso no es útil.

5. Quizás este sería el punto más importante, “la lentitud” del desarrollo. Hoy en día para ser competitivos es necesario la agilidad y flexibilidad a la hora de la creación de los productos.

6. Las metodologías tradicionales se encuentran con las dificultades al final del proyecto, lo que termina retrasando las entregas.

Todos estos inconvenientes han hecho que las metodologías clásicas no hayan sido capaces de eliminar los fallos y que haya una explosión de creación de software orientado a la Web, en la que se requieren cambios constantes, y que los tiempos de desarrollo sean más cortos hacen que aparezca el concepto de “metodología ágil” como una alternativa atractiva.

El desarrollo ágil está centrado en la iteración, comunicación y en reducir elementos intermedios.

Además el método de desarrollo ágil fomenta la comunicación entre los miembros del equipo lo que previene problema que en otra metodología quedan escondidos.

Esta comunicación no solo se establece de forma cerrada entre los miembros del equipo sino que también se realiza con la figura que representa al cliente.

El representante del cliente es necesario como elemento de apoyo para los desarrolladores puesto que será la persona a la que se podrán hacer las preguntas necesarias y que junto con el resto de personas involucradas en el negocio comprobarán si se cumplen los objetivos.

Por lo tanto trabajar con una buena comunicación permite que se puedan tomar las decisiones de forma más rápida y aplicarlas.

La característica realmente nueva que aportan estas metodologías es reconocer a las personas como el principal valor para que un proyecto consiga terminarse de forma correcta.

Podemos deducir que las metodologías ágiles a diferencia de las metodologías tradicionales o clásicas son más adecuadas cuando el entorno presenta una cierta incertidumbre o es cambiante.

En este contexto se podrían definir las siguientes ventajas:

1. Gran capacidad de respuesta ante los cambios, los cuales no se entienden como un problema sino como algo necesario para que el producto sea mejor y satisfaga al cliente. Los cambios formarán parte del proceso de desarrollo.
2. Las entregas no se hacen al final sino que se hacen pequeñas entregas. Estas entregas permiten al cliente valorar el producto además de ir trabajando con algunas funcionalidades.
3. Los ciclos cortos de entrega ayudarán a disminuir los riesgos sobre todo al principio del proyecto.
4. Se trabaja en equipo entre el cliente y los desarrolladores mediante una comunicación casi diaria para evitar errores y documentación innecesaria.
5. Eliminar el trabajo que no es necesario y que realmente no aporta un valor al negocio.
6. Buscar la mejor técnica y el mejor diseño para conseguir productos de calidad.
7. Mejorar los procesos y al equipo que realiza el desarrollo.

3.5 Metodologías Ágiles vs. Metodologías Tradicionales

Son varias las ventajas que ofrecen las metodologías ágiles respecto a las tradicionales, permiten una mejor gestión de los recursos humanos e intentan acortar las distintas fases de desarrollo, realizando muchas más pruebas a lo largo de todo el proceso, reduciendo el tiempo y el costo de desarrollo de aplicaciones.

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo de software, con el objetivo de asegurar que el código que se obtenga satisfaga los requerimientos del usuario y reúna estándares aceptables de calidad. El trabajo de planificación es riguroso, aun cuando en la práctica

muchas veces estas planificaciones no se respeten. Por su parte, las metodologías ágiles aportan nuevos métodos de trabajo que apuestan por una cantidad apropiada de procesos, no se desgastan con una excesiva cantidad de cuestiones administrativas (planificación, control, documentación) ni tampoco defienden la postura extremista de total falta de proceso. Ya que se tiene conciencia de que se producirían cambios, lo que se pretende es reducir el costo de rehacer el trabajo.

Como ventaja de las metodologías ágiles se puede decir que son adaptativas más que predictivas, proponen procesos que se adaptan y progresan con el cambio, llegando hasta el punto de cambiar ellos mismos. Una metodología tradicional potencia la planificación detallada y de largo alcance de prácticamente todo el desarrollo de software, en contraste con las metodologías ágiles

Las metodologías ágiles están orientadas más a los desarrolladores que a los procesos de desarrollo. Intentan trabajar con las personas asignadas a un proyecto, más que contra ellos, de tal forma que permiten que las actividades de desarrollo de software se conviertan en una actividad de colaboración grata e interesante.

Los métodos ágiles se fundamentan en la socialización, por medio de la comunicación cara a cara, la colaboración y la programación en parejas, para compartir conocimiento táctico entre los desarrolladores y usuarios, estas metodologías privilegian las personas, la comunicación y la colaboración lo que facilita el compartir conocimiento táctico. Fomenta también una cultura de confianza y motivación, lo que a su vez posibilita la implementación de herramientas para compartir conocimiento explícito.

3.6. Manifiesto Ágil

Los 11 principios del manifiesto ágil son:

1. Satisfacer al cliente a través de la entrega temprana y continua del software de valor.
2. Son bienvenidos los requisitos cambiantes, aun llegando tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.

5. Construcción de proyectos en torno a individuos motivados.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funcione es la principal medida de progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los desarrolladores y usuarios han de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño, mejora la calidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de equipos que se auto organizan.

3.7. Scrum

Es un proceso de ingeniería de software que aplica un conjunto de buenas prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto (Figura. 3.1).

Se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, y la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

También se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costos se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

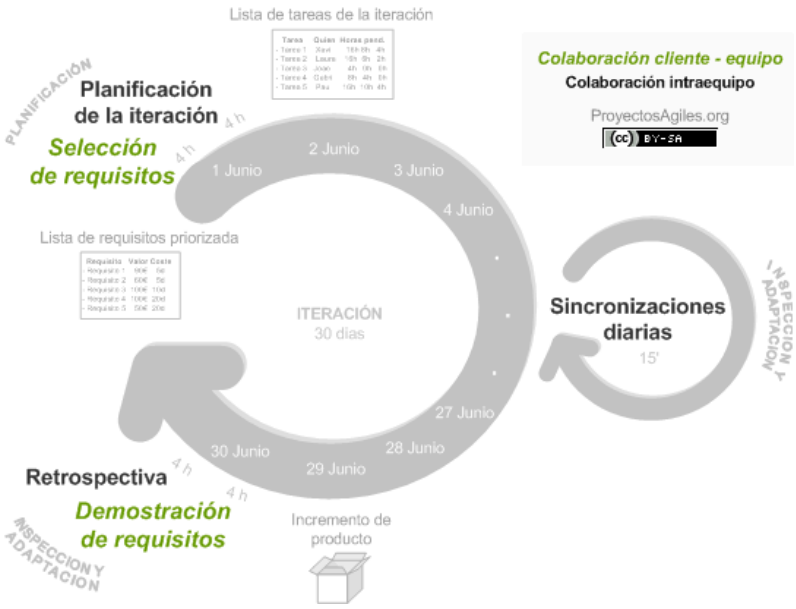


Figura. 3.1 Metodología Scrum

Scrum al ser una metodología de desarrollo ágil tiene como base la idea de creación de ciclos breves para el desarrollo, que comúnmente se llaman *iteraciones* y que en Scrum se llamarán "*Sprints*".

Para entender el ciclo de desarrollo de Scrum es necesario conocer las 5 fases que definen el ciclo de desarrollo ágil:

1. **Concepto:** Se define de forma general las características del producto y se asigna el equipo que se encargará de su desarrollo.

2. **Especulación:** en esta fase se hacen disposiciones con la información obtenida y se establecen los límites que marcarán el desarrollo del producto, tales como costes y agendas.

   Esta fase se repite en cada iteración y consiste, en rasgos generales, en:

   - Desarrollar y revisar los requisitos generales.
   - Mantener la lista de las funcionalidades que se esperan.
   - Plan de entrega. Se establecen las fechas de las versiones, hitos e iteraciones. Medirá el esfuerzo realizado en el proyecto.

3. **Exploración:** Se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.
4. **Exploración:** Se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.
5. **Revisión:** El equipo revisa todo lo que se ha construido y se contrasta con el objetivo deseado.
6. **Cierre:** Se entregará en la fecha acordada una versión del producto deseado. Al tratarse de una versión, el cierre no indica que se ha finalizado el proyecto, sino que seguirá habiendo cambios, denominados "mantenimiento", que hará que el producto final se acerque al producto final deseado.

Se construirá el producto a partir de las ideas principales y se comprueban las partes realizadas y su impacto en el entorno.

3.7.1. Componentes de Scrum.

Para entender todo el proceso de desarrollo del Scrum, se describirá de forma general las fases y los roles. Estas fases y roles se detallarán de forma más concisa más adelante.

Scrum se puede dividir de forma general en 3 fases, que podemos entender como *reuniones*. Las reuniones forman parte de los artefactos de esta metodología junto con los roles y los elementos que lo forman.

3.7.1.1. Las Reuniones.

Planificación del Backlog
Se definirá un documento en el que se reflejarán los requisitos del sistema por prioridades.

En esta fase se definirá también la planificación del Sprint 0, en la que se decidirá cuáles van a ser los objetivos y el trabajo que hay que realizar para esa iteración.

Se obtendrá además en esta reunión un Sprint Backlog, que es la lista de tareas y que es el objetivo más importante del Sprint.

Seguimiento del Sprint

En esta fase se hacen reuniones diarias en las que las 3 preguntas principales para evaluar el avance de las tareas serán:

- ¿Qué trabajo se realizó desde la reunión anterior?
- ¿Qué trabajo se hará hasta una nueva reunión?
- Inconvenientes que han surgido y qué hay que solucionar para poder continuar.

Revisión del Sprint

Cuando se finaliza el Sprint se realizará una revisión del incremento que se ha generado. Se presentarán los resultados finales y una demo o versión, esto ayudará a mejorar el feedback con el cliente.

3.7.1.2. Los Roles

Los roles se dividen en 2 grupos:

Las personas que están comprometidas con el proyecto y el proceso de Scrum.

- **Product Owner:** Es la persona que toma las decisiones, y es la que realmente conoce el negocio del cliente y su visión del producto. Se encarga de escribir las ideas del cliente, las ordena por prioridad y las coloca en el Product Backlog.
- **ScrumMaster:** Es el encargado de comprobar que el modelo y la metodología funciona. Eliminará todos los inconvenientes que hagan que el proceso no fluya e interactuará con el cliente y con los gestores.
- **Equipo De Desarrollo:** suele ser un equipo pequeño de unas 5-9 personas y tienen autoridad para organizar y tomar decisiones para conseguir su objetivo. Está involucrado en la estimación del esfuerzo de las tareas del Backlog.

Aunque no son parte del proceso de Scrum, es necesario que parte de la retroalimentación dé la salida del proceso y así poder revisar y planear cada sprint.

- **Usuarios:** Es el destinatario final del producto.
- **Stakeholders:** Las personas a las que el proyecto les producirá un beneficio. Participan durante las

revisiones del Sprint.
- **Managers:** Toma las decisiones finales participando en la selección de los objetivos y de los requisitos.

3.7.1.3. Elementos de Scrum.

Los elementos que forman a Scrum son:

- **Product Backlog:** lista de necesidades del cliente.
- **Sprint Backlog:** lista de tareas que se realizan en un Sprint.
- Incremento: parte añadida o desarrollada en un Sprint, es un parte terminada y totalmente operativa.2.7.1.4. Product Backlog.

Es el inventario en el que se almacenan todas las funcionalidades o requisitos en forma de lista priorizada. Estos requisitos serán los que tendrá el producto o los que irá adquiriendo en sucesivas iteraciones.

La lista será gestionada y creada por el cliente con la ayuda del Scrum Master, quien indicará el coste estimado para completar un requisito, y además contendrá todo lo que aporte un valor final al producto.

Las tres características principales de esta lista de objetivos serán:

- Contendrá los objetivos del producto, se suele usar para expresarlos las historias de usuario.
- En cada objetivo, se indicará el valor que le da el cliente y el coste estimado; de esta manera, se realiza la lista, priorizando por valor y coste, se basará en el ROI.
- En la lista se tendrán que indicar las posibles iteraciones y los releases que se han indicado al cliente.
- La lista ha de incluir los posibles riesgos e incluir las tareas necesarias para solventarlos.

Es necesario que antes de empezar el primer Sprint se definan cuáles van a ser los objetivos del producto y tener la lista de los requisitos ya definida. No es necesario que sea muy detallada, simplemente deberá contener los requisitos principales para que el equipo pueda trabajar. Realizar este orden de tareas tiene como beneficios:

- El proyecto no se paraliza simplemente por no tener claro los requisitos menos relevantes, y el cliente podrá ver resultados de forma más rápida.
- Los requisitos secundarios aparecerán a medida que se va desarrollando el proyecto, por lo tanto, no se pierde tanto tiempo en analizarlos al principio y el cliente será más consciente de sus necesidades.
- Los requisitos secundarios puede que no se lleguen a necesitar porque se han sustituido o porque no reportan un retorno ROI interesante.

Una vez definidos los requisitos se tendrá que acordar cuándo se tiene que entender un objetivo como terminado o completado.

Se entiende que un producto está completado si:

- Asegura que se puede realizar un entregable para realizar una demostración de los requisitos y ver qué se han cumplido.
- Incluirá todo lo necesario para indicar que se está realizando el producto que el cliente desea.

Como complemento a la definición de completado, se debería de asociar una condición de aceptación o no aceptación a cada objetivo en el mismo momento en el que se crea la lista.

3.7.1.5. Sprint Backlog

Es la lista de tareas que elabora el equipo durante la planificación de un Sprint. Se asignan las tareas a cada persona y el tiempo que queda para terminarlas.

De esta manera el proyecto se descompone en unidades más pequeñas y se puede determinar o ver en qué tareas no se está avanzando e intentar eliminar el problema.

Requisito	Tarea	Quien	Estado (No iniciada / en progreso / completada)	Dia:	1	2	3	4	5	6	7	8	9	10
				Horas pendientes	1120	1088	1076	1048	1040	1032	1020	1008	992	972
Requisito A	Tarea 1	Joao	Completada		16	8								
Requisito A	Tarea 4	Laura	Completada		4									
Requisito A	Tarea 5	Laura	Completada		4									
Requisito A	Tarea 3	Gabri	Completada		8									
Requisito A	Tarea 2	Laura	Completada		16	8	4							
Requisito A	Tarea 6	Gabri	Completada		8	8	8							
Requisito A	Tarea 7	Joao	Completada		16	16	16	8						
Requisito A	Tarea 8	Laura	Completada		8	8	8							
Requisito A	Tarea 9	Laura	Completada		8	8	8	8	8					
Requisito A	Tarea 10	Laura	Completada		8	8	8	8	8	8	4			
Requisito A	Tarea 11	Joao	Completada		16	16	16	16	16	16	8			
Requisito B	Tarea 12	Gabri	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 13	Laura	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 14	Joao	En progreso		8	8	8	8	8	8	8	8	8	4
Requisito B	Tarea 15	Gabri	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito B	Tarea 16	Laura	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 17	Joao	No iniciada		4	4	4	4	4	4	4	4	4	4
Requisito C	Tarea 18	Gabri	No iniciada		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 19	Laura	No iniciada		16	16	16	16	16	16	16	16	16	16
Requisito C	Tarea 20	Joao	No iniciada		8	8	8	8	8	8	8	8	8	8

Figura. 3.3 Ejemplo de Sprint Backlog.

3.7.1.6. Incremento

Representa los requisitos que se han completado en una iteración y que son perfectamente operativos.

Según los resultados que se obtengan, el cliente puede ir haciendo los cambios necesarios y replanteando el proyecto.

Finalmente el Product Backlog irá evolucionando mientras el producto exista en el mercado. Esta es la forma para evolucionar y tener un valor de producto para el cliente suficiente para ser competitivo.

3.7.1.7. El Proceso

Un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas, límite máximo de feedback y reflexión). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos o requisitos priorizados del producto, en los cuales el cliente define los objetivos prioritarios balanceando el valor que le aportan respecto a su costo y quedan repartidos en iteraciones y entregas.

Las actividades que se llevan a cabo en Scrum son las siguientes:

3.7.1.8. Planificación de la iteración

El primer día de la iteración se realiza la reunión de planificación de la iteración. Tiene dos partes:

Selección de requisitos (4 horas máximo). El cliente presenta al equipo la lista de requisitos primordiales del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más importantes que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

Planificación de la iteración (4 horas máximo). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a los que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se auto asignan las tareas.

3.7.1.9. Ejecución de la Iteración

Cada día el equipo realiza una reunión de sincronización (15 minutos máximo) en donde cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. En la reunión cada miembro del equipo responde a tres preguntas:

¿Qué he hecho desde la última reunión de sincronización?

¿Qué voy a hacer a partir de este momento?

¿Qué impedimentos tengo o voy a tener?

Durante la iteración el Facilitador (Scrum Master) se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad.

Elimina los obstáculos que el equipo no puede resolver por sí mismo.

Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

Durante la iteración, el cliente junto con el equipo refinan la lista de requisitos (para prepararlos para las siguientes iteraciones) y, si es necesario, cambian los objetivos del proyecto para maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

3.7.1.10. Inspección y Adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

Demostración (4 horas máximo). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, cambiando el proyecto.

Retrospectiva (4 horas máximo). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de ir eliminando los obstáculos identificados.

4 Análisis y Diseño del Sistema

4.1. Metodología a utilizar: SCRUM

La metodología que se va a utilizar para la elaboración de este proyecto es SCRUM, por ser una metodología actual, orientada a objetos, dirigida por casos de uso y con arquitectura céntrica, iterativa e incremental; la cual, se detalla a continuación.

Utilizando el ciclo de vida iterativo o incremental, el sistema fue liberándose periódicamente ya que cada entrega fue representando un incremento de funcionalidad respecto a la entrega anterior (sprint[1]).

1. El segundo pilar más importante de scrum son las revisiones. Las reuniones con el encargado del laboratorio de Microsoft (cliente) fuero la base para lograr transparencia en el proceso y comunicación efectiva, y posibilitan las características de un equipo ágil:

    - **Reunión de planificación.** Se tiene una reunión inicial con el cliente (sprint) para ver la problemática que tenía el sistema de préstamos actual y así definir los objetivos del sistema. El cliente presenta la lista de requisitos primordiales del proyecto. Se preguntan las dudas que surgen y se selecciona los requisitos más importantes, comprometiéndose a cumplirlos en cada iteración, de manera que puedan ser entregados si él cliente lo solicita. A su vez, el desarrollador elaboró la lista de tareas de la iteración necesarias para desarrollar los requisitos a los que se ha comprometido.

    - **Reunión diaria.** Se realiza una reunión de aproximadamente 15 min, viéndose los avances que tuvo el sistema y los problemas que surgieron, para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. Es en esta reunión donde se cambiaron algunos de los objetivos del proyecto para mejorar la utilidad de lo que se está desarrollando.

    - **Reunión de revisiones.** El último día de la iteración se realizó una reunión de revisión, en la cual se presentó al cliente los requisitos completados. En función de los resultados mostrados y de los cambios que se hicieron del proyecto, el cliente realizó las adaptaciones necesarias.

- **Retrospectiva del Sprint.** El desarrollador analizó cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle avanzar adecuadamente, mejorando así su productividad.

4.2. Arquitectura a utilizar

4.2.1. Programación por capas

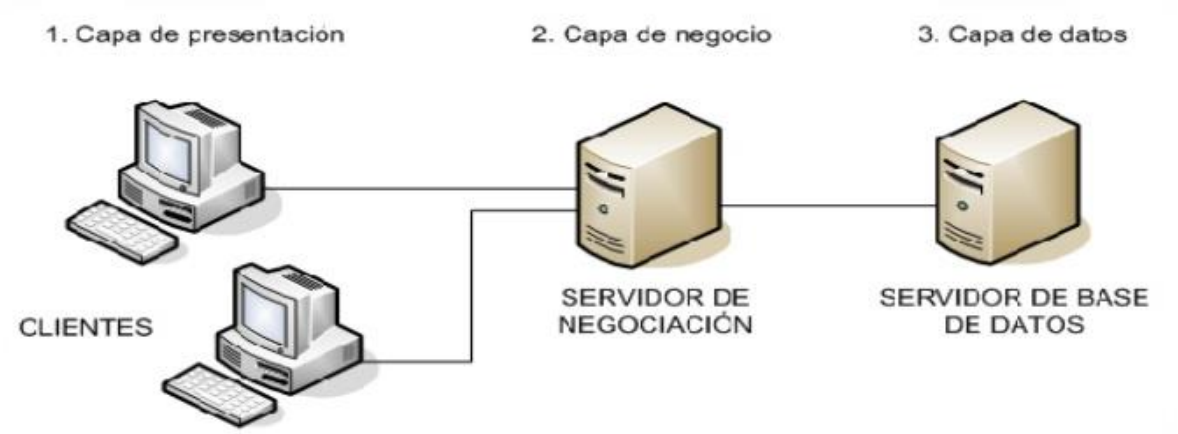


Figura.4.1 Programación por Capas

Es un estilo de programación, su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos (Figura. 3.1).

Las ventajas de este estilo de programación son las siguientes:

1. El desarrollo se puede llevar en varios niveles y en caso de que sobrevenga algún cambio.
2. En el diseño de sistemas informáticos actuales se suelen usar las arquitecturas multilineal o programación por capas.
3. Permite distribuir el trabajo de creación de una aplicación por niveles; cada grupo de trabajo está totalmente separado del resto de los niveles, de forma q basta con conocer la API (Application Programming Interface)[2] que existe entre niveles.

4.2.1.1. Capa de Presentación

Es la capa que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. Se comunica únicamente con la capa de negocio

Esta capa es conocida también como la interfaz gráfica y debe tener la característica de ser "amigable" para el usuario (Figura. 3.2).

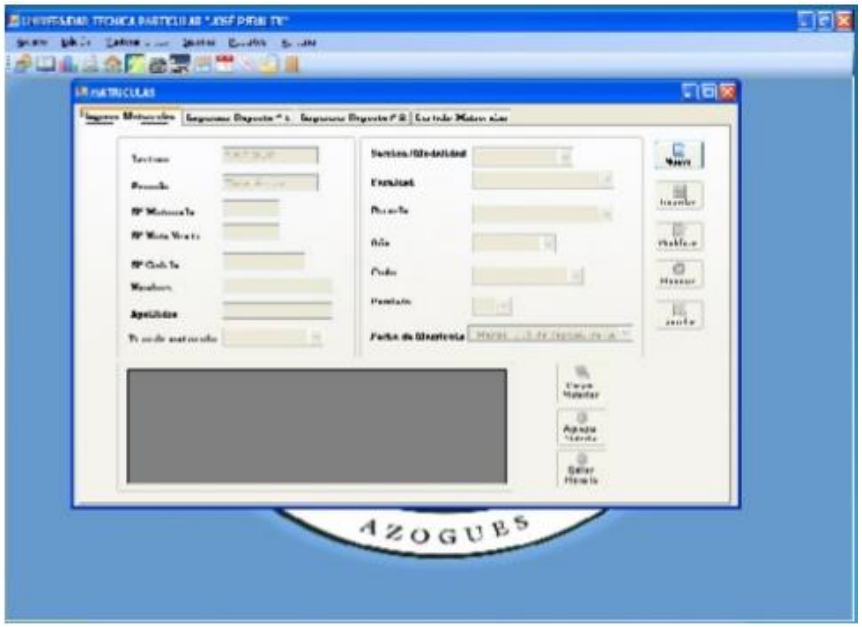


Figura. 3.2 Capa de Presentación

4.2.1.2. Capa de Negocio

Se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica de negocio) porque es aquí donde se establece todas la reglas que deben cumplirse.

Esta capa se comunica con la capa de presentación, para recibir los solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar la información.

Toda aplicación tiene código para implementar reglas de negocio. Se puede almacenar la lógica de negocio sobre cada estación de cliente u optar por ejecutar la lógica de negocio sobre un servidor de aplicación.
No toda la lógica de negocio es la misma algunas no requieren un frecuente acceso a los datos, pero una interfaz de usuario robusta necesitara de la lógica de negocios para la validación en la entrada de campos, cálculos en tiempo real u otras interacciones de usuario.

- **ADO.NET**
  Conjunto de componentes del software que pueden ser usados por los programadores para acceder a datos y a servicios de datos.

- ✓ Data provider. Clase que proporciona el acceso a una fuente de datos, como Microsoft SQL Server y Oracle.
- ✓ Data sets.
  - Los *objetos DataSets.* Grupo de clases que describen una simple base de datos relacional en memoria. Representa una base de datos entera, puede contener las tablas y relaciones.
  - *Objeto DataTable.* Representa una sola tabla en la base de datos.
  - *DataSets* es llenado desde una base de datos por un DataAdapter cuyas propiedades son Connection y Command.

- Conjunto común de clases de utilidad

  - ✓ Connection. Conexión con la fuente de datos.
  - ✓ Command. Acción en la fuente de datos.
  - ✓ Parameter. Parámetro para un procedimiento almacenado.
  - ✓ DataAdapter. Puente entre la fuente de datos y los objetos DataSet.
  - ✓ DataReader. Procesa la lista de resultados de un registro a la vez.

4.2.1.3. Capa de Datos

Es donde se encuentran los datos y se accede a los mismos. Administrada por uno o más gestores de bases de datos que realizan todo el almacenamiento de base de datos, recibe solicitudes de almacenamientos o recuperación de información desde la capa de negocios.

4.2.1.4. Arquitectura de Tres Capas para la WEB

- Los datos y servicios aparecen por separado.
- Fácil de separar los datos de la "lógica de negocio".
- El cliente recibe los datos y la información de forma indirecta a través del servidor.

4.2.1.5. Aplicaciones Orientadas a la WEB

En la capa de presentación está el navegador que permite visualizar la página, él mismo se comunica con el servidor web y a su vez con el servidor de prestaciones conformando la lógica de negocios y posteriormente se accede a la base de datos.

4.2.1.6. Aplicaciones No Orientadas a la WEB

La interfaz gráfica es presentada en formularios, luego la capa de negocios es implementada en el servidor de aplicaciones y en la capa de datos está la base de datos.

4.2.1.7. Arquitectura de Aplicaciones WEB

Una manera de administrar la complejidad de las aplicaciones es dividir la aplicación según sus responsabilidades o intereses. La arquitectura en capas ofrece una serie de ventajas que van más allá de la simple organización del código. Al organizar el código en capas, la funcionalidad común de bajo nivel se puede reutilizar en toda la aplicación.

Cuando se cambia o reemplaza una capa, solo deberían verse afectadas aquellas capas que funcionan con ella. Mediante la limitación de qué capas dependen de otras, se puede mitigar el impacto de los cambios para que un único cambio no afecte a toda la aplicación, además de la posibilidad de intercambiar las implementaciones en respuesta a cambios futuros en los requisitos así como facilitar el intercambio de implementaciones con fines de prueba.

Estas capas se suelen abreviar como UI (interfaz de usuario), BLL (capa de lógica de negocios) y DAL (capa de acceso a datos). Con esta arquitectura, los usuarios realizan solicitudes a través de la capa de interfaz de usuario, que interactúa con la capa BLL. BLL, a su vez, puede llamar a DAL para las solicitudes de acceso de datos (Figura. 3.3).

La capa de interfaz de usuario no debe realizar solicitudes directamente a DAL, ni debe interactuar con la persistencia directamente a través de otros medios. Del mismo modo, BLL solo debe interactuar con la persistencia a través de DAL. De este modo, cada capa tiene su propia responsabilidad conocida.

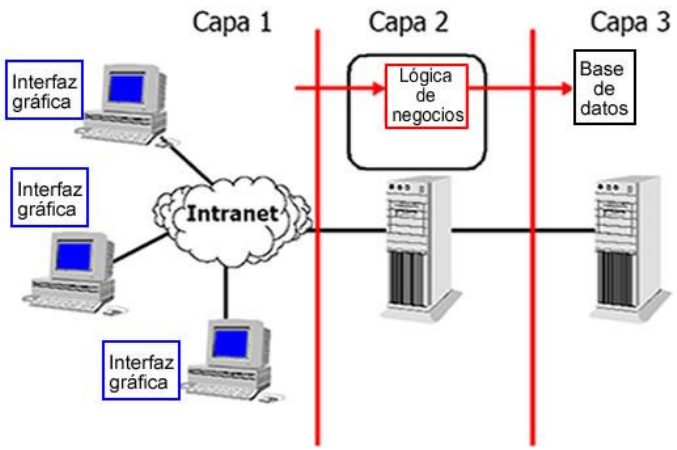


Figura.4.3 Arquitectura de Aplicación Web

4.3. Entity Framework

Es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Los datos pueden abarcar varios sistemas de almacenamiento, cada uno con sus propios protocolos; incluso las aplicaciones que funcionan con un único sistema de almacenamiento deben equilibrar los requisitos del sistema de almacenamiento con respecto a los requisitos de escribir un código de aplicación eficaz y fácil de mantener.

Permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicos del dominio, como clientes y direcciones de cliente, sin tener que preocuparse por las tablas y columnas de la base de datos donde se almacenan estos, además pueden trabajar en un nivel mayor de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales. Pueden ejecutarse en cualquier equipo en el que esté instalado .NET Framework.

4.3.1. Tipos de Modelos

Para crear una aplicación se tiene que dividir en tres partes: un modelo de dominio, un modelo lógico y un modelo físico.

- El modelo de dominio define las entidades y relaciones del sistema que se está modelando.
- El modelo lógico de una base de datos relacional normaliza las entidades y relaciones en tablas con restricciones de claves externas.
- El modelo físico abarca las capacidades de un motor de datos determinado especificando los detalles del almacenamiento en forma de particiones e índices.

4.3.2. Asignar Objetos a Datos

La programación orientada a objetos supone un desafío al interactuar con sistemas de almacenamiento de datos. Aunque la organización de clases suele reflejar la organización de las tablas de bases de datos relacionales, el ajuste no es perfecto. Varias tablas normalizadas suelen corresponder a una sola clase y las relaciones entre las clases se representan a menudo de forma diferente a las relaciones entre tablas, por esto se proporciona a los desarrolladores de software la flexibilidad para representar las relaciones de esta manera, o para modelar más estrechamente las relaciones tal como se representan en la base de datos.

Entity Framework asigna las tablas relacionales, columnas y restricciones FOREIGN KEY de los modelos lógicos a las entidades y relaciones de los modelos conceptuales, permitiendo una mayor flexibilidad al definir los objetos y optimizar el modelo lógico.

Las herramientas de Entity Data Model generan clases de datos extensibles según el modelo conceptual, clases parciales que se pueden extender con miembros adicionales que el programador agrega. De forma predeterminada, las clases que se generan para un modelo conceptual determinado derivan de las clases

base que proporcionan servicios para materializar las entidades como objetos y para realizar un seguimiento de los cambios y guardarlos.

4.3.3. Obtener Acceso a los Datos de Entidad

Se utiliza la información de los archivos del modelo y de asignación para traducir las consultas de objeto con los tipos de entidad que se representan en el modelo conceptual en consultas específicas del origen de datos, con dos entidades fundamentales:

- LINQ to Entities. Proporciona compatibilidad con Language-Integrated Query (LINQ) para consultar los tipos de entidad que se definen en un modelo conceptual.
- Entity SQL. Un dialecto de SQL, independiente del almacenamiento, que trabaja directamente con entidades del modelo conceptual y que admite conceptos de Entity Data Model. Entity SQL se utiliza tanto con consultas de objeto como con consultas que se ejecutan utilizando el proveedor EntityClient.

Entity Framework incluye el proveedor de datos de EntityClient, el cual administra las conexiones, traduce las consultas de entidad en consultas específicas del origen de datos y devuelve un lector de datos usado para materializar los datos de la entidad en los objetos. El diagrama siguiente muestra la arquitectura de Entity Framework para el acceso a datos (Figura 3.4).

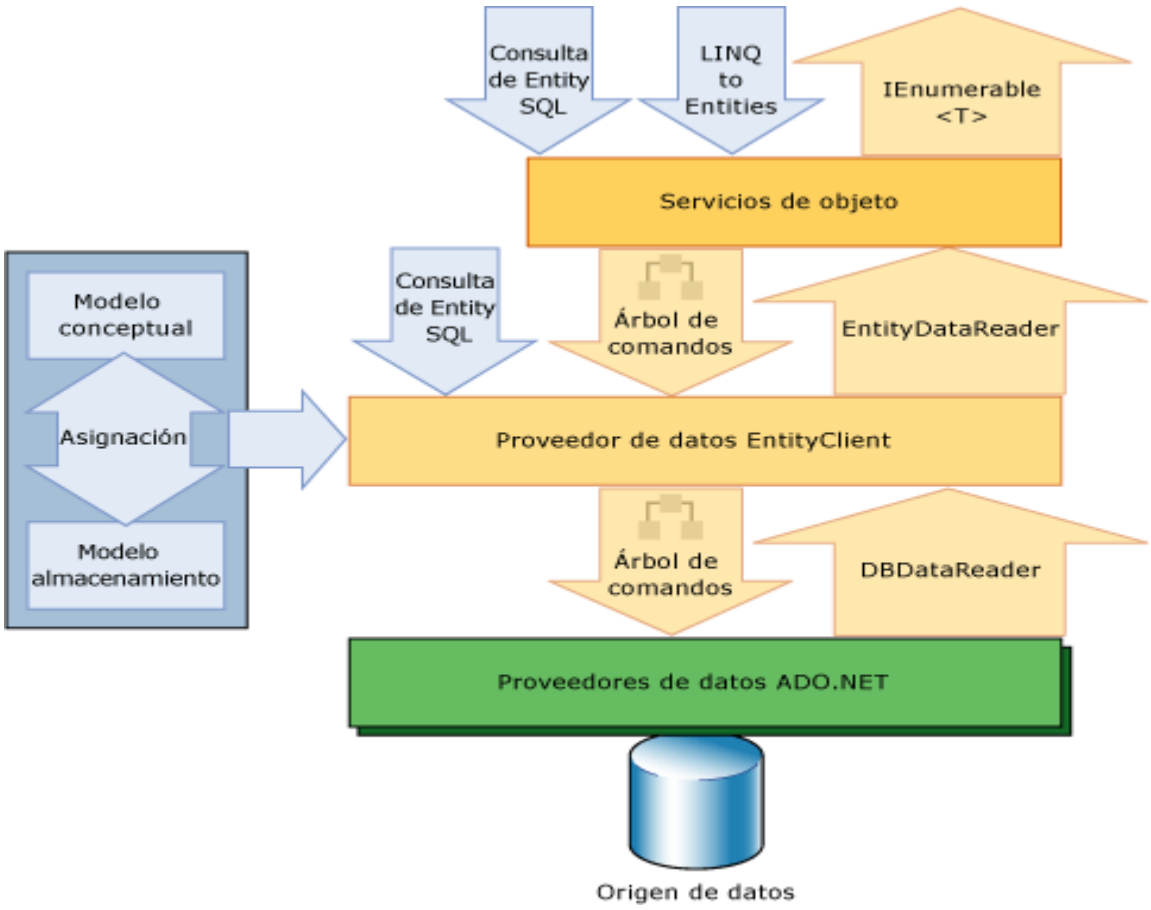


Figura 4.4. Arquitectura Entity Framework

Las herramientas de Entity Data Model pueden generar una clase derivada de **System.Data.Objects.ObjectContext** o **System.Data.Entity.DbContext** las cuales representan el contenedor de entidades definido en el modelo conceptual.

4.3.4. Proveedor de Datos

EntityClient extiende el modelo de proveedor de ADO.NET teniendo acceso a los datos en lo que respecta a las entidades conceptuales y relaciones. Ejecuta consultas que utilizan Entity SQL. Entity SQL proporciona el lenguaje de consultas subyacente que permite a EntityClient comunicarse con la base de datos.

4.3.5 Herramientas de Entity Data

Las Herramientas de Entity Data Model en Visual Studio permiten crear un archivo .edmx a partir de una base de datos o un modelo gráfico y, a continuación, actualizar ese archivo cuando la base de datos o el modelo cambie.

A partir de Entity Framework 4.0.1 también es posible crear un modelo mediante programación usando desarrollo Code First. Hay dos escenarios diferentes para el desarrollo Code First, en ambos casos, el desarrollador define un modelo codificando definiciones de clase de .NET.

También puede usar el generador de EDM, herramienta de línea de comandos utilizada para trabajar con archivos de modelo y asignación de Entity Framework generando los archivos .csdl, .ssdl y .msl a partir de un origen de datos existente.

Se puede utilizar la herramienta EdmGen.exe para:

- Conectarse a un origen de datos utilizando un proveedor de datos .NET Framework específico del origen de datos y generar los archivos de asignación.
- Validar un modelo existente.
- Generar un archivo de código de C# o Visual Basic que contenga las clases de objetos generados a partir de un archivo de modelo conceptual (.csdl).
- Generar un archivo de código de C# o Visual Basic que contenga las vistas generadas previamente para un modelo existente.

4.4 WCF Servicios Web

Windows Communication Foundation (WCF) es un marco de trabajo para la creación de aplicaciones orientadas a servicios, es un motor de ejecución y un conjunto de APIs para la creación de sistemas que envíen mensajes entre servicios y clientes. Se utilizan la misma infraestructura y API tanto para crear aplicaciones que se comuniquen entre sí en el mismo sistema, como para aplicaciones en equipos separados en distintas compañías que se comuniquen a través de Internet.

4.4.1. Mensajería

WCF se basa en la noción de comunicación basada en mensajes, y cualquier cosa que se pueda modelar como un mensaje (por ejemplo, una solicitud HTTP o un transporte de cola de mensajes) se puede representar de manera uniforme en el modelo de programación. El modelo distingue entre *clientes*, que son aplicaciones que inician la comunicación y *servicios*, que son aplicaciones que esperan a que los clientes se comuniquen con ellos y responden a esa comunicación.

Los mensajes se envían entre *endpoints*. Los *endpoints* son los lugares donde los mensajes se envían o reciben (o ambos), y definen toda la información requerida para el intercambio de mensajes.

Protocolos de comunicaciones

Un elemento requerido de la pila de la comunicación es el *protocolo de transporte*. Los mensajes se pueden enviar a través de intranets e Internet utilizando transportes comunes, como HTTP y TCP.

Patrones de mensajes

WCF admite varios patrones de mensajería, incluida la comunicación de solicitud-respuesta unidireccional y dúplex.

4.5. Arquitectura WCF

Windows Communication Foundation o WCF es una plataforma de comunicación y mensajería que forma parte del entorno de trabajo .NET. La idea fundamental es permitir el desarrollo de aplicaciones basadas en la arquitectura orientada a servicios (SOA), donde éstas se pueden ejecutar, desde una maquina local hasta internet, de manera simple y segura.

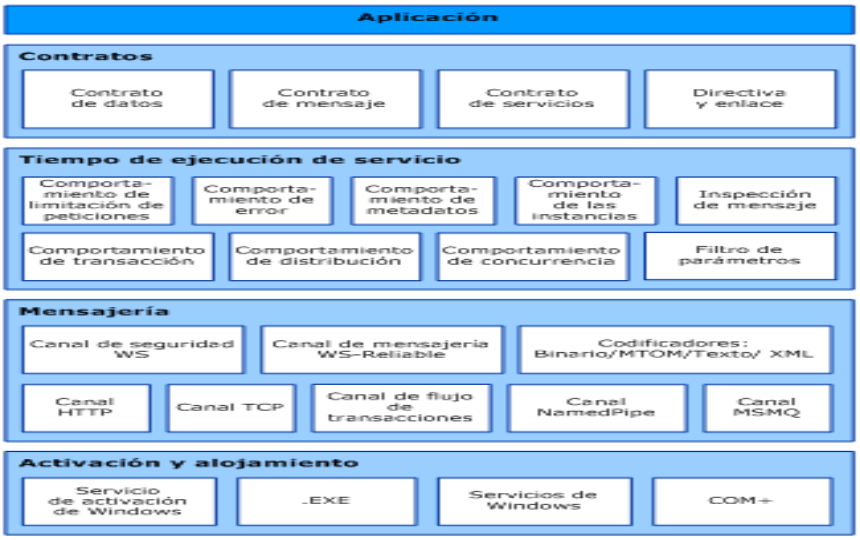


Figura 4.5 Arquitectura WCF

4.5.1. Contratos y Descripciones

Definen varios aspectos del sistema de mensajes, describe cada parámetro que constituye cada mensaje que un servicio puede crear o utilizar. Los documentos de Lenguaje de definición de esquemas XML definen los parámetros de mensaje, permitiendo a cualquier sistema que entienda y procese los documentos. Define partes específicas del mensaje utilizando los protocolos SOAP y permite el control sobre las partes del mensaje, especifica las firmas de método actuales del servicio y se distribuye como una interfaz en uno de los lenguajes de programación compatibles, como Visual Basic o Visual C#.

4.5.2. Tiempo de Ejecución de Servicio

La capa del tiempo de ejecución del servicio contiene los comportamientos que solo se producen durante la operación actual del servicio, es decir, los comportamientos en tiempo de ejecución del servicio.

4.5.3. Mensajería

La capa de la mensajería está compuesta por *canales*. Un canal es un componente que procesa un mensaje de alguna manera. Un conjunto de canales también se conoce como una *pila de* canales.Los canales funcionan en los mensajes y encabezados del mensaje. Esto es diferente de la capa en tiempo de ejecución del servicio, que se ocupa principalmente de procesar el contenido de los cuerpos de los mensajes.

- Hay dos tipos de canales: Canales de transporte: Los canales de transporte leen y escriben mensajes de la red (o algún otro punto de la comunicación con el mundo externo).

- Canales de protocolo: Los canales de protocolo implementan protocolos de procesamiento de mensajes, a menudo leyendo o escribiendo encabezados adicionales en el mensaje.

4.5.4. Alojamiento y Activación

Un programa se debe iniciar en un ejecutable, conocido como servicio con host propio.

Los servicios también se pueden hospedar o activar en un ejecutable administrado por un agente externo, como IIS o Servicio de activación de Windows (WAS). Los componentes COM también se pueden hospedar como servicios WCF.

4.6. ASP. NET

ASP.NET es un marco web gratuito para crear sitios y aplicaciones web con HTML, CSS y JavaScript. También permite crear las API Web y usar tecnologías en tiempo real como Sockets Web.

ASP.NET ofrece tres marcos de trabajo para crear aplicaciones web: páginas Web de ASP.NET, formularios Web Forms y ASP.NET MVC. Todos los marcos son estables y consolidados, y permiten crear aplicaciones web excelentes. Independientemente del framework elegido, se obtendrán todas las ventajas y características de ASP.NET (Figura 3.6).

	Si tiene experiencia	Estilo de desarrollo	Experiencia
formularios Web Forms	Windows Forms, WPF, .NET	Desarrollo rápido mediante una biblioteca de controles que encapsulan código HTML enriquecida	RAD de nivel intermedio, avanzado
MVC	Ruby sobre raíles, .NET	Control total sobre marcado HTML, código y marcado separado y fácil de escribir pruebas. La mejor opción para las aplicaciones móviles y de página (SPA).	Nivel intermedio, avanzado
Páginas web	Clásico ASP, PHP	Marcado HTML y el código conjuntamente en el mismo archivo	Nuevo, de nivel intermedio

Figura 4.6 Marcos de trabajo ASP

4.6.1. Aplicaciones de la Página

ASP.NET única (SPA) le ayuda a crear aplicaciones que incluyen importantes interacciones del lado cliente mediante HTML 5, 3 de CSS y JavaScript. Visual Studio incluye una plantilla para la creación de aplicaciones de una página con ASP.NET Web API, además de la plantilla.

4.7. Java Script

JavaScript es un sencillo lenguaje de programación que presenta una característica especial, sus programas llamados scripts, se encuentran en páginas HTML y se ejecutan en un navegador (Mozila Firefox, Microsoft Internet Explorer,...). Estos scripts consisten en unas funciones que son llamadas desde HTML cuando un evento sucede. De ese modo podemos añadir efectos o abrir una ventana nueva al pulsar en un enlace.

4.8. CSS (Hojas de Estilo en Cascada)

Cascading Style Sheets es el lenguaje utilizado para describir la presentación de documentos HTML o XML, esto incluye varios lenguajes basados en XML como son XHTML o SVG. CSS describe como debe ser diseñado el elemento estructurado en pantalla, en papel, hablado o en otros medios.

4.9. Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc.

Visual Studio permite crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas, entre otros.

4.10. Microsoft SQL Server Express

Es un sistema de gestión de bases de datos de calidad empresarial que generalmente está alojado en un servidor y puede escalar a través de múltiples servidores y ubicaciones, caracterizado principalmente por:

- Desplegar soluciones de escritorio de Windows que usan bases de datos de SQL Server para usuarios que no poseen SQL Server.
- Desplegar aplicaciones web livianas que usan bases de datos de SQL Server.
- Desarrolladores que desean crear y probar aplicaciones con una base de datos de SQL Server por sí mismas, en lugar de una base de datos alojada en un servidor.
- Aplica en cualquier lenguaje y plataforma, tanto localmente como en cloud, en contenedores Windows, Linux y Docker.
- Base de datos menos vulnerable protegiendo los datos en reposo y en movimiento.
- Análisis en tiempo real a una velocidad de un millón de predicciones por segundo.
- Convierte los datos sin procesarse en informes útiles que pueden ser distribuidos en cualquier dispositivo, con un costo menor otras soluciones.

4.11. Servidor Web IIS

Proporciona una plataforma segura, fácil de administrar, modular y extensible para hospedar sitios web, servicios y aplicaciones de manera confiable. Puede compartir información con usuarios en Internet, en una intranet o en una extranet. Esta plataforma web es unificada, integra IIS, ASP.NET, servicios de FTP, PHP y Windows Communication Foundation (WCF).

A continuación se indican algunas ventajas de usar IIS:

- La seguridad web se refuerza gracias a una superficie reducida de servidor y al aislamiento automático de aplicaciones.
- Implementar y ejecutar aplicaciones web de ASP.NET, ASP clásico y PHP en el mismo servidor de forma sencilla.
- Aislamiento de aplicaciones al proporcionar a los procesos de trabajo una identidad única y una configuración en espacio aislado de manera predeterminada, lo que reduce aún más los riesgos de seguridad.
- Agrega y elimina componentes IIS integrados e incluso reemplazarlos fácilmente por módulos personalizados que se adapten a las necesidades del cliente.
- Aumenta la velocidad del sitio web mediante el almacenamiento en caché dinámico integrado y la compresión mejorada.

4.12. Diseño de la Base de Datos

Una base de datos bien estructurada ahorra espacio en el disco eliminando los datos redundantes, mantiene la precisión e integridad de los datos y ofrece acceso a los datos de formas útiles. Diseñar una base de datos útil y eficiente requiere seguir el proceso adecuado, incluidas las siguientes etapas:

4.12.1. Análisis de los Requisitos

Comprender el propósito de la base de datos determinará las opciones en todo el proceso de diseño. Asegurarse de observar la base de datos desde todas las perspectivas. Algunas formas de reunir información antes de crear la base de datos:

- Entrevistar a las personas que la usarán.
- Analizar formularios de negocio, como facturas, plantillas de horas trabajadas, encuestas.
- Examinar cualquier sistema de datos existente (incluidos archivos físicos y digitales).

Esta etapa se inicia reuniendo cualquier dato existente que se incluirá en la base de datos. Luego enumera los tipos de datos que quieres almacenar y las entidades o personas, cosas, ubicaciones y eventos que esos datos describen, del siguiente modo:

Usuario

- Usu_matricula
- Usu_nombre

- Usu_appaterno
- Usu_appaterno
- Usu_email

Libro

- Lib_id
- Lib_nombre
- Lib_editorial
- Lib_autor
- Lib_cantidad

Comprobante

- Comp_id
- Comp_estado
- Comp_fecha
- Comp_fecha_dev

Más adelante, esta información se volverá parte del directorio de datos, que describe las tablas y los campos dentro de la base de datos. Asegurarse de dividir la información en partes útiles lo más pequeñas posibles. Además, evita ubicar el mismo punto de datos en más de una tabla porque agregarás una complejidad innecesaria.

Cuando se sabe qué tipos de datos incluir, las bases de datos de dónde provienen esos datos y cómo se usarán, estarás listo para comenzar a planificar la base de datos real.

4.12.2. Estructura de la Base de Datos

El siguiente paso es organizar la representación visual de tu base de datos. Para ello, debes comprender exactamente cómo se estructuran las bases de datos relacionales.

Dentro de una base de datos, los datos relacionados se agrupan en tablas, cada una de ellas consiste en filas (también llamadas "tuplas") y columnas, como una hoja de cálculo.

Para convertir tus listas de datos en tablas, comienza creando una tabla para cada tipo de entidad.Cada fila de una tabla se llama "registro". Los registros incluyen datos sobre algo o alguien, como un cliente específico. En cambio, las columnas (también conocidas como "campos

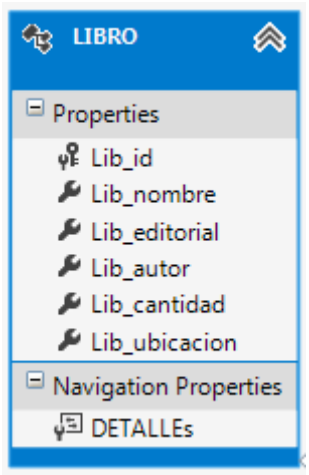
" o "atributos") contienen un único tipo de información que aparece en cada registro, como las direcciones de todos los clientes enumerados en la tabla.

	Usu_matricula	Usu_nombre	Usu_appatemo	Usu_apmatemo	Usu_email
1	9876543	HECTOR	AGUIRRE	HERNANDEZ	hector_78@yahoo.com
2	9976543	ALAN	MENDOZA	MARQUEZ	alan@gmail.com
3	93001243	miguel	diaz	lopez	slugger_09810@yahoo.com
4	99389765	LUIS	LOPEZ	DIAZ	luis@gmail.com

Con el fin de que los datos sean consistentes de un registro al siguiente, asigna el tipo de datos apropiado a cada columna. Los tipos de datos comunes incluyen:

- CHAR - una longitud específica de texto.
- VARCHAR - texto de longitudes variables.
- TEXT - grandes cantidades de texto.
- INT - número entero positivo o negativo.
- FLOAT, DOUBLE - también puede almacenar números de punto flotante.
- BLOB - datos binarios.

Algunos sistemas de gestión de bases de datos también ofrecen el tipo de datos denominado "Autonumeración", que genera automáticamente un número único en cada fila. El título de cada recuadro debería indicar qué describen los datos en la tabla, mientras que los atributos están enumerados.



Por último, se decide que atributo o atributos funcionarán como clave primaria para cada tabla, si procede. Una clave primaria (PK) es un identificador único para una entidad determinada, esto significa que puedes seleccionar un cliente concreto incluso si solo se conoce ese valor.

Los atributos seleccionados como claves primarias deben ser únicos, inalterables y estar siempre presentes (nunca NULL o vacíos). Por este motivo, las matrículas y los nombres de usuario son excelentes claves primarias, mientras que los números de teléfono o direcciones postales no lo son. También puedes usar múltiples campos conjuntamente como la clave primaria (esto se denomina "clave compuesta").

4.12.3. Relación Entre Entidades

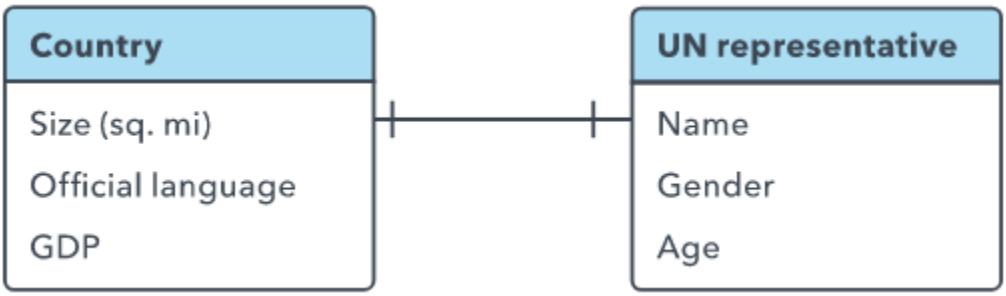
Cuando se tienen establecidas las tablas de la base de datos, se comienza a analizar las relaciones entre ellas. La cardinalidad se refiere a la cantidad de elementos que interactúan entre dos tablas relacionadas, identificarla ayuda a verificar que se han dividido los datos en tablas de la forma más eficiente.

Cada entidad puede, potencialmente, tener una relación con todas las demás, pero por lo general esas relaciones pueden ser de uno de tres tipos:

1. Relaciones uno a uno
2. Relaciones uno a muchos
3. Relaciones muchos a muchos

4.12.3.1. Relación Uno a Uno

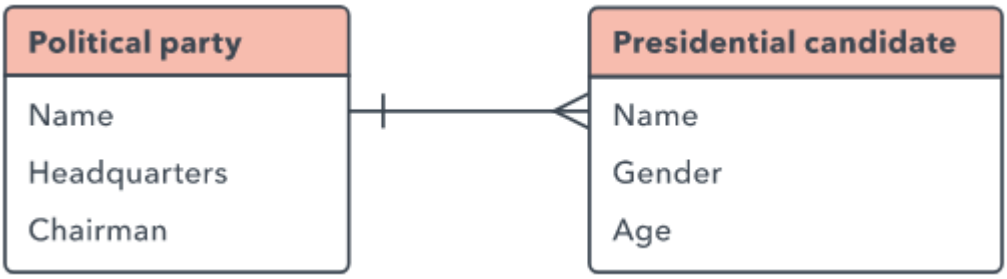
Si hay una única instancia de la Entidad A para cada instancia de la Entidad B, se dice que tienen una relación de uno a uno (a menudo se escribe 1:1). Se indica en un diagrama ER mediante una línea con un guión en cada extremo:



Generalmente indica que la mejor opción sería combinar los datos de las dos tablas en una sola tabla. Sin embargo, quizás desees crear tablas con una relación de uno a uno en una serie particular de circunstancias.

4.12.3.2. Relación Uno a Muchos

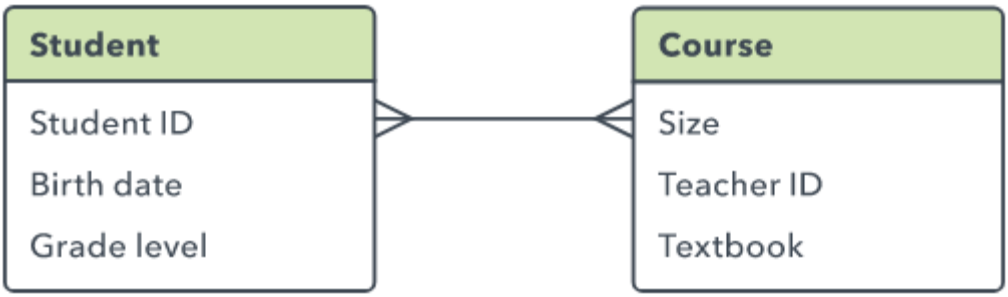
Estas relaciones suceden cuando un registro de una tabla está asociado a múltiples entradas en otra tabla. Por ejemplo, un solo alumno puede haber solicitado múltiples pedidos o una persona haberse llevado varios artículos del laboratorio a la vez. Las relaciones uno a muchos (1:M) se indican con lo que se denomina "notación patas de gallo".



Para implementar una relación uno a muchos (1:M) se agrega la llave primaria de un lado de la relación como un atributo en la otra tabla. Cuando una llave primaria se coloca en otra tabla, se denomina "llave foránea". La tabla en el lado "1" de la relación es considerada una tabla principal respecto de la tabla secundaria que se encuentra del otro lado.

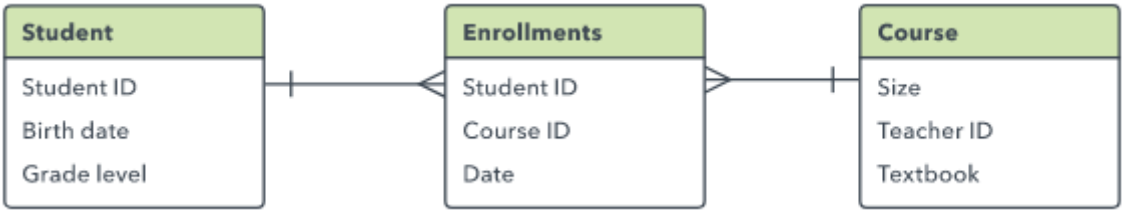
4.12.3.3. Relación Muchos a Muchos

Cuando múltiples entidades de una tabla se pueden asociar a múltiples entidades de otra tabla, se dice que tienen una relación de muchos a muchos (M:N). Esto puede suceder en el caso de estudiantes y clases, ya que un estudiante puede inscribirse en muchas clases, y una clase puede tener numerosos estudiantes.



Lamentablemente, no es posible implementar directamente este tipo de relación en una base de datos, debe dividirse en dos relaciones uno a muchos, para ello, se crear una nueva entidad entre esas dos tablas. Esta clase de entidad intermedia se llama "tabla de enlaces", "entidad asociativa" o "tabla de unión" en diversos modelos.

Cada registro de la tabla de enlaces se correspondería con dos de las entidades de las tablas contiguas (también puede incluir información adicional).



4.12.4. Normalización de la Base de Datos

Una vez que se tiene un diseño preliminar para la base de datos, se pueden aplicar reglas de normalización para asegurar que las tablas estén estructuradas correctamente.

4.12.4.1. Primera Forma Normal

La primera forma normal (abreviada como "1FN") especifica que cada celda de la tabla puede tener un solo valor, nunca una lista de valores. Por lo tanto, una tabla como esta no cumple con los requisitos:

ID del producto	Color	Precio
1	marrón, amarillo	$15
2	rojo, verde	$13
3	azul, naranja	$11

Quizás se crea que la mejor solución sea dividir los datos en columnas adicionales, pero eso también rompería las reglas: una tabla con grupos de atributos repetidos o estrechamente relacionados entre sí no cumple con la primera forma normal. Por ejemplo, la tabla a continuación no cumple con los requisitos:

En cambio, divide los datos en múltiples tablas o registros hasta que cada celda contenga solo un valor y no halla columnas adicionales. En este punto, se dice que los datos son "atómicos", es decir que se dividen en partes útiles lo más pequeñas posibles. Para la tabla anterior, podrías crear una tabla adicional llamada "Datos de ventas", que haría coincidir productos específicos con ventas. Así, "Ventas" tendría una relación 1:M con "Datos de ventas".

4.12.4.2. Segunda Forma Normal

La segunda forma normal (2NF) establece que todos los atributos deben ser totalmente dependientes de toda la clave primaria. Eso significa que cada atributo debería depender directamente de la clave primaria, en lugar de indirectamente a través de algún otro atributo.

Por ejemplo, se considera que el atributo "edad" que depende de "fecha de nacimiento", que a su vez depende de "ID de estudiante" tiene una dependencia funcional parcial; y una tabla que contenga estos atributos no cumpliría con la segunda forma normal.

Además, una tabla con una llave primaria compuesta de múltiples campos viola la segunda forma normal si uno o más de los otros campos no dependen de cada parte de la clave.

Por lo tanto, una tabla con estos campos no respetaría la segunda forma normal porque el atributo "Nombre del producto" depende del ID del producto, pero no del número de pedido:

- Número de pedido (llave primaria).
- ID de producto (llave primaria).
- Nombre del producto.

4.12.4.3. Tercera Forma Normal

La tercera forma normal (3NF) agrega a estas reglas el requisito de que cada columna que no sea de clave sea independiente de las demás columnas. Si modificar el valor en una columna que no sea de clave hace que cambie otro valor, entonces esa tabla no cumple con los requisitos de la tercera forma normal.

Esto evita que almacenes cualquier dato derivado en la tabla, tal como la columna "Impuestos" a continuación, que depende directamente del precio final del pedido:

Pedido	Precio	Impuestos
14325	$40.99	$2.05
14326	$13.73	$.69
14327	$24.15	$1.21

Se han propuesto formas adicionales de normalización, incluidas la forma normal de Boyce-Codd, la cuarta, quinta y sexta forma normal, y la forma normal de dominio/llave, pero las primeras tres son las más comunes.

Si bien estas formas explican las buenas prácticas que se deben seguir generalmente, el grado de normalización depende del contexto de la base de datos.

4.12.5. Sistema de Gestión de Base de Datos

Muchas de las elecciones de diseño que se toman dependen del sistema de gestión de base de datos que elijas. Algunos de los sistemas más comunes incluyen:

- Oracle DB
- MySQL
- Microsoft SQL Server
- PostgreSQL
- IBM DB2

Cuando se pueda elegir, seleccionar un sistema de gestión de base de datos en función del costo, los sistemas operativos, las funciones y más.

4.13. Modelo Entidad Relación

Un diagrama entidad relación es un modelo de datos que representará un conjunto de objetos llamados entidades y relaciones, así como las interrelaciones y propiedades.

Es una herramienta enfocada en el modelado de comunicación para la administración de la base de datos, a través de la cual se pueden identificar el tipo de claves, índices o apuntadores que se necesitarán para llegar de manera eficiente a los registros de base de datos.

Una entidad se define como un objeto real, que existe en un contexto determinado el cual puede llegar a existir y del cual se desea almacenar la información o sus atributos, dado que son las características o propiedades que se asocian a una determinada entidad y que toman valor en una determinada instancia en particular.

Los diagramas de entidad relación tienen como características los siguientes elementos:

Entidad: Es la representación de un objeto u elemento del mundo real que tiene existencia independiente, lo que implica que podrá ser descrito por lo que puede diferenciarse respecto de otro objeto incluso de otro elemento del mismo tipo o una misma entidad.

Atributos: Son las característica que identifican a una entidad, de manera general utilizan o implementan características relevantes y que deberán ser consideradas para el diseño.

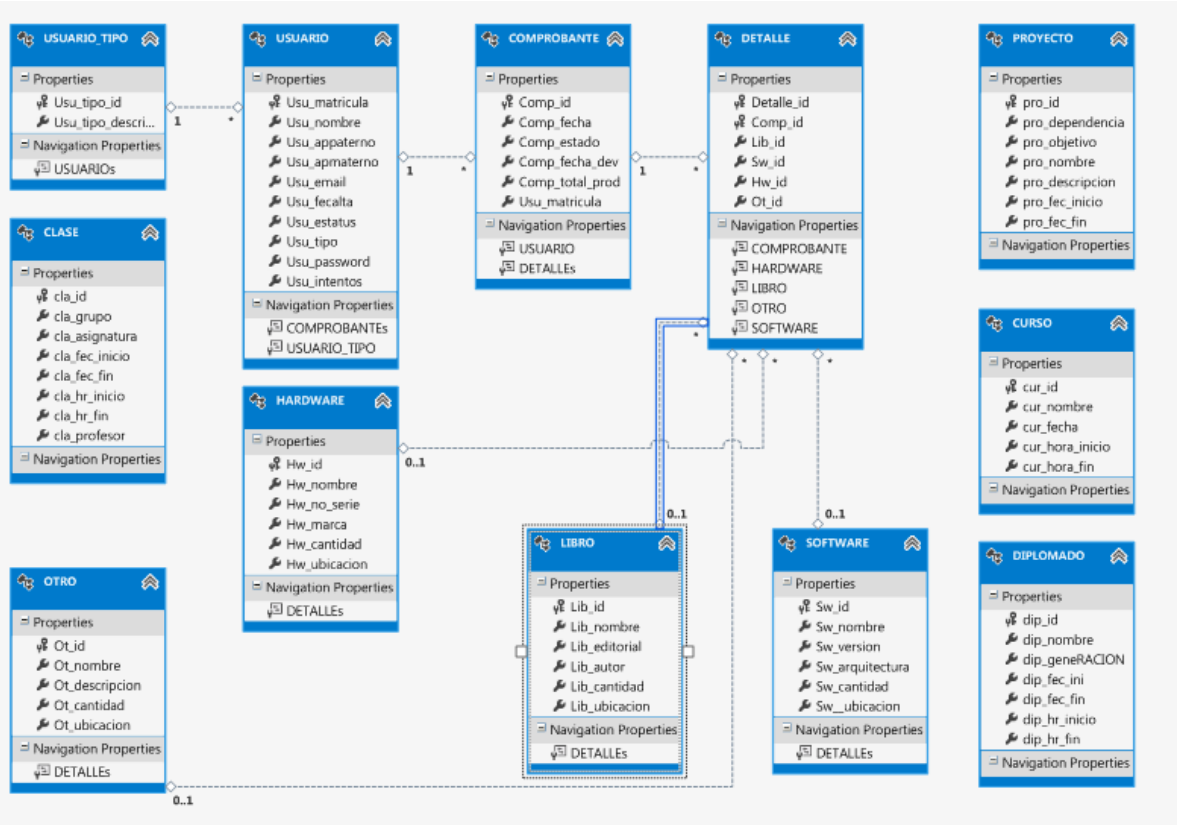
Relaciones: Se define como la relación entre dos entidades, que tienen como características el mismo origen (verbo) o bien algún campo en común.

Cardinalidad: Se define como un número de ocurrencias que pueden existir entre un par de entidades.

Llave primaria: Esta característica tiene como funcionalidad identificar a los elementos de un conjunto de entidades, por lo que permite identificar de manera única cada renglón dentro de una tabla, así mismo permite distinguir las relaciones de un conjunto de relaciones previamente definido.

Estructura de Datos: Se llama estructura de datos a la relación que se tiene dentro de un grupo de datos que están relacionados con otros y que de manera conjunta describen un componente determinado del sistema.

A continuación se muestra el diagrama entidad relación del sistema



4.14. Diagramas de Caso de Uso

Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto. Son una excelente forma de especificar el comportamiento externo de un sistema, ayudando al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema.

La mayoría de los sistemas tienen muchos tipos de usuarios. Cada tipo de usuarios se representa mediante un actor. Los actores utilizan el sistema al interactuar con los casos de uso. Los actores se comunican con el sistema mediante el envío y recepción de mensajes hacia y desde el sistema. Para el sistema deseado se identifican los siguientes actores:

- Alumno: Estudiantes de la facultad de ingeniería pertenecientes a las carreras de electrónica, computación y telecomunicaciones que pueden realizar una solicitud de servicio.
- Académico: Personal de la facultad de Ingeniería de las carreras de electrónica, computación y telecomunicaciones
- Administrador: Personal del laboratorio de Microsoft.

Los diagramas de caso de uso para cada uno de los actores del sistema se detallan a continuación, de acuerdo a la notación básica de UML (Figura 3.7)

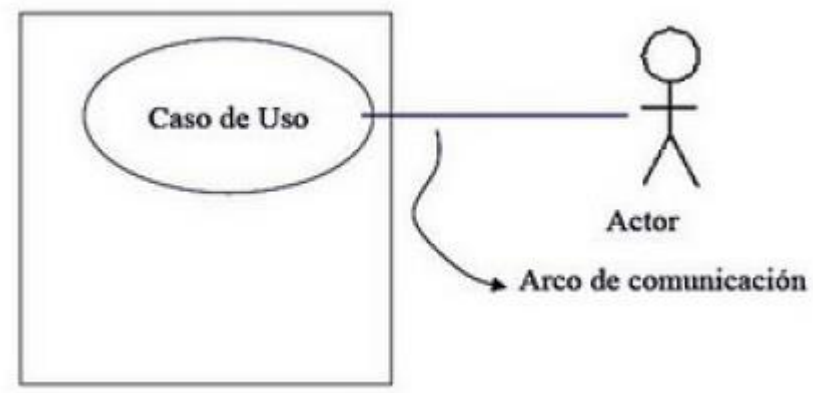
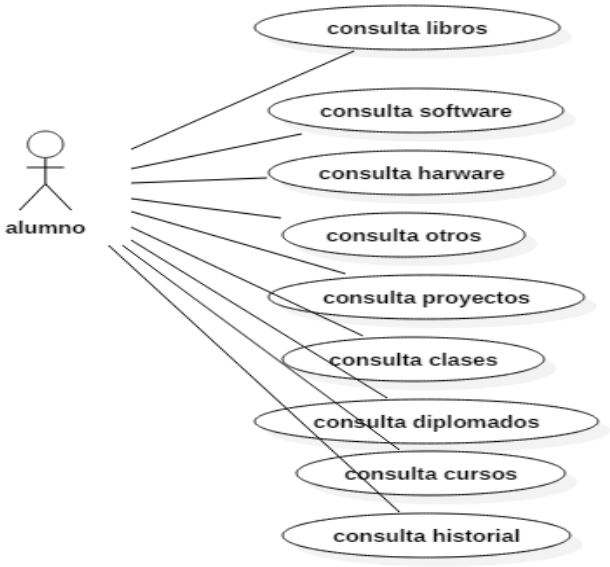
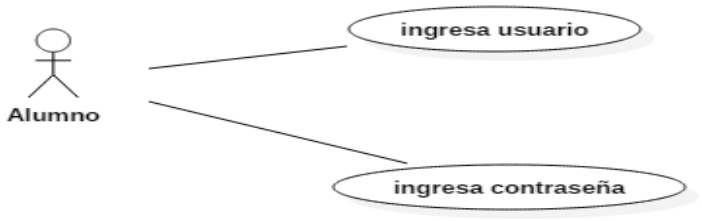


Figura 4.7 Notación básica de UML

4.14.1. Consulta Alumno

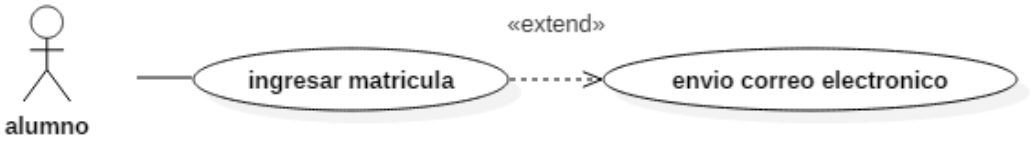


4.14.2. Alumno Login

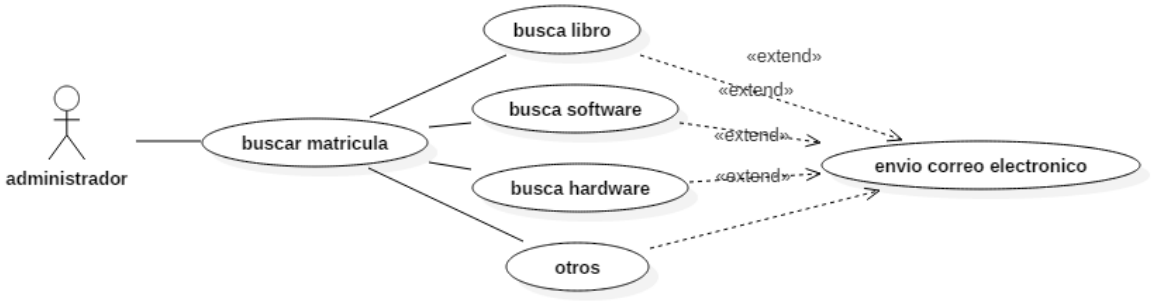


4.13.3. Alumno Cambia Password

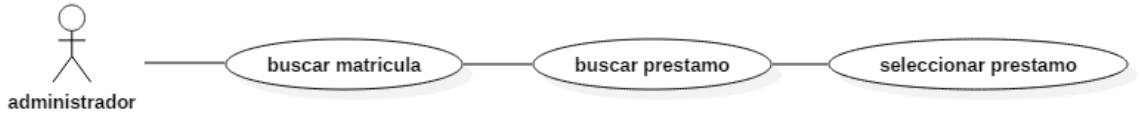
4.13.4. Alumno Recupera Contraseña



4.13.5. Administrador Préstamo

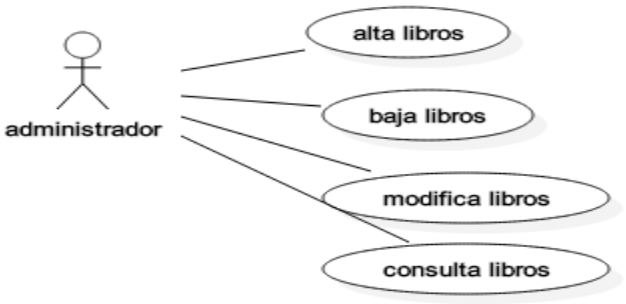


4.13.6. Administrador Devolución

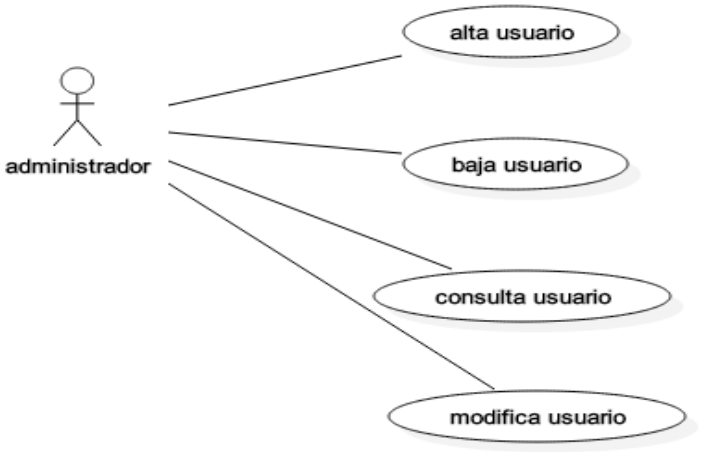


4.13.7. Administrador Alta Clases

4.13.8. Administrador Alta Libros



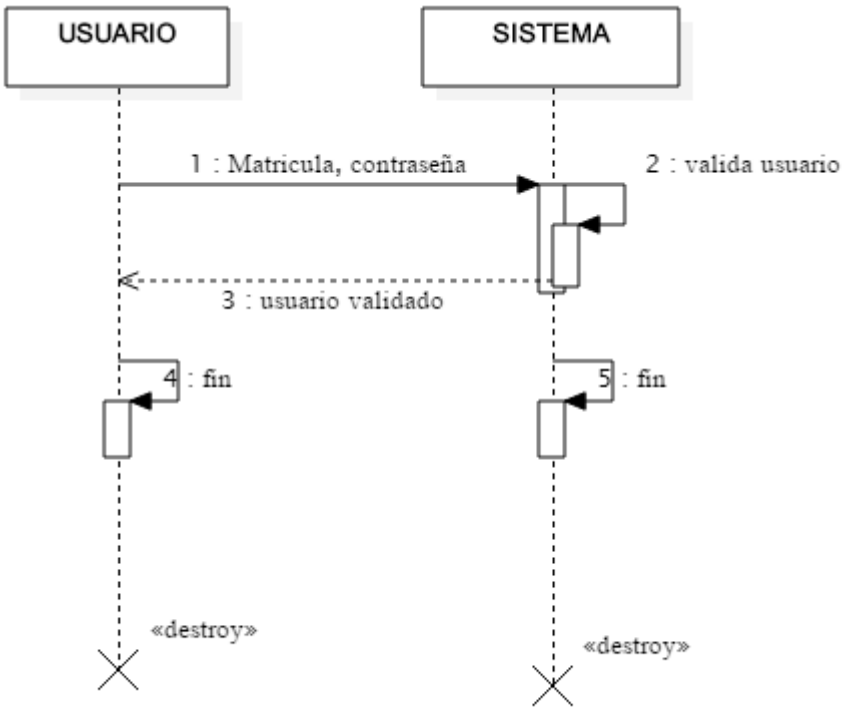
4.13.9. Administrador Alta Usuarios



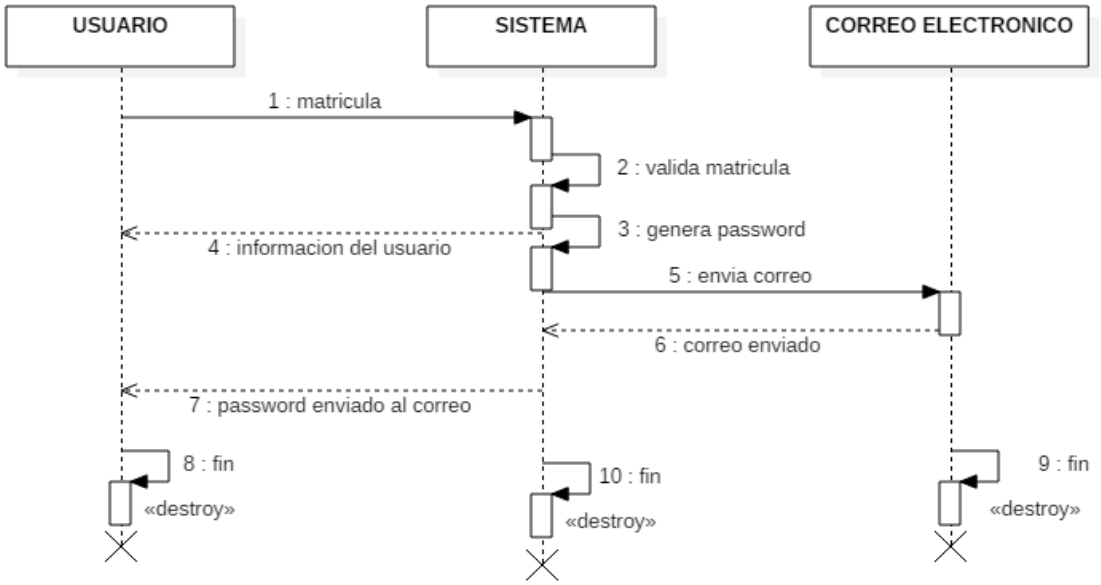
4.15. Diagramas de Secuencia

Muestra la interacción que representa la secuencia de mensajes entre instancias de clases, componentes, subsistemas o actores. El tiempo fluye por el diagrama y muestra el flujo de control de un participante a otro, visualizando instancias y eventos. A continuación se muestran algunos diagramas de secuencia usados.

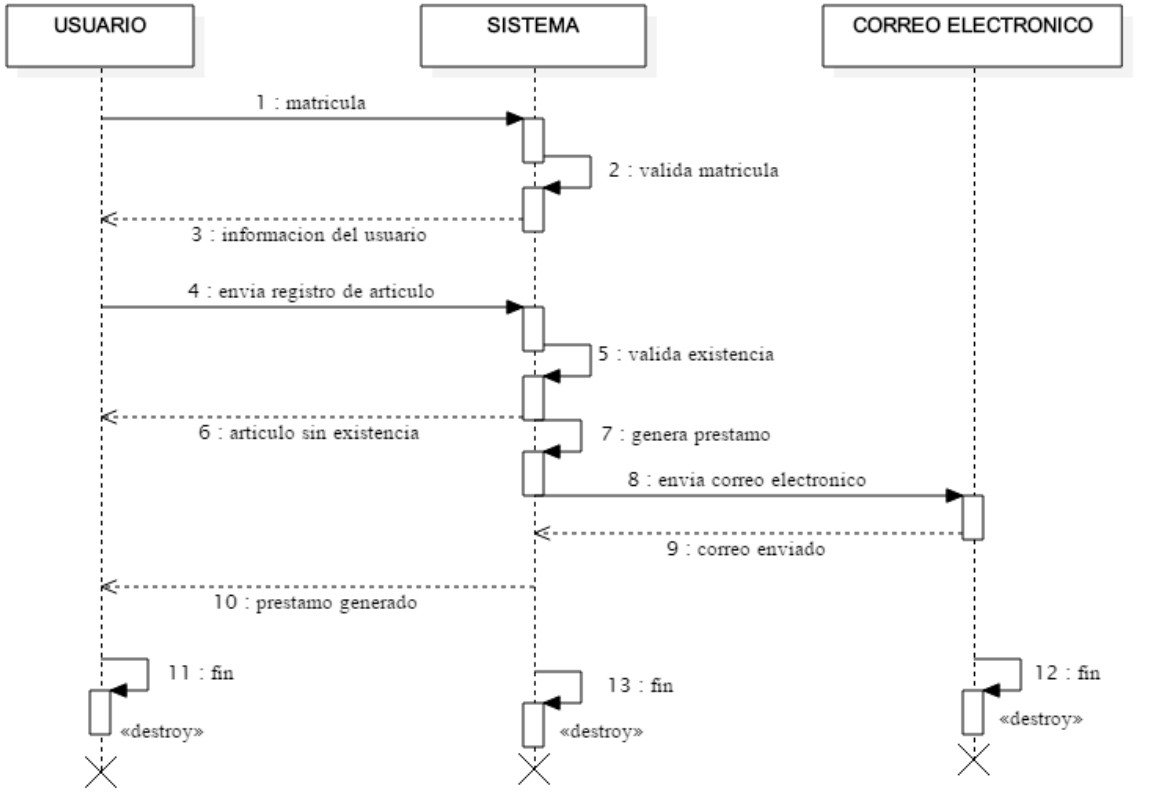
4.15.1. Diagrama de Secuencia Inicio de Sesión



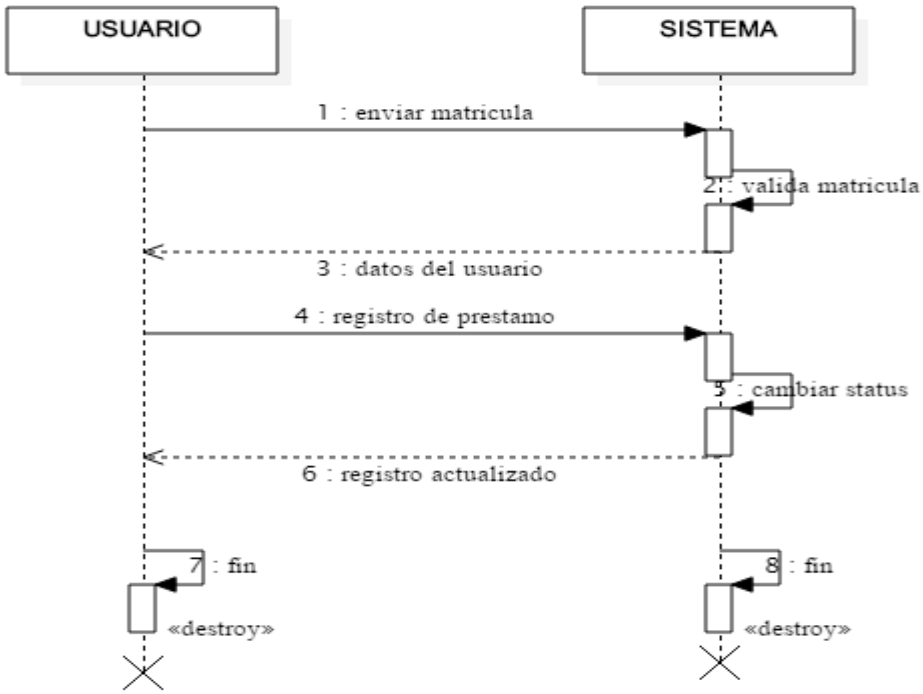
4.15.2. Diagrama de Secuencia Recuperar Password



4.15.3. Diagrama de Secuencia Préstamo



4.15.4. Diagrama de Secuencia Devolución Préstamo



4.16. Diccionario de Datos

TABLA CLASE

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
cla_id	Int		llave primaria del registro
cla_grupo	Int		clave del grupo
cla_asignatura	nchar	100	nombre de la asignatura
cla_fec_inicio	date		fecha inicio de asignatura
cla_fec_fin	date		fecha fin de asignatura
cla_hr_inicio	time	7	hora de inicio de la asignatura
cla_hr_fin	time	7	hora de término de la asignatura
cla_profesor	varchar	100	nombre del profesor de asignatura

TABLA
COMPROBANTE

NTE	TIPO DE DATO	LONGITUD	DESCRIPCION
comp_id	int		llave primaria del registro
comp_fecha	date		fecha del comprobante
comp_estado	vatchar	50	estado del comprobante
comp_fecha_dev	date		fecha de devolución del préstamo
comp_total_prod	int		total de productos
Usu_matricula	int		matricula del usuario

TABLA CURSO

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
cur_id	int		llave primaria del registro
cur_nombre	nchar	100	nombre del curso
cur_fecha	date		fecha del curso
cur_hr_inicio	time	7	hora de inicio del curso
cur_hr_fin	time	7	hora de término del curso

TABLA DETALLE

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
Detalle_id	int		llave primaria del registro
Comp_id	int		llave primaria y foránea que relaciona la tabla COMPROBANTE
Lib_id	int		llave foránea de la tabla LIBRO
Sw_id	int		llave foránea de la tabla SOFTWARE
Hw_id	int		llave foránea de la tabla HARDWARE
Ot_id			llave foránea de la tabla OTRO

TABLA DIPLOMADO

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
dip_id	int		llave primaria del registro
dip_nombre	nchar	100	nombre del diplomado
dip_generacion	nchar	100	Generación

dip_fec_ini	date		fecha de inicio del diplomado
dip_fec_fin	date		fecha de término del diplomado
dip_hr_inicio	time	7	hora de inicio del diplomado
dip_hr_fin	time	7	hora de término del diplomado

TABLA HARDWARE

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
hw_id	int		llave primaria del registro
hw_nombre	varchar	50	nombre del hardware
hw_no_serie	varchar	50	número de serie del hardware
hw_marca	varchar		marca del hardware
hw_cantidad	int		cantidad de artículos
hw_ubicacion	varchar	50	ubicación del hardware

TABLA LIBRO

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
Lib_id	int		llave primaria del registro
Lib_nombre	varchar	50	nombre del libro
Lib_editorial	varchar	50	editorial del libro
Lib_autor	varchar		autor del libro
Lib_cantidad	int		cantidad de libros
Lib_ubicacion	varchar	50	ubicación del libro

TABLA OTRO

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
Ot_id	int		llave primaria del registro
Ot_nombre	varchar	50	nombre del otro
Ot_descripcion	varchar	50	descripción del otro
Ot_cantidad	int		cantidad de otros
Ot_ubicacion	varchar	50	ubicación de otros

TABLA PROYECTO

pro_	TIPO DE DATO	LONGITUD	DESCRIPCION
pro_id	int		llave primaria del registro
pro_dependencia	nchar	100	nombre de la dependencia
pro_objetivo	nchar	100	objetivo del proyecto
pro_nombre	nchar	100	nombre del proyecto
pro_descripcion	nchar	100	descripción del proyecto
pro_fec_ini	date		fecha de inicio del proyecto
pro_fec_fin	date		fecha de término del proyecto

TABLA SOFTWARE

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
sw_id	int		llave primaria del registro
sw_nombre	varchar	50	nombre del software
sw_version	varchar	50	versión del software
sw_arquitectura	varchar		arquitectura del software

sw_cantidad	int		cantidad de software
sw_ubicacion	varchar	50	ubicación del software

TABLA USUARIO

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
usu_matricula	int		llave primaria del registro
usu_nombre	varchar	50	nombre del usuario
usu_appaterno	varchar	50	apellido paterno del usuario
usu_apmaterno	varchar	5	apellido materno del usuario
usu_email	varchar	100	email del usuario
usu_fecalta	date		fecha de alta del usuario
usu_estatus	bit		estatus del usuario
usu_tipo	int		tipo de usuario
usu_password	varbinary	MAX	password del usuario
usu_intentos	int		intentos permitidos para ingresar

TABLA
USUARIO_TIPO

NOMBRE DEL CAMPO	TIPO DE DATO	LONGITUD	DESCRIPCION
usu_tipo_id	int		llave primaria del registro
usu_tipo_descripcion	varchar	50	descripción del tipo de usuario

5 Manual de Usuario

Lo primero que se tiene que hacer es ingresar a la página del Laboratorio de Microsoft Research *microsoft.fi.b.unam.mx/SARLM* (Figura 5.1).

Figura 5.1 Página principal del Laboratorio de Microsoft Research

Seleccionar la opción de "Servicios" y se abre un desplegable con tres opciones: Inicio de Sesión, Productos y Actividades Escolares (Figura 5.2).

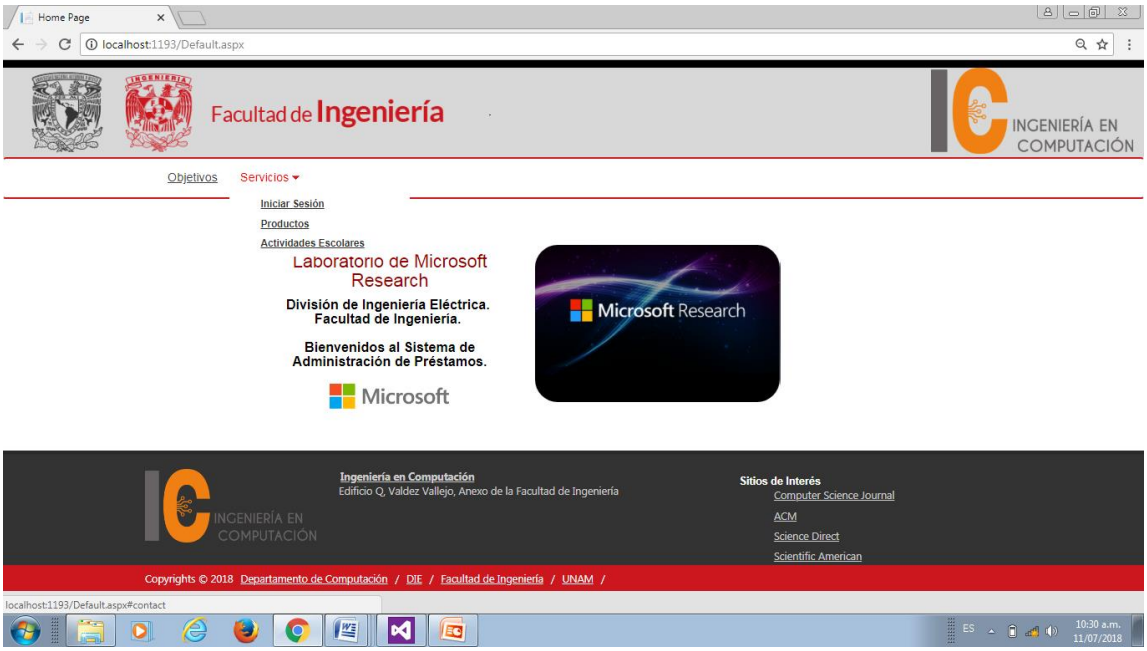


Figura 5.2 Pagina del Sistema de Administración de Prestamos del Laboratorio de Microsoft Research

Seleccionar la opción "Inicio de Sesión" para poder ingresar al sistema ya sea como administrador, alumno o becario (Figura 5.3).
Una vez dentro de la página colocar el numero de usuario y la contraseña asignada.

Figura 5.3 Pagina Inicio de Sesión.

En caso de haber ingresado como administrador del sistema, nos enviara a la página de bienvenida del sistema como administrador en la que se muestra el menú principal con los diferentes desplegables con que cuenta el sistema (Figura 5.4).

Figura 5.4 Pagina de Bienvenido al sistema como Administrador.

En la opción "Objetivos" se muestran los objetivos principales del laboratorio de Microsoft (Figura 5.5).

Figura 4.5 Página Objetivos.

El siguiente desplegable es el de "Servicios" en el cual se muestran los diferentes servicios que posee el sistema. Hay dos tipos de servicios tiene el sistema:

- Actividades escolares
  - Cursos
  - Proyectos
  - Diplomados
  - Clases
- Productos
  - Software
  - Hardware
  - Libros
  - Otros

Al seleccionar la opción de "Productos" nos muestra un buscador de los productos con que cuenta el laboratorio, así como una tabla donde muestra los productos con que cuenta el laboratorio (Figura 5.6).

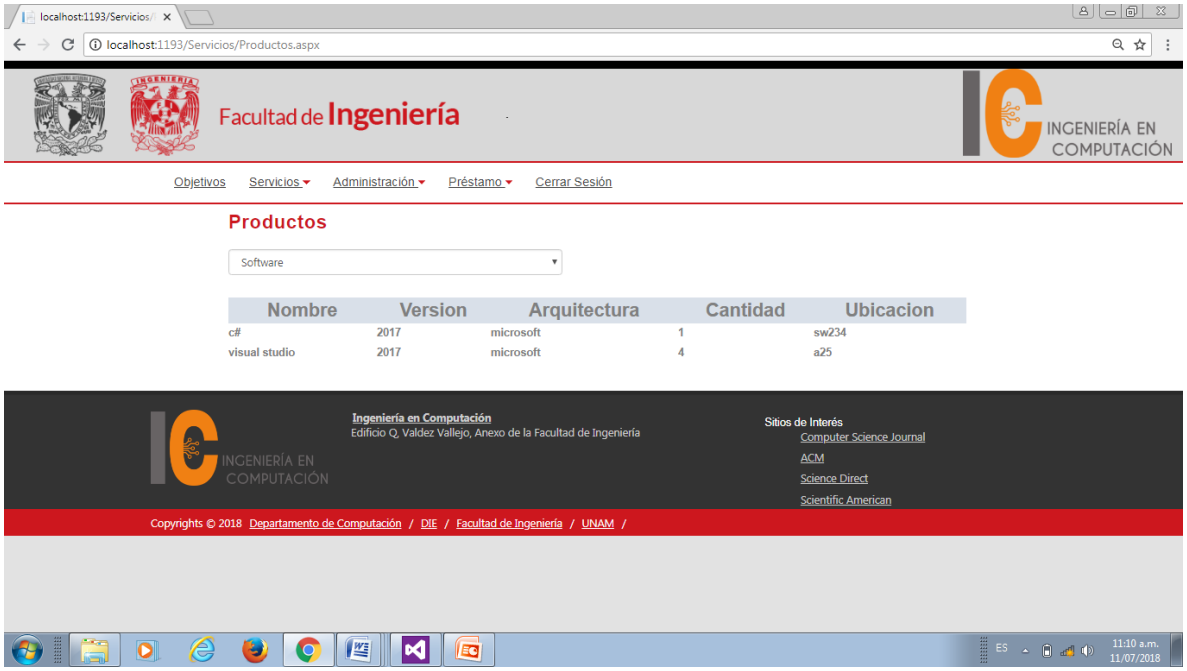


Figura 5.6 Página Productos.

En la opción "Actividades Escolares" podemos seleccionar las diferentes actividades que ofrece el laboratorio. Al seleccionar alguna de ellas nos muestra un calendario donde se muestra la actividad, el horario y los días que se imparte (figura 5.7).

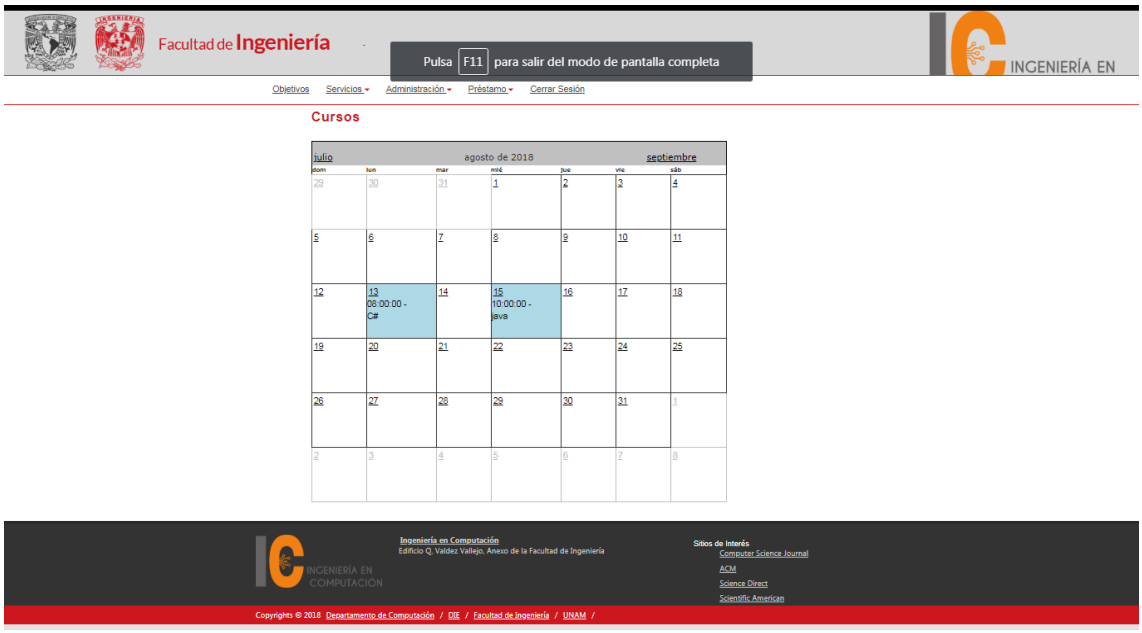


Figura 5.7 Página Cursos.

En el desplegable "Administración" se muestran las siguientes opciones:

- Cambio Contraseña
- Usuarios
- Productos
- Actividades Escolares

Al seleccionar la opción "Cambio Contraseña" nos manda a la página donde se puede cambiar la contraseña del usuario. Es necesario colocar la contraseña actual e ingresar la nueva contraseña, así como confirmar la nueva contraseña. Si los datos son incorrectos nos mandara un mensaje de error, de lo contrario enviara un mensaje en el cual nos indica que la contraseña se ha cambiado se ha creado exitosamente (Figura 5.8).

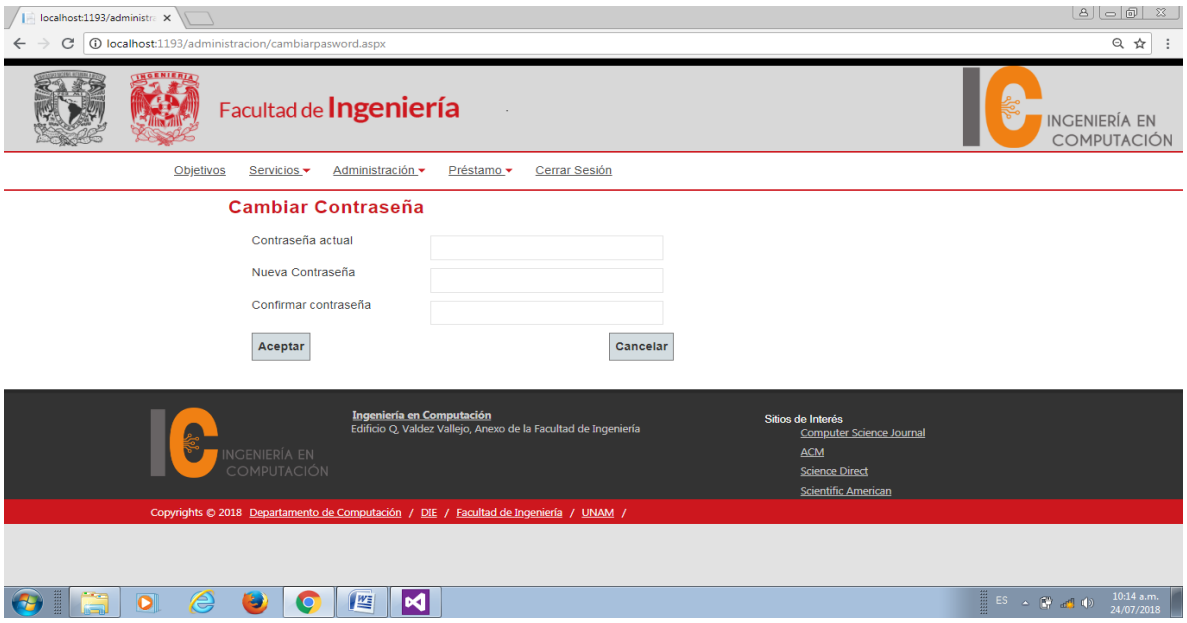


Figura 5.8 Página Cambio Contraseña

En la opción "Usuarios" podemos dar de alta, modificar y eliminar usuarios. También nos muestra un buscador de los usuarios dados de alta en el laboratorio (Figura 5.9).

Figura 5.9 Página Crear Usuario

Cuando se va a agregar un nuevo usuario se debe de dar click en el botón "Aceptar", se desplegara un cuadro el cual será llenado con los datos requeridos indicando el tipo de usuario q se quiera dar de alta (Figura 5.10). Si el usuario es dado de alta correctamente nos enviara in mensaje de "Usuario creado exitosamente" y nos mostrara el nuevo registro. Para modificar los usuarios se tiene forzosamente que seleccionar un solo registro, de lo contrario enviara un mensaje indicando que solo debe de seleccionar un registro, posteriormente dar click en "Modificar" y hacer las modificaciones deseadas. Para eliminar uno o varios registros seleccionar el o los registros y dar click "Eliminar" y los registros serán eliminados del sistema.

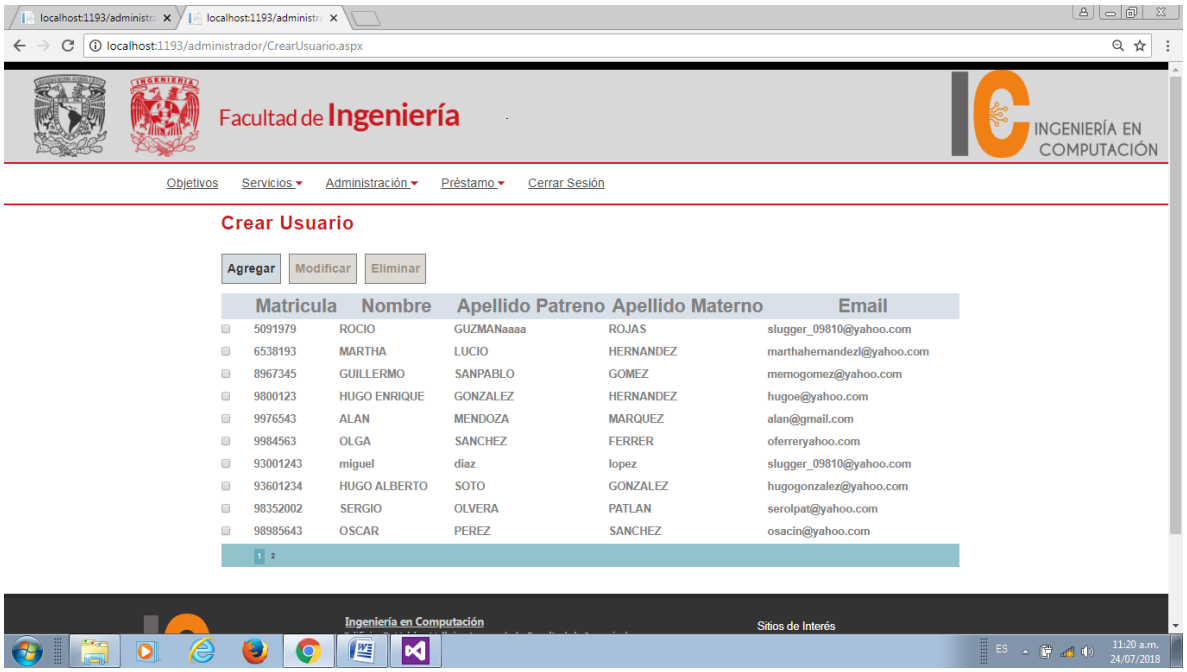


Figura 5.10 Página para agregar un usuario

En la opción "Productos" se puede dar de alta, modificar y eliminar los productos con que cuenta el Laboratorio de Microsoft (Figura 5.11). La página contiene un combo en el cual se puede seleccionar el producto que se desea agregar. Cuando se va a agregar un nuevo producto se debe de dar click en el botón "Aceptar", se desplegara un cuadro el cual será llenado con los datos requeridos. Si el producto es dado de alta correctamente nos enviara in mensaje de "Registro creado exitosamente" y nos mostrara el nuevo registro. Para modificar los productos se tiene forzosamente que seleccionar un solo registro, de lo contrario enviara un mensaje indicando que solo debe de seleccionar un registro, posteriormente dar click en "Modificar" y hacer las modificaciones deseadas. Para eliminar uno o varios registros seleccionar el o los registros y dar click "Eliminar" y los registros serán eliminados del sistema.

Figura 5.11 Página para agregar un producto

En la opción "Actividades Escolares" se puede dar de alta, modificar y eliminar las actividades q imparte el Laboratorio de Microsoft (Figura 5.12) las cuales son: Cursos, Proyectos, Diplomados, Clases y Tesis. Cuando se va a agregar una nueva actividad se debe de dar click en el botón "Agregr", se desplegara un cuadro el cual será llenado con los datos requeridos. Si la actividad es dada de alta correctamente nos enviara in mensaje de "Registro creado exitosamente" y nos mostrara el nuevo registro. Para modificar las actividades se tiene forzosamente que seleccionar un solo registro, de lo contrario enviara un mensaje indicando que solo debe de seleccionar un registro, posteriormente dar click en "Modificar" y hacer las modificaciones deseadas. Para eliminar uno o varios registros seleccionar el o los registros y dar click "Eliminar" y los registros serán eliminados del sistema.

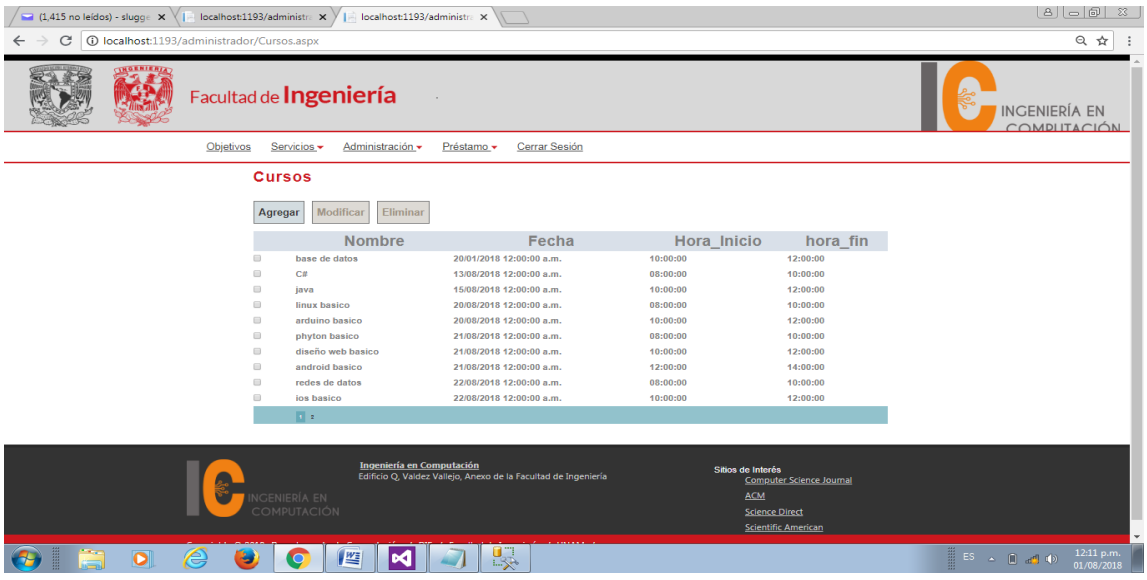


Figura 5.12 Página Curso

En el desplegable "Préstamo" se muestran las siguientes opciones:

- Préstamo
- Devolución
- Historial

Al seleccionar la opción "Préstamo" sirve para solicita el préstamo de alguno de los artículos que tiene a su disposición el laboratorio. Se tiene que poner la matricula del alumno o académico el cual tiene que haber sido dado de alta y presionar el botón buscar para localizar la matricula en la base de datos. Una vez que se haya encontrado la matricula, el sistema arrojara los datos del alumno o académico q este solicitando el préstamo (Figura 5.13). A continuación seleccionar el artículo que se desea solicitar y presionar el botón "Buscar", el sistema arrojara todos opciones que se tengan para elegir, presionar el botón "Préstamo" cuando se tenga la opción deseada. Si el préstamo es correcto nos enviara el mensaje “Registro creado exitosamente".

Figura 5.13 Página Préstamo

Para poder hacer la devolución de un artículo seleccionamos la opción "Devolución", se bebe de ingresar la matrícula del alumno o académico que desea devolver el artículo y buscarla en la base de datos presionando el botón "Buscar". Presionar el botón "Buscar Préstamo" y se desplegara la lista de los artículos que han sido prestados, seleccionar los artículos que han sido prestados y dar click en el botón "Devolver" (Figura 5.14). Cabe aclarar que se deben de devolver todos los artículos que han sido prestados ya que no se podrá prestar más artículos sin que se hayan devuelto en su totalidad todos los q se hayan solicitado. En caso de que se haya omitido seleccionar algún artículo para su devolución se enviara un mensaje con la leyenda "La persona no está devolviendo todos los artículos", de lo contrario enviara el mensaje "Devolución exitosa".

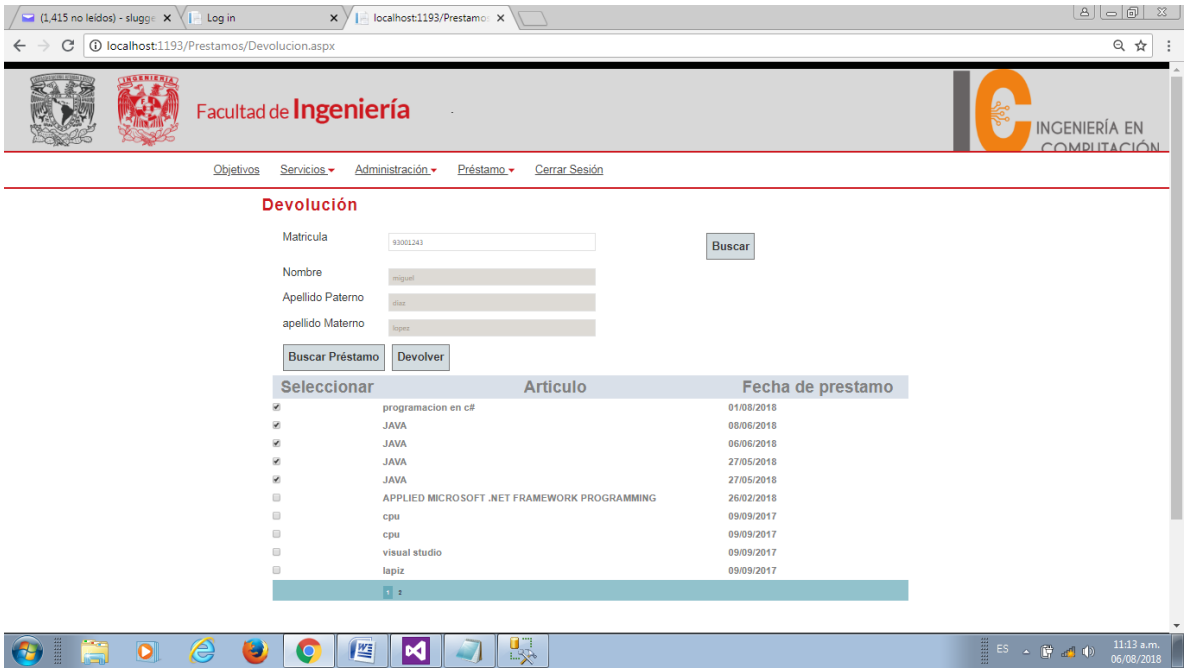


Figura 5.14 Página Devolución

Para consultar el historial de préstamos es necesario seleccionar la opción "Historial ", al igual que en las opciones anteriores se debe de poner la matrícula del alumno o académico y dar click en el botón "Buscar" para que nos despliegue los datos. Seleccionar el botón "Historial" y nos arrojara el historial de préstamos que ha tenido el alumno o académico (Figura 5.15).

Figura 5.15 Página Historial

En el botón "Cerrar sesión" como su nombre lo indica cerrará la sesión del alumno, académico o administrador que se encuentre en ese momento dentro del sistema y nos enviará a la página principal del sistema.

6 Manual de Procesos

Ingresar Password

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using ProyectoTesis.clases;
using ProyectoTesis.TesisSW;

namespace ProyectoTesis.administracion
{
    public partial class cambiar_pasword : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Context.User.Identity.IsAuthenticated == false)
            {
                Response.Redirect("../account/login.aspx");
                return;
            }
            if (Session[constantes.Usuario] == null)
            {
                FormsAuthentication.SignOut();
                Response.Redirect("../account/login.aspx");
                return;
            }
        }

        protected void btnAceptar_Click(object sender, EventArgs e)
        {
             ItesisSWClient con = null;
             try
            {
                var usuario = (clsUsuario)Session[constantes.Usuario];

                con = new ItesisSWClient();
                con.Open();
                bool valida =
con.CambiarPassword(usuario.Usu_matricula,txtContActual.Text,txtNuevaCont.Text);
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('contreseña creada exitosamente')", true);
                    txtContActual.Text = "";
```

```
                    txtNuevaCont.Text = "";
                                    }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se coambio la contgraseña')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if(con!=null)
                    con.Close();
            }
        }

    }
```

Crear Usuario

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using ProyectoTesis.clases;
using ProyectoTesis.TesisSW;

namespace ProyectoTesis.administrador
{
    public partial class CrearUsuario : System.Web.UI.Page
    {
        private const string registros = "registros";

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Context.User.Identity.IsAuthenticated == false)
            {
                Response.Redirect("../account/login.aspx");
                return;
            }
            if (Session[constantes.Usuario] == null)
            {
                FormsAuthentication.SignOut();
                Response.Redirect("../account/login.aspx");
                return;
            }

            if (IsPostBack)
            {
```

```
                return;
            }
            refreshGrid();
            btnEliminar.Enabled = false;
            btnModificar.Enabled = false;
        }

        protected void btnAceptar_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.InsertarUsuarios(int.Parse(txtMatricula.Text),
txtNombre.Text,txtAppaterno.Text,txtApmaterno.Text, txtEmail.Text,
int.Parse(ddlTipo.SelectedValue));
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('usuario creado exitosamente')", true);
                    txtMatricula.Text = "";
                    txtNombre.Text = "";
                    txtAppaterno.Text = "";
                    txtApmaterno.Text = "";
                    txtEmail.Text = "";
                    refreshGrid();
                    btnEliminar.Enabled = false;
                    btnModificar.Enabled = false;
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se creo el usuario')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void chbUsuarios_CheckedChanged(object sender, EventArgs e)
        {
            CheckBox chbaux = (CheckBox)sender;
            GridViewRow grid = ((GridViewRow)((Control)sender).Parent.Parent);
            int id = int.Parse(grUsuarios.DataKeys[grid.RowIndex].Value.ToString());
            List<int> aggId = (List<int>)ViewState[registros];
            if (chbaux.Checked)
            {
                aggId.Add(id);
            }
            else
```

```
                aggId.Remove(id);

            btnEliminar.Enabled = aggId.Count > 0;
            btnModificar.Enabled = aggId.Count > 0;
        }

        private void refreshGrid()
        {
            ViewState[registros] = new List<int>();
            ItesisSWClient conect = new ItesisSWClient();
            conect.Open();
            grUsuarios.DataSource = conect.MostrarDatosUsu();
            grUsuarios.DataBind();
            conect.Close();
        }

        protected void btnEliminar_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool elimina = con.EliminarUsuarios((List<int>)ViewState[registros]);
                con.Close();
                if (elimina == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('usuario eliminado')", true);
                    refreshGrid();
                    btnEliminar.Enabled = false;
                    btnModificar.Enabled = false;
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se elimino el usuario')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void btnAceptar1_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
```

```
                bool valida = con.ModificarUsuarios(int.Parse(txtMatricula1.Text),
txtNombre1.Text, txtAppaterno1.Text, txtApmaterno1.Text, txtEmail1.Text,
int.Parse(ddlTipo1.SelectedValue));
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('usuario modificado exitosamente')", true);
                    txtMatricula1.Text = "";
                    txtNombre1.Text = "";
                    txtAppaterno1.Text = "";
                    txtApmaterno1.Text = "";
                    txtEmail1.Text = "";
                    refreshGrid();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se modifico el usuario')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void grUsuarios_PageIndexChanging(object sender, GridViewPageEventArgs e)
        {
            grUsuarios.PageIndex = e.NewPageIndex;
            refreshGrid();
        }
    }
```

Ingresar Productos

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using ProyectoTesis.clases;
using ProyectoTesis.TesisSW;

namespace ProyectoTesis.administrador
{
```

```
    public partial class Productos : System.Web.UI.Page
    {
        private const string registros = "registros";

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Context.User.Identity.IsAuthenticated == false)
            {
                Response.Redirect("../account/login.aspx");
                return;
            }
            if (Session[constantes.Usuario] == null)
            {
                FormsAuthentication.SignOut();
                Response.Redirect("../account/login.aspx");
                return;
            }

            if (IsPostBack)
            {
                return;
            }

            Panelhardware.Visible = true;
            Panelsoftware.Visible = false;
            Panellibros.Visible = false;
            Panelotros.Visible = false;

            refreshGridH();

        }

        protected void btnAceptarH_Click(object sender, EventArgs e)
        {

            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.InsertarArticulosHw(0, txtNombreH.Text, txtNo_serieH.Text,
txtMarcaH.Text, int.Parse(txtCantidadH.Text), txtUbicacionH.Text);
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro creado exitosamente')", true);
                    txtNombreH.Text = "";
                    txtNo_serieH.Text = "";
                    txtMarcaH.Text = "";
                    txtCantidadH.Text = "";
                    txtUbicacionH.Text = "";
                    refreshGridH();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se creo registro')", true);
                }
```

```
            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }


        }

        protected void chbHardware_CheckedChanged(object sender, EventArgs e)
        {
            CheckBox chbaux = (CheckBox)sender;
            GridViewRow grid = ((GridViewRow)((Control)sender).Parent.Parent);
            int id =
int.Parse(grHardware.DataKeys[grid.RowIndex].Values["Hw_id"].ToString());
            List<int> aggId = (List<int>)ViewState[registros];
            if (chbaux.Checked)
            {
                aggId.Add(id);
            }
            else
                aggId.Remove(id);

        }

        private void refreshGridH()
        {
            ViewState[registros] = new List<int>();
            ItesisSWClient conect = new ItesisSWClient();
            conect.Open();
            grHardware.DataSource = conect.MostrarDatosHw();
            grHardware.DataBind();
            conect.Close();
        }

        protected void btnEliminarH_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool elimina = con.EliminarArticulosHw((List<int>)ViewState[registros]);
                con.Close();
                if (elimina == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro eliminado')", true);
                    refreshGridH();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se elimino registro')", true);
                }
```

```
            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void btnAceptarH1_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.ModificaArticulosHw(((List<int>)ViewState[registros])[0],
txtNombreH1.Text, txtNo_serieH1.Text, txtMarcaH1.Text, int.Parse(txtCantidadH1.Text),
txtUbicacionH1.Text);
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro modificado exitosamente')", true);
                    txtNombreH1.Text = "";
                    txtNo_serieH1.Text = "";
                    txtMarcaH1.Text = "";
                    txtCantidadH1.Text = "";
                    txtUbicacionH1.Text = "";
                    refreshGridH();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se modifico el registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void btnAceptarS_Click(object sender, EventArgs e)
        {

            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
```

```
                con.Open();
                bool valida = con.InsertarArticulosSw(0, txtNombreS.Text, txtVersionS.Text,
txtArquitecturaS.Text, int.Parse(txtCantidadS.Text), txtUbicacionS.Text);
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro creado exitosamente')", true);
                    txtNombreS.Text = "";
                    txtVersionS.Text = "";
                    txtArquitecturaS.Text = "";
                    txtCantidadS.Text = "";
                    txtUbicacionS.Text = "";
                    refreshGridS();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se creo registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }


        }

        protected void chbSoftware_CheckedChanged(object sender, EventArgs e)
        {
            CheckBox chbaux = (CheckBox)sender;
            GridViewRow grid = ((GridViewRow)((Control)sender).Parent.Parent);
            int id =
int.Parse(grSoftware.DataKeys[grid.RowIndex].Values["Sw_id"].ToString());
            List<int> aggId = (List<int>)ViewState[registros];
            if (chbaux.Checked)
            {
                aggId.Add(id);
            }
            else
                aggId.Remove(id);

        }

        private void refreshGridS()
        {
            ViewState[registros] = new List<int>();
            ItesisSWClient conect = new ItesisSWClient();
            conect.Open();
            grSoftware.DataSource = conect.MostrarDatosSw();
            grSoftware.DataBind();
            conect.Close();
        }
```

```
        protected void btnEliminarS_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool elimina = con.EliminarArticulosSw((List<int>)ViewState[registros]);
                con.Close();
                if (elimina == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro eliminado')", true);
                    refreshGridS();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se elimino registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void btnAceptarS1_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.ModificaArticulosSw(((List<int>)ViewState[registros])[0],
txtNombreS1.Text, txtVersionS1.Text, txtArquitecturaS1.Text, int.Parse(txtCantidadS1.Text),
txtUbicacionS1.Text);
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro modificado exitosamente')", true);
                    txtNombreS1.Text = "";
                    txtVersionS1.Text = "";
                    txtArquitecturaS1.Text = "";
                    txtCantidadS1.Text = "";
                    txtUbicacionS1.Text = "";
                    refreshGridS();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se modifico el registro')", true);
```

```
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void ddlArticulos_SelectedIndexChanged(object sender, EventArgs e)
        {
            Panelhardware.Visible = false;
            Panelsoftware.Visible = false;
            Panellibros.Visible = false;
            Panelotros.Visible = false;

            if (ddlArticulos.SelectedIndex == 0)
            {
                Panelhardware.Visible = true;
                refreshGridH();
            }

            if (ddlArticulos.SelectedIndex == 1)
            {
                Panelsoftware.Visible = true;
                refreshGridS();
            }

            if (ddlArticulos.SelectedIndex == 2)
            {
                Panellibros.Visible = true;
                refreshGridL();
            }

            if (ddlArticulos.SelectedIndex == 3)
            {
                Panelotros.Visible = true;
                refreshGridO();
            }
        }


        protected void btnAceptarL_Click(object sender, EventArgs e)
        {

            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.InsertarArticulosLib(0, txtNombreL.Text,
txtEditorialL.Text, txtAutorL.Text, int.Parse(txtCantidadL.Text), txtUbicacionL.Text);
                con.Close();
                if (valida == true)
                {
```

```
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro creado exitosamente')", true);
                    txtNombreL.Text = "";
                    txtEditorialL.Text = "";
                    txtAutorL.Text = "";
                    txtCantidadL.Text = "";
                    txtUbicacionL.Text = "";
                    refreshGridL();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se creo registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }


        }

        protected void chbLibros_CheckedChanged(object sender, EventArgs e)
        {
            CheckBox chbaux = (CheckBox)sender;
            GridViewRow grid = ((GridViewRow)((Control)sender).Parent.Parent);
            int id = int.Parse(grLibro.DataKeys[grid.RowIndex].Values["Lib_id"].ToString());
            List<int> aggId = (List<int>)ViewState[registros];
            if (chbaux.Checked)
            {
                aggId.Add(id);
            }
            else
                aggId.Remove(id);

        }

        private void refreshGridL()
        {
            ViewState[registros] = new List<int>();
            ItesisSWClient conect = new ItesisSWClient();
            conect.Open();
            grLibro.DataSource = conect.MostrarDatosLib();
            grLibro.DataBind();
            conect.Close();
        }

        protected void btnEliminarL_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
```

```
                con.Open();
                bool elimina = con.EliminarArticulosLib((List<int>)ViewState[registros]);
                con.Close();
                if (elimina == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro eliminado')", true);
                    refreshGridL();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se elimino registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void btnAceptarL1_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.ModificaArticulosLib(((List<int>)ViewState[registros])[0],
txtNombreL1.Text, txtEditorialL1.Text, txtAutorL1.Text, int.Parse(txtCantidadL1.Text),
txtUbicacionL1.Text);
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro modificado exitosamente')", true);
                    txtNombreL1.Text = "";
                    txtEditorialL1.Text = "";
                    txtAutorL1.Text = "";
                    txtCantidadL1.Text = "";
                    txtUbicacionL1.Text = "";
                    refreshGridL();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se modifico el registro')", true);
                }


            }
            catch
            {
```

```
            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }


        protected void btnAceptarO_Click(object sender, EventArgs e)
        {

            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.InsertarArticulos(0, txtNombreO.Text, txtDescripcionO.Text,
int.Parse(txtCantidadO.Text), txtUbicacionO.Text);
                con.Close();
                if (valida == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro creado exitosamente')", true);
                    txtNombreO.Text = "";
                    txtDescripcionO.Text = "";
                    txtCantidadO.Text = "";
                    txtUbicacionO.Text = "";
                    refreshGridO();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se creo registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }


        }

        protected void chbOtros_CheckedChanged(object sender, EventArgs e)
        {
            CheckBox chbaux = (CheckBox)sender;
            GridViewRow grid = ((GridViewRow)((Control)sender).Parent.Parent);
            int id = int.Parse(grOtros.DataKeys[grid.RowIndex].Values["Ot_id"].ToString());
            List<int> aggId = (List<int>)ViewState[registros];
            if (chbaux.Checked)
            {
                aggId.Add(id);
            }
```

```
            else
                aggId.Remove(id);

        }

        private void refreshGridO()
        {
            ViewState[registros] = new List<int>();
            ItesisSWClient conect = new ItesisSWClient();
            conect.Open();
            grOtros.DataSource = conect.MostrarDatos();
            grOtros.DataBind();
            conect.Close();
        }

        protected void btnEliminarO_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool elimina = con.EliminarArticulos((List<int>)ViewState[registros]);
                con.Close();
                if (elimina == true)
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro eliminado')", true);
                    refreshGridO();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se elimino registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void btnAceptarO1_Click(object sender, EventArgs e)
        {
            ItesisSWClient con = null;
            try
            {
                con = new ItesisSWClient();
                con.Open();
                bool valida = con.ModificaArticulos(((List<int>)ViewState[registros])[0],
txtNombreO1.Text, txtDescripcionO1.Text, int.Parse(txtCantidadO1.Text), txtUbicacionO1.Text);
                con.Close();
                if (valida == true)
                {
```

```
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('registro modificado exitosamente')", true);
                    txtNombreO1.Text = "";
                    txtDescripcionO1.Text = "";
                    txtCantidadO1.Text = "";
                    txtUbicacionO1.Text = "";
                    refreshGridO();
                }
                else
                {
                    ScriptManager.RegisterStartupScript(this, GetType(), "mensaje",
"mensaje('no se modifico el registro')", true);
                }


            }
            catch
            {

            }
            finally
            {
                if (con != null)
                    con.Close();
            }
        }

        protected void grOtros_PageIndexChanging(object sender, GridViewPageEventArgs e)
        {
            grOtros.PageIndex = e.NewPageIndex;
            refreshGridO();
        }

        protected void grLibro_PageIndexChanging(object sender, GridViewPageEventArgs e)
        {
            grLibro.PageIndex = e.NewPageIndex;
            refreshGridL();
        }

        protected void grSoftware_PageIndexChanging(object sender, GridViewPageEventArgs e)
        {
            grSoftware.PageIndex = e.NewPageIndex;
            refreshGridS();
        }

        protected void grHardware_PageIndexChanging(object sender, GridViewPageEventArgs e)
        {
            grHardware.PageIndex = e.NewPageIndex;
            refreshGridH();
        }

    }
```

7 Conclusiones

Los conocimientos que he aplicado en esta tesis los obtuve a lo largo de mi formación universitaria, en las asignaturas de base de datos, ingeniería de software, computadoras y programación, sistemas operativos e ingeniería de programación.

Al realizar este sistema me di cuenta que es muy importante entender claramente las necesidades del usuario final ya que este será quien utilice el sistema.

Este trabajo me permitió aplicar conocimientos adquiridos durante la carrera, la formación obtenida nos hace ser más analíticos y por consiguiente tratar de siempre obtener los mayores y mejores resultados cuando los aplicamos en nuestras actividades personales y profesionales.

En la actualidad existe un gran número de herramientas de las cuales podemos hacer uso, con las cuales podemos implementar alguna aplicación que optimice nuestras actividades laborales, muchas de ellas son de uso libre y algunas tienen costo pero finalmente podemos adquirir y conocer dichas herramientas mediante internet.

El sistema se realizó utilizando una metodología scrum con el cual logramos tener una buena comunicación con el cliente (Laboratorio de Microsoft), un progreso continuo y lo principal satisfacer sus necesidades.

El entorno de desarrollo utilizado fue Visual Studio 2012 ya que nos permite crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET, se programo en lenguaje c# ya que es uno de los más utilizados y en la parte web nos apoyamos La base de datos utilizada fue Microsoft SQL Server ya que nos permitió manejar y obtener datos de la red,

Se utilizó una programación en capas, en la cual se separaron las partes que conforman el software en la lógica de negocios, la capa de presentación y en la capa de datos.

Para la lógica de negocios se creó un proyecto llamado "Tesis Web" el cual es un servicio web de tipo WCF. Aquí se conectan los datos con la interfaz gráfica, validando la información que entra de la web. En la capa de presentación se creó un proyecto llamado "Proyecto tesis" ahí se tienen todos loa aspx, los scripts y las hojas de estilo. Y la tercera capa la de datos donde tenemos el proyecto "Tesis datos" la cual nos ayudo a la comunicación entre el Entyty Framework y la base de datos.

El sistema cumplió con los objetivos ya que nos proporciona:

- Una interfaz de fácil manejo
- Agiliza el registro de usuarios así como el alta y baja de material
- Lleva a cabo el inventario del material del laboratorio
- Proporciona varios niveles de seguridad de acuerdo al perfil.

- La información siempre está disponible, segura y protegida.

Bibliografía

[1] Ingeniería de Software: Un Enfoque Práctico, Roger Pressman, Sexta Edición.

[2] Ingenieria del Software, Ian Sommerville, Séptima Edición.

[3] PHP Y MySQL: Tecnología para el desarrollo de aplicaciones web. Angel Cobo.

[4] Pfleeger S. "Ingeniería de Software, Teoría y Práctica" - Primera Edición Editorial Prentice Hall - 2002.

[5] Diseño de Base de Datos Relacionales. Ed. Alfaomega.

[6] Guía de Aprendizaje: MYSQL Larry Ullman (Pearson Education) Idioma: Español 352 páginas.

[7] Desarrollo de aplicaciones C con Visual Studio .NET. Curso Practica, Borja Obregón Arana, Ed. Alfaomega, 2015.

[8] https://proyectosagiles.org

[9] https://msdn.microsoft.com/es-es/library/bb399567(v=vs.110).aspx

[10] https://www.asp.net/learn

[11] https://www.w3schools.com/js/

[12] https:// www.w3schools.com/css/

[13] https://es.slideshare.net/Decimo/arquitectura-3capas