BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA WEB: GLOSARIO DE TÉRMINOS COMPUTACIONALES PARA LA MATERIA DE INTRODUCCIÓN A LA DISCIPLINA COMPUTACIONAL.

TESIS

QUE PARA OBTENER EL TÍTULO DE LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
LIDICE SHACCELL SÁNCHEZ HERNÁNDEZ.

ASESORA:
DRA. MARÍA JOSEFA SOMODEVILLA GARCÍA.

PUEBLA, PUEBLA 2008

RESUMEN

Este proyecto tiene como finalidad la creación de un Sistema Web para la materia de Introducción a la Disciplina Computacional el cual servirá a los usuarios para tener las definiciones de algunos de los conceptos computacionales fundamentales de esta disciplina así como incluir ligas a dichos conceptos.

Para la creación de este proyecto se utilizará una metodología, la cual nos permitirá tener resultados satisfactorios. La metodología que utilizaremos está basada en el Ciclo de Vida de Cascada el cual presenta las siguientes fases: Requerimientos, Análisis, Diseño, Implementación y Pruebas.

Análisis de requerimientos nos servirá para identificar de manera correcta las diferentes funciones que realizará el Sistema. Para una mejor comprensión de estas funciones utilizamos una herramienta basada en UML la cual nos permitirá hacer diagramas para así poder representar de una manera más sencilla estas funciones.

En la fase del diseño se desarrollará una base de datos basada en el modelo Entidad-Relación, el cual facilitará la comprensión de todos los elementos de la base de datos como lo son las entidades, los atributos y las relaciones que existen entre ellos.

Para la fase de la implementación del sistema se utilizarán diferentes herramientas como: MySql que nos servirá para la manipulación de la Base de Datos y PHP como software de capa media.

Después de este proceso se procederán a realizar las pruebas pertinentes para comprobar que el Sistema cumple con las funciones antes especificadas de una manera correcta y eficiente.

INTRODUCCIÓN

La computación es una ciencia que está en constante evolución y que abarca diferentes ámbitos, por lo tanto, para las personas que desean estudiar esta ciencia es necesario tener perfiles, los cuales les ayudarán a saber a que parte de esta ciencia se van a enfocar.

Además de los perfiles es necesario que todo individuo que desee introducirse al mundo de la computación cuente con un mínimo de conocimientos básicos de los términos que se utilizan en esta área, los cuales sean conceptos bien definidos, ya que estos ayudan a entender mejor la computación. Aunque actualmente en internet se encuentra un sin fin de términos con sus correspondientes definiciones, aún no hay ningún glosario de términos enfocado a la Introducción de la Disciplina Computacional.

Es por eso que este proyecto se dedicará a satisfacer esta necesidad, teniendo como finalidad que los usuarios cuenten con conceptos correctos, básicos y fundamentales de la Disciplina Computacional. Para esto se realizará un sistema que reúna toda la información sobre los conceptos computacionales de la materia de Introducción a la Disciplina Computacional y pueda manejarse a través de una interfaz sencilla, agradable y entendible para cualquier usuario.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar un sistema Web que permita al usuario adquirir conocimientos básicos de términos computacionales utilizados en la materia de Introducción a la Disciplina Computacional.

OBJETIVOS PARTICULARES

Permitir al usuario consultar mediante la introducción de un término o palabra teniendo como resultado el significado de estas ó ligas con direcciones electrónicas en donde podrá haces más extenso el estudio del término o palabra introducida.

Permitir al usuario explorar el sistema a través de las interfaces creadas para este y poder realizar las tareas de búsqueda, eliminación e inserción de términos en la base de datos.

ESTRUCTURA DEL DOCUMENTO

CAPÍTULO 1

El capítulo de Marco Teórico abarca fundamentalmente 4 temas; la historia de la Computación, Ingeniería de Software, Bases de Datos y Herramientas para la creación del proyecto.

CAPÍTULO 2

En el capítulo llamado Análisis del Sistema, encontramos 3 temas: el primero es una introducción sobre la materia de Introducción a la Disciplina Computacional, el segundo tema abarca el planteamiento del problema y también los requisitos del sistema y por ultimo encontramos los casos de uso relacionados al sistema.

CAPÍTULO 3

El capítulo denominado Diseño del Sistema conceptual, lógico y físico, trata de la creación de la base de datos para el sistema, por lo tanto se presentan las entidades, el diagrama Entidad – Relación, el proceso del mapeo al modelo relacional, así como también la normalización de la base de datos asegurando así un correcto funcionamiento.

CAPÍTULO 4

En este capítulo titulado Implantación y Pruebas encontraremos: las herramientas empleadas y finalmente se muestran las diferentes interfaces terminadas con ejemplos de su uso.

Finalmente, se presentan las conclusiones y perspectivas, seguidas de la bibliografía.

1. MARCO TEÓRICO

En este capítulo se abordarán los fundamentos teóricos que se tomarán como base para el desarrollo del trabajo de tesis.

Dadas las características del sistema a desarrollar que implementa una herramienta de apoyo al curso “Introducción a la Disciplina Computacional”. El marco teórico incluye conceptos que ayudarán al estudiante a la comprensión y aprovechamiento del curso.

1.1. ANTECEDENTES

1.1.1. GLOSARIOS

Un *Glosario* puede definirse como:

Repertorio de términos pertenecientes a un área del conocimiento, disciplina o ámbito, añadiendo por lo general las definiciones o explicaciones necesarias para su comprensión.

Lista de palabras poco conocidas o desusadas de una obra, junto con su definición [1].Por lo tanto podemos concluir que pueden existir diferentes tipos de glosarios.

1.1.1.1. GLOSARIO DE TÉRMINOS COMPUTACIONALES

Existen varios glosarios de términos computacionales los cuales se enfocan generalmente en los términos básicos de la computación sin profundizar en las diferentes ramas. Por lo tanto aunque son de gran ayuda en términos generales, no son la mejor opción cuando se refiere a que los usuarios tengan una idea de lo que se va a estudiar en los diferentes perfiles en los que se divide la Computación.

1.1.2. COMPUTACIÓN

La Computación se define como el conjunto de conocimientos, métodos, técnicas y habilidades necesarias para comprender, analizar, modelar y actuar sobre la complejidad del mundo en forma eficiente, sistemática y automatizable mediante los equipos de cómputo [2].

No podemos hablar de la Computación sin hablar de las *Computadora* por lo tanto la definiremos como: máquina capaz de efectuar una secuencia de operaciones mediante un programa, de tal manera, que se realice un procesamiento sobre un conjunto de datos de entrada, obteniéndose otro conjunto de datos de salida.

1.2. INGENIERÍA DE SOFTWARE

Desde que se acuñara el término de Ingeniería del Software en la conferencia de la OTAN de 1968 [6], se ha recorrido un largo camino en el que se han ido incorporando nuevos campos de investigación y desarrollo bajo el amparo del término general, que cubren de una manera excepcional cada una de las fases que componen el ciclo de vida del software.

Se encuentran diferentes definiciones de Ingeniería de Software a continuación citaremos algunas:

Ingeniería del software es el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales. **Frizt Bauer, 1968**.

Tratamiento sistemático de todas las fases del ciclo de vida del software. Se refiere a la aplicación de metodologías para el desarrollo del sistema software. **AECC, 1986**.

Construcción de software multi-versión por un equipo de varias personas. **David Parnas, 1987** [7].

En la actualidad podemos decir que la **Ingeniería de Software** designa el conjunto de técnicas destinadas a la producción de un programa de computadora, más allá de la sola actividad de programación. Forman parte de

esta disciplina las ciencias computacionales y el manejo de proyectos, entre otros campos, propios de la rama más genérica denominada Ingeniería informática [8].

1.2.1. PROCESO DEL SOFTWARE

El proceso del software es el modo en que programamos. Comienza con la exploración del concepto y termina cuando por último se entrega el producto. Durante este tiempo, el producto recorre una serie de pasos, como los requisitos, el análisis (especificación), diseño, implementación, integración, mantenimiento pos entrega y finalmente el retiro.**[6]**

1.3. MODELO DEL CICLO DE VIDA

Un Modelo Del Ciclo de Vida es una descripción de los pasos que deben realizarse cuando se construye un producto de Software [6]. Debido a que casi siempre es más fácil llevar a cabo una secuencia de tareas más pequeña que una grande, el modelo del ciclo de vida total se divide en una serie de pasos más pequeños, denominados *fases*. [9]

1.3.1. MODELO DE CASCADA

Llamado algunas veces «ciclo de vida básico» o «*modelo lineal secuencial*» es el paradigma más antiguo y más extensamente utilizado en la ingeniería del software. Sin embargo, la crítica del paradigma ha puesto en duda su eficacia

Sugiere un enfoque secuencial para el desarrollo del Software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento.

El Modelo de Cascada que se muestra gráficamente en la Fig. 1.1 comprende las siguientes actividades:

Análisis de los requisitos del software. El proceso de reunión de requisitos se intensifica y se centra especialmente en el software. Para comprender la naturaleza del (los) programa(s) a construirse, el ingeniero («analista») del software debe comprender el dominio de información del software**,** así como la función requerida, comportamiento, rendimiento e interconexión.

Diseño. El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos de programa: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo). El proceso del diseño traduce requisitos en una representación del software donde se pueda evaluar su calidad antes de que comience la codificación.

Generación de código. El diseño se debe traducir en una forma legible por la máquina. El paso de generación de código lleva a cabo esta tarea. Si se lleva a cabo el diseño de una forma detallada, la generación de código se realiza mecánicamente.

Pruebas. Una vez que se ha generado el código, comienzan las pruebas del programa. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

Mantenimiento. El software indudablemente sufrirá cambios después de ser entregado al cliente (una excepción posible es el software empotrado). Se producirán cambios porque se han encontrado errores, porque el software debe adaptarse para acoplarse a los cambios de su entorno externo (por ejemplo: se requiere un cambio debido a un sistema operativo o dispositivo periférico nuevo), o porque el cliente requiere mejoras funcionales o de rendimiento. El soporte y mantenimiento del software vuelve a aplicar cada una de las fases precedentes a un programa ya existente y no a uno nuevo. [9]

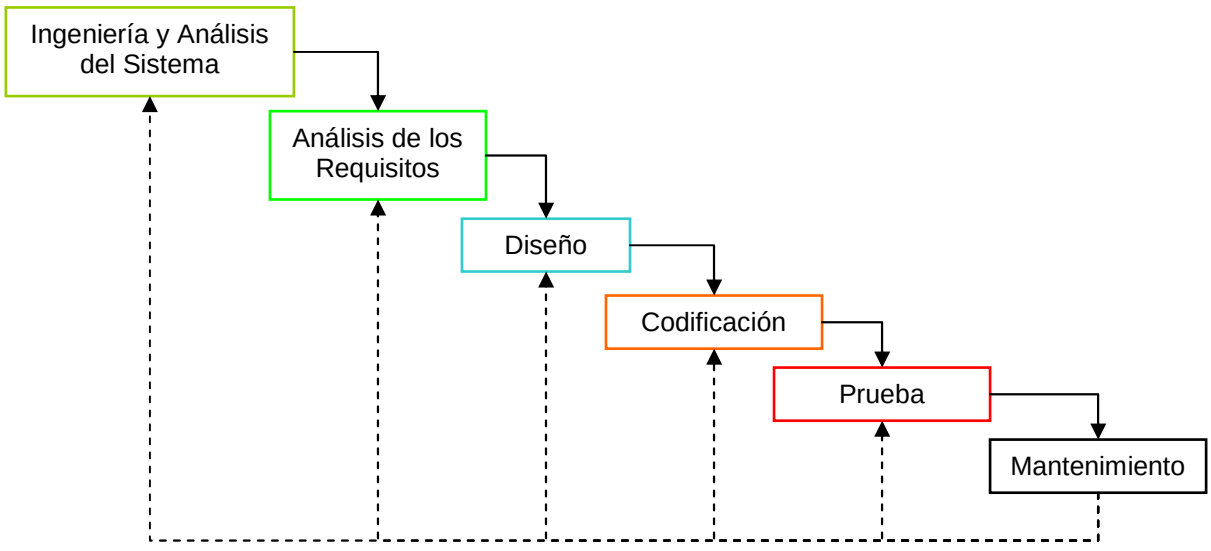


Figura 1.1 Modelo de Cascada.

1.3.2. MODELO DE CONSTRUCCIÓN DE PROTOTIPOS

El paradigma de construcción de prototipos como se muestra en la Figura 1.2 comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un «diseño rápido». El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente (por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

Lo ideal sería que el prototipo sirviera como un mecanismo para identificar los requisitos del software. Si se construye un prototipo de trabajo, el desarrollador intenta hacer uso de los fragmentos del programa ya existentes o aplica herramientas (por ejemplo: generadores de informes, gestores de ventanas, etc.) que permiten generar rápidamente programas de trabajo.

En la mayoría de los proyectos, el primer sistema construido apenas se puede utilizar. Puede ser demasiado lento, demasiado grande *o* torpe en su uso, *o* las tres a la vez. No hay otra alternativa que comenzar de nuevo, aunque nos duela pero es más inteligente, y construir una versión rediseñada en la que se resuelvan estos problemas. Cuando se utiliza un concepto nuevo de sistema *o* una tecnología nueva, se tiene que construir un sistema que no sirva y se tenga que tirar, porque incluso la mejor planificación no es omnisciente como para que esté perfecta la primera vez. Por lo tanto la pregunta de la gestión no es si construir un sistema piloto y tirarlo. Tendremos que hacerlo. La única pregunta es si planificar de antemano construir un desechable, *o* prometer entregárselo a los clientes.

El prototipo puede servir como «primer sistema». El que Brooks recomienda tirar. Aunque esta puede ser una visión idealizada. Es verdad que a los clientes y a los que desarrollan les gusta el paradigma de construcción de prototipos. A los usuarios les gusta el sistema real y a los que desarrollan les gusta construir algo inmediatamente. [9]

Figura 1.2 Modelo de Construcción de Prototipos.

1.3.3. EL MODELO DRA

El *Desarrollo Rápido de Aplicaciones* ***(DRA)*** como se muestra en la Figura 1.3 es un modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. El modelo DRA es una adaptación a «alta velocidad» del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un «sistema completamente funcional» dentro de períodos cortos de tiempo (por ejemplo: de ***60*** a 90 días). Cuando se utiliza principalmente para aplicaciones de sistemas de información, el enfoque DRA comprende las siguientes fases:

Modelado de Gestión. El flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la procesa?

Modelado de datos. El flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.

Modelado del proceso. Los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos.

Generación de aplicaciones. El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas para facilitar la construcción del software.

Pruebas y entrega. Como el proceso **DRA** enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo.

Si una aplicación de gestión puede modularse de forma que permita completarse cada una de las funciones principales en menos de tres meses (utilizando el enfoque descrito anteriormente), es un candidato del DRA. Cada una de las funciones pueden ser afrontadas por un equipo **DRA** separado y ser integradas en un solo conjunto. Al igual que todos los modelos de proceso, el enfoque **DRA** tiene inconvenientes.

- Para proyectos grandes aunque por escalas, el **DRA** requiere recursos humanos suficientes como para crear el número correcto de equipos **DRA.**

- **DRA** requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado. Si no hay compromiso por ninguna de las partes constituyentes, los proyectos **DRA** fracasarán.

- No todos los tipos de aplicaciones son apropiados para **DRA.** Si un sistema no se puede modularizar adecuadamente, la construcción de los componentes necesarios para **DRA** será problemático. **Si** está en juego el alto rendimiento, y se va a conseguir el rendimiento convirtiendo interfaces en componentes de sistemas, el enfoque **DRA** puede que no funcione.

- **DRA** no es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando el software nuevo requiere un alto grado de interoperatividad con programas de computadora ya existentes. [9]

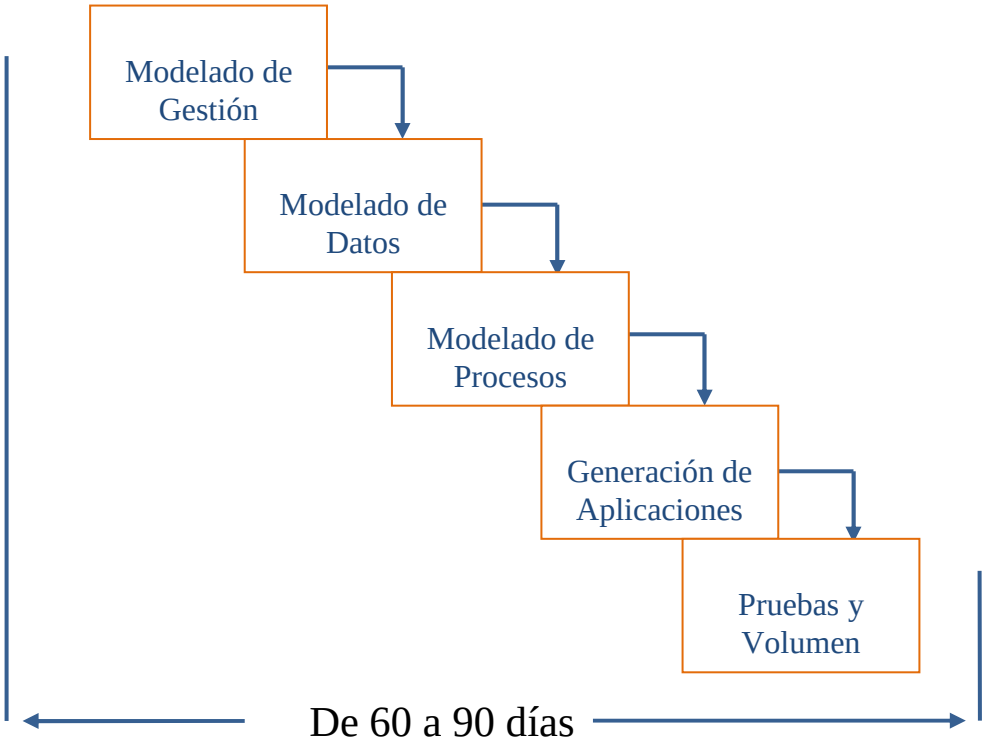


Figura 1.3 Modelo DRA.

1.3.4. MODELO INCREMENTAL

El *modelo incremental* como se muestra en la Figura 1.4 combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. El modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un «incremento» del software. Por ejemplo, el software de tratamiento de textos desarrollado con el paradigma incremental podría extraer funciones de gestión de archivos básicos y de producción de documentos en el primer incremento; funciones de edición más sofisticadas y de producción de documentos en el segundo incremento; corrección ortográfica y gramatical en el tercero; y una función avanzada de esquema de página en el cuarto. Se debería tener en cuenta que el flujo del proceso de cualquier incremento puede incorporar el paradigma de construcción de prototipos.

Cuando se utiliza un modelo incremental, el primer incremento a menudo es un producto esencial. Es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias (algunas conocidas, otras no) quedan sin extraer. El

cliente utiliza el producto central (o sufre la revisión detallada). Como un resultado de utilización y/o de evaluación, se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto central a fin de cumplir mejor las necesidades del cliente y la entrega de funciones, y características adicionales. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo.

El modelo de proceso incremental, como la construcción de prototipos y otros enfoques evolutivos, es iterativo por naturaleza. Pero a diferencia de la construcción de prototipos, el modelo incremental se centra en la entrega de un producto operacional con cada incremento. Los primeros incrementos son versiones «incompletas» del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

El desarrollo incremental es particularmente útil cuando la dotación de personal no está disponible para una implementación completa en la fecha límite que se haya establecido para el proyecto. Los primeros incrementos se pueden implementar con menos personas. [9]

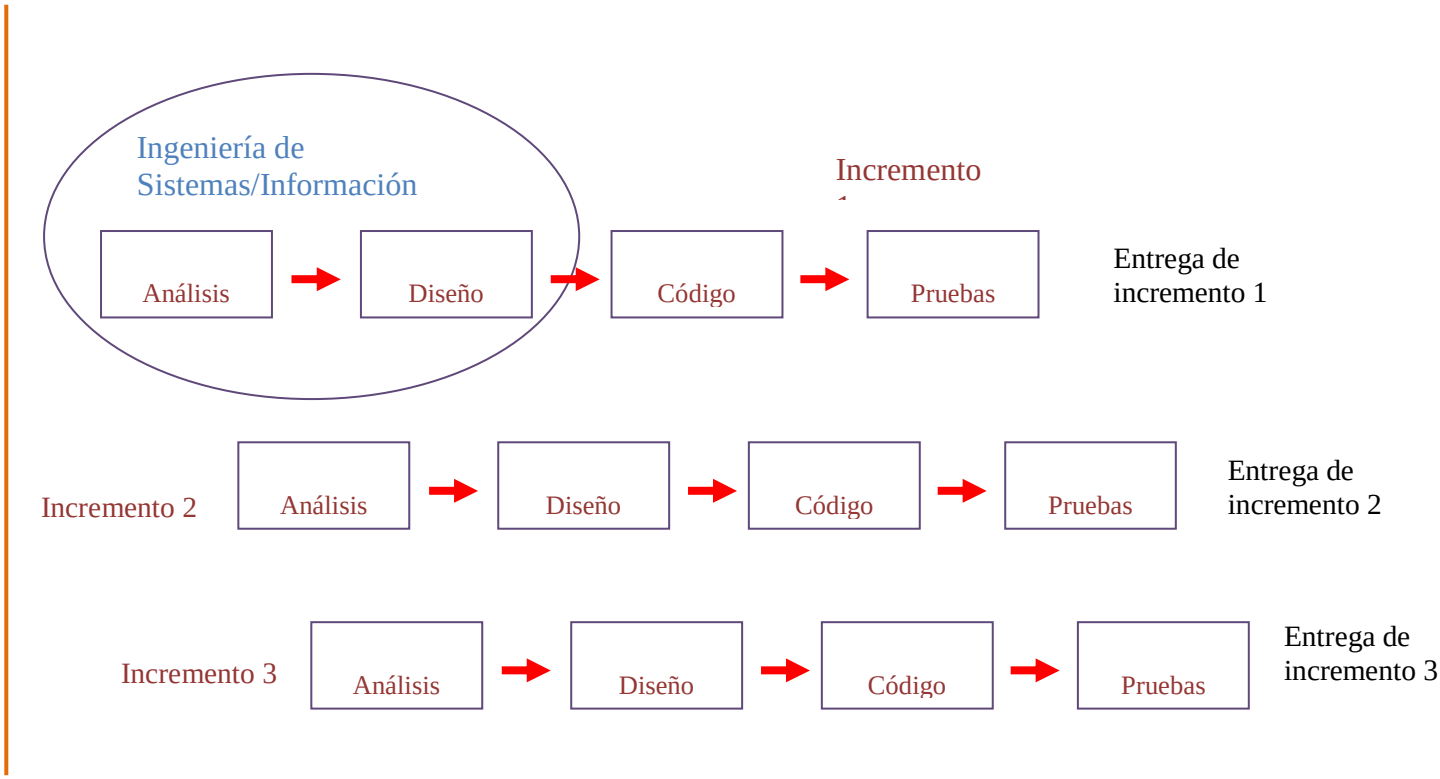


Figura 1.4 Modelo Incremental.

1.3.5. EL MODELO ESPIRAL

El *modelo en espiral,* como se muestra en la Figura 1.5 propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

El modelo en espiral se divide en un número de actividades de marco de trabajo, también llamadas *regiones de tareas.* Generalmente, existen entre tres y seis regiones de tareas.

Comunicación con el cliente - las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.

Planificación - las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.

Análisis de riesgos - las tareas requeridas para evaluar riesgos técnicos y de gestión.

Ingeniería - las tareas requeridas para construir una o más representaciones de la aplicación.

Construcción y acción - las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica)

Evaluación del cliente - las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

Cada una de las regiones están compuestas por un conjunto de tareas del trabajo, llamado *conjunto de tareas,* que se adaptan a las características del proyecto que va a emprenderse. Para proyectos pequeños, el número de tareas de trabajo y su formalidad es bajo. Para proyectos mayores y más críticos cada región de tareas contiene tareas de trabajo que se definen para lograr un nivel más alto de formalidad. En todos los casos, se aplican las actividades de

protección (por ejemplo: gestión de configuración del software y garantía de calidad del software)

Cuando empieza este proceso evolutivo, el equipo de ingeniería del software gira alrededor de la espiral en la dirección de las agujas del reloj, comenzando por el centro. El primer circuito de la espiral puede producir el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software. Cada paso por la región de planificación produce ajustes en el plan del proyecto.

El coste y la planificación se ajustan con la realimentación ante la evaluación del cliente. Además, el gestor del proyecto ajusta el número planificado de iteraciones requeridas para completar el software.

A diferencia del modelo de proceso clásico que termina cuando se entrega el software, el modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora. Una visión alternativa del modelo en espiral puede ser considerada examinando el *eje de punto de entrada en el proyecto* también reflejado en la Figura 1.5 Cada uno de los cubos situados a lo largo del eje pueden usarse para representar el punto de arranque para diferentes tipos de proyectos. Un «proyecto de desarrollo de conceptos» comienza en el centro de la espiral y continuará (aparecen múltiples iteraciones a lo largo de la espiral que limita la región sombreada central) hasta que se completa el desarrollo del concepto. Si el concepto se va a desarrollar dentro de un producto real, el proceso continúa a través del cubo siguiente (punto de entrada del proyecto de desarrollo del producto nuevo) y se inicia un «nuevo proyecto de desarrollo”. El producto nuevo evolucionará a través de iteraciones alrededor de la espiral siguiendo el camino que limita la región algo más brillante que el centro. En esencia, la espiral, cuando se caracteriza de esta forma, permanece operativa hasta que el software se retira. Hay veces en que el proceso está inactivo, pero siempre que se inicie un cambio, el proceso arranca en el punto de entrada adecuado (por ejemplo: mejora del producto).

El modelo en espiral es un enfoque realista del desarrollo de sistemas y de software a gran escala. Como el software evoluciona, a medida que progresa el proceso el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. El modelo en espiral utiliza la construcción de prototipos como mecanismo de reducción de riesgos, pero, lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo que refleja de forma más realista el mundo real. El modelo en espiral demanda una consideración directa de los riesgos técnicos en

todas las etapas del proyecto, y, si se aplica adecuadamente, debe reducir los riesgos antes de que se conviertan en problemáticos.

Pero al igual que otros paradigmas, el modelo en espiral no es la panacea. Puede resultar difícil convencer a grandes clientes (particularmente en situaciones bajo contrato) de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la evaluación del riesgo, y cuenta con esta habilidad para el éxito. Si un riesgo importante no es descubierto y gestionado, indudablemente surgirán problemas. Finalmente, el modelo no se ha utilizado tanto como los paradigmas lineales secuenciales o de construcción de prototipos. Todavía tendrán que pasar muchos años antes de que se determine con absoluta certeza la eficacia de este nuevo e importante paradigma. [9]

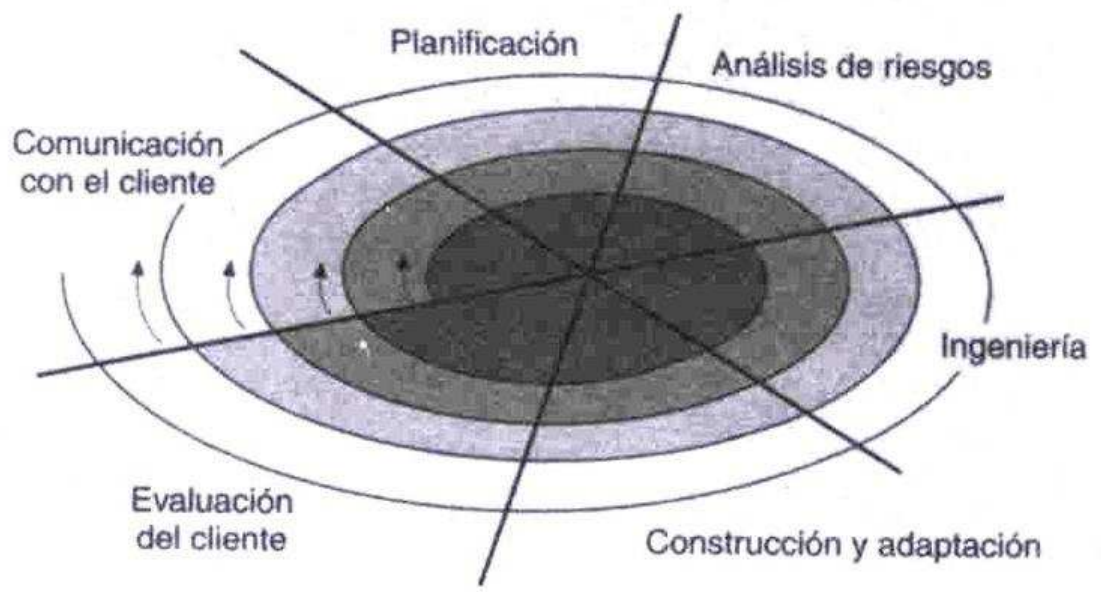


Figura 1.5 Modelo Espiral.

1.3.6. EL MODELO DE DESARROLLO CONCURRENTE

Davis y Sitaram han descrito el modelo de desarrollo concurrente ver Figura 1.6, llamado algunas veces ingeniería concurrente, de la forma siguiente:

Los gestores de proyectos que siguen los pasos del estado del proyecto en lo que se refiere a las fases importantes [del ciclo de vida clásico] no tienen idea del estado de sus proyectos. Estos son ejemplos de un intento por seguir los pasos extremadamente complejos de actividades mediante modelos demasiado simples. Tenga en cuenta que aunque un proyecto [grande] esté en la fase de codificación, hay personal de ese proyecto implicado en actividades asociadas

generalmente a muchas fases de desarrollo simultáneamente. Por ejemplo,. . . El personal está escribiendo requisitos, diseñando, codificando, haciendo pruebas y probando la integración [todo al mismo tiempo]. Los modelos de procesos de ingeniería del software de Humphrey y Kellner, han mostrado la concurrencia que existe para actividades que ocurren durante cualquier fase. El trabajo más reciente de Kellner utiliza diagramas de estado [una notación que representa los estados de un proceso] para representar la relación concurrente que existe entre actividades asociadas a un acontecimiento específico (por ejemplo: un cambio de requisitos durante el último desarrollo), pero falla en capturar la riqueza de la concurrencia que existe en todas las actividades de desarrollo y de gestión del software en un proyecto. .. La mayoría de los modelos de procesos de desarrollo del software son dirigidos por el tiempo; cuanto más tarde sea, más atrás se encontrará en el proceso de desarrollo. [Un modelo de proceso concurrente] está dirigido por las necesidades del usuario, las decisiones de la gestión y los resultados de las revisiones.

El modelo de proceso concurrente se puede representar en forma de esquema como una serie de actividades técnicas importantes, tareas y estados asociados a ellas. Por ejemplo, la actividad de ingeniería definida para el modelo en espiral, se lleva a cabo invocando las tareas siguientes: modelado de construcción de prototipos y/o análisis, especificación de requisitos y diseño.

La actividad -análisis- se puede encontrar en uno de los estados"' destacados anteriormente en cualquier momento dado. De forma similar, otras actividades (por ejemplo: diseño o comunicación con el cliente) se puede representar de una forma análoga. Todas las actividades existen concurrentemente, pero residen en estados diferentes. Por ejemplo, al principio del proyecto la actividad de *comunicación con el cliente* (no mostrada en la figura) ha finalizado su primera iteración y está en el estado de cambios, en espera. La actividad de *análisis* (que estaba en el estado ninguno mientras que se iniciaba la comunicación inicial con el cliente) ahora hace una transición al estado bajo desarrollo**.** Sin embargo, si el cliente indica que se deben hacer cambios en requisitos, la actividad *análisis* cambia del estado bajo desarrollo al estado cambios en espera.

El modelo de proceso concurrente define una serie de acontecimientos que dispararán transiciones de estado a estado para cada una de las actividades de la ingeniería del software. Por ejemplo, durante las primeras etapas del diseño, no se contempla una inconsistencia del modelo de análisis. Esto genera la *corrección del modelo de análisis* de sucesos, que disparará la actividad de *análisis* del estado hecho al estado cambios en espera.

El modelo de proceso concurrente se utiliza a menudo como el paradigma de desarrollo de aplicaciones cliente/servidor". Un sistema cliente/servidor se compone de un conjunto de componentes funcionales. Cuando se aplica a cliente/servidor, el modelo de proceso concurrente define actividades en dos

dimensiones: una dimensión de sistemas y una dimensión de componentes. Los aspectos del nivel de sistemas se afrontan mediante tres actividades: diseño, ensamblaje y uso.

La dimensión de componentes se afronta con dos actividades: diseño y realización. La concurrencia se logra de dos formas: **(1)** las actividades de sistemas y de componentes ocurren simultáneamente y pueden modelarse con el enfoque orientado a objetos descrito anteriormente; **(2)** una aplicación cliente/servidor típica se implementa con muchos componentes, cada uno de los cuales se pueden diseñar y realizar concurrentemente. En realidad, el modelo de proceso concurrente es aplicable a todo tipo de desarrollo de software y proporciona una imagen exacta del estado actual de un proyecto.

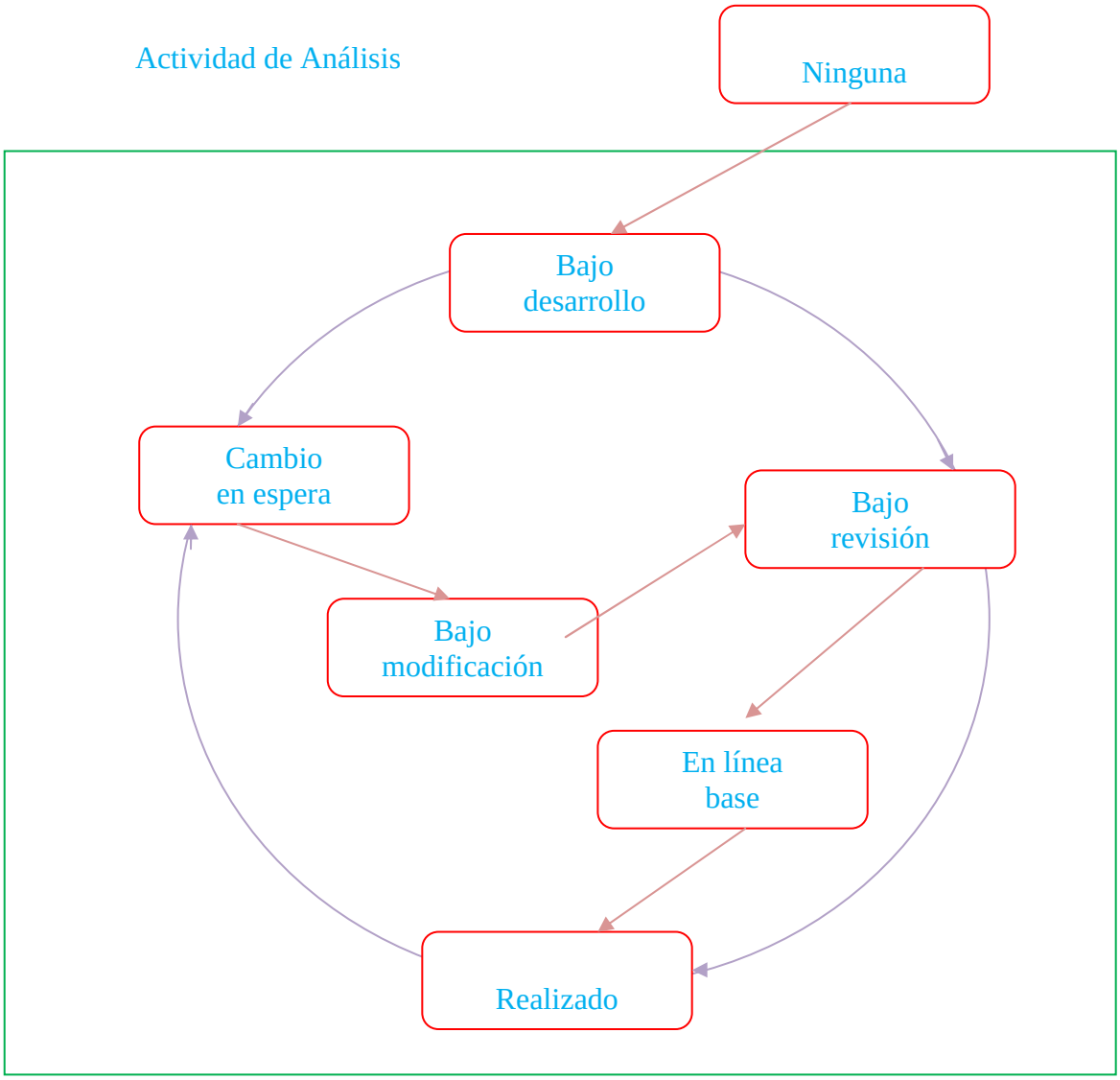


Figura 1.6 Modelo de Desarrollo Concurrente.

1.4. DISEÑO E IMPLEMENTACION DEL SOFTWARE

En el Diseño las especificaciones atraviesan dos procesos de diseño consecutivos. Primero surge el diseño arquitectónico, en el cual producto como un todo se divide en sus componentes, denominados módulos. Después, se diseña cada módulo; este proceso se denomina diseño detallado. Los dos documentos de diseño resultantes describen “cómo lo hará el producto”. Más específicamente, su objetivo es afinar los artefactos del flujo del trabajo del análisis hasta que el material se encuentre en tal forma que pueda ser implementado por los programadores.

En la Implementación cada uno de los diferentes componentes atraviesa por procesos de codificación y pruebas (pruebas unitarias). Después, los componentes del producto se combinan y prueban como un todo; esto se denomina integración. Cuando los desarrolladores están satisfechos de que el producto funciona correctamente, lo prueba el cliente (pruebas de aceptación)**.** La fase de implementación termina cuando el cliente acepta el producto y se instala en su computadora.

1.5. VERIFICACION Y VALIDACIÓN DEL SOFTWARE

La verificación y la validación del software (V y V) incluyen un conjunto de procedimientos, actividades, técnicas y herramientas que se utilizan paralelamente al desarrollo del software, para asegurar que el producto resuelve el problema para el que fuera diseñado [9].

La V y V actúa sobre los productos intermedios intentando detectar y corregir cuanto antes sus defectos y desviaciones del objetivo si las hubiera. Las tareas que abarca son las siguientes: pruebas de verificación, revisiones y auditoría e incluye las tareas de validación y pruebas de validación que se realizan durante el ciclo de vida del software para asegurar la satisfacción con los requisitos.

Cuando ya existe código ejecutable, se pueden realizar las pruebas del mismo, como verificación y validación del software. Las pruebas consisten en ejecutar el software con determinados datos de entrada y producir resultados que luego serán comparados con los teóricos.

1.6. CONCEPTOS DE BASE DE DATOS

Una **base de datos** o **banco de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso [10].

Una **base de datos** es un conjunto de información almacenada en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos

Una **base de datos** es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo [11].

Una **base de datos** es una colección de datos ordenados e interrelacionados, los cuales se almacenan independientemente de los programas de aplicación, empleando métodos bien definidos para dar de alta, modificar o extraer información, dichos datos son de relevante importancia para una empresa [12].

1.6.1. SISTEMA GESTOR DE BASE DE DATOS (SGBD)

Un **Sistema Gestor de Bases de Datos (SGBD)** consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. Su objetivo principal es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente [13].

1.6.2. ABSTRACCIÓN DE DATOS

Uno de los propósitos principales de un Sistema de Bases de Datos es proporcionar a los usuarios una visión *abstracta* de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos.

Existen tres niveles de abstracción:

- **Nivel físico:** el nivel más bajo de abstracción describe cómo se almacenan realmente los datos. En el nivel físico se describen a detalle las estructuras de datos complejas de bajo nivel.

- **Nivel lógico:** el siguiente nivel más alto de abstracción describe qué datos se almacenan en la base de datos y qué relaciones existen entre esos datos. Los administradores que deben decidir la información que se mantiene en la base de datos, usan el nivel lógico de abstracción.

- **Nivel de vistas:** el nivel más alto de abstracción describe sólo parte de la base de datos completa. A pesar del uso de estructuras más simples en el nivel lógico, queda algo de complejidad, debido a la variedad de información almacenada en una gran base de datos. El sistema puede proporcionar muchas vistas para la misma base de datos [13].

1.6.3. ESQUEMAS

La colección de información almacenada en la base de datos en un momento particular se denomina un **ejemplar** de la base de datos. El diseño completo de la base de datos se llama **esquema.**

Los esquemas son raramente modificados, si es que lo son alguna vez. Existen diferentes tipos de esquemas según los niveles de abstracción:

- Esquema físico: describe el diseño físico, puede ser fácilmente cambiado usualmente sin afectar los programas de aplicación.

- Esquema lógico: describe el diseño de la base de datos en el nivel lógico, los programadores construyen las aplicaciones usando el esquema lógico.

- Esquema en el nivel de vistas: una base de datos puede tener varios esquemas en este nivel, a menudo denominados *subesquemas.*

1.6.4. INDEPENDENCIA DE LOS DATOS

La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

1.7. MODELOS DE DATOS

Modelo de datos: es una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Los diferentes modelos de datos se clasifican en tres grupos diferentes:

- Modelos lógicos basados en objetos.
- Modelos lógicos basados en registros.
- Modelos físicos.

1.7.1.MODELO ENTIDAD-RELACIÓN

El **modelo de datos Entidad-Relación (E-R)** está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades y de relaciones entre estos objetos:

- **Entidad:** es una «cosa» u «objeto» en el mundo real que es distinguible de todos los demás objetos.
- **Atributo:** nos sirve para describir las entidades en una base de datos.
- **Relación:** es una asociación entre varias entidades.

1.7.1.1. DIAGRAMA ENTIDAD-RELACIÓN

Es la representación gráfica de la estructura lógica general de una base de datos y sus componentes son:

- **Rectángulos:** que representan conjuntos de entidades.
- **Elipses:** que representan atributos.
- **Rombos:** que representan relaciones entre conjuntos de entidades.
- **Líneas:** que en los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

1.7.1.2. CORRESPONDENCIA DE CARDINALIDADES

Es una restricción importante que expresa el número de entidades con las que otra entidad se puede asociar a través de un conjunto de relaciones.

1.7.2. MODELO RELACIONAL

En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla esta compuesta por varias columnas y cada columna tiene un nombre único.

1.8. CLAVES

Los valores de los tributos de una entidad de ben ser tales que permitan identificar unívocamente a la entidad. En otras palabras no se permite que ningún par de entidades tengan exactamente los mismos valores de sus atributos.

Una clave permite identificar un conjunto de atributos suficiente para distinguir las entidades entre sí.

Una superclave es un conjunto de uno o más atributos que tomados colectivamente permiten identificar de forma única una entidad en el conjunto de entidades.

Clave primaria es una clave que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades.

1.9. LA MANIPULACIÓN DE LOS DATOS

Podemos definir a la **manipulación de los datos** como:

- La recuperación de información almacenada en la base de datos.
- La inserción de información nueva en la base de datos.

- El borrado de información de la base de datos.
- La modificación de información almacenada en la base de datos.

SQL (**S**tructured **Q**uery **L**enguage, lenguaje estructurado de consultas) es un lenguaje de bases de datos ampliamente utilizado que conjuga la definición de datos así como la manipulación de los mismos.

1.10. LA NORMALIZACIÓN

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede [14].

Existen varios niveles de normalización:

1.10.1. Primera Forma Normal (1FN)

Una tabla está en Primera Forma Normal sólo si:

Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son indivisibles, mínimos.

La tabla contiene una clave primaria

La tabla no contiene atributos nulos

Una columna no puede tener múltiples valores. Los datos son atómicos. (Si a cada valor de X le pertenece un valor de Y, entonces a cada valor de Y le pertenece un valor de X).

1.10.2. Segunda Forma Normal (2FN)

Dependencia Funcional Total. Una relación esta en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existen dependencias parciales. En otras palabras podríamos decir que la segunda forma normal está completamente basada en el concepto funcional.

1.10.3. Tercera Forma Normal (3FN)

La tabla se encuentra en 3FN si es 2FN y cada atributo que no forma parte de ninguna clave, depende directamente y no transitivamente, de la clave primaria.

1.10.4. Forma Normal de Boyce-Codd (FNBC)

La tabla se encuentra en BCNF si está en 3FN y si cada determinante, atributo que determina completamente a otro, es clave candidata.

1.10.5. Cuarta Forma Normal (4FN)

Una tabla se encuentra en 4FN si está en 3FN, y sólo si, para cada una de sus dependencias múltiples no funcionales X->->Y, siendo X una super-clave que, X es o una clave candidata o un conjunto de claves primarias.

1.10.6. Quinta Forma Normal (5FN)

Una tabla se encuentra en 5FN si:

La tabla está en 4FN y No existen relaciones de dependencias no triviales que no siguen los criterios de las claves. Una tabla que se encuentra en la 4FN se dice que está en la 5FN si, y sólo si, cada relación de dependencia se encuentra definida por las claves candidatas [15].

1.11. LENGUAJE DE MODELADO UNIFICADO (UML)

El **Lenguaje de Modelado Unificado (UML, Unified Modeling Language)** es un estándar propuesto para la creación de especificaciones de varios componentes de un sistema de software. Algunas de los diagramas de UML son:

Diagrama de caso de uso los diagramas de caso de uso muestran la interacción entre los usuarios y el sistema, en particular los pases de las tareas que realiza el usuario.

Diagrama de actividad los diagramas de actividad describen el flujo de tareas entre varios componentes de un sistema.

Diagrama de implementación los diagramas de implementación muestran los componentes del sistema y sus interconexiones tanto en el nivel del componente software como el hardware.

Diagrama de objetos son similares a un diagrama de clases, pero en él se muestran instancias especificas de clases, es decir, objetos, en un momento particular del sistema.

Diagrama de secuencia estos constan de objetos, mensajes y líneas de tiempo, en ellos se establece la interacción entre objetos por medio de mensajes a través del tiempo.

1.12. MYSQL

MySQL es un sistema gestor de bases de datos relacionales, que además ofrece compatibilidades con PHP, Perl, C y HTML y funciones avanzadas de administración y optimización de bases de datos para facilitar las tareas habituales. Es un sistema cliente servidor de administración de bases de datos relacionales diseñado para el trabajo tanto en Windows como en UNIX/LINUX.

Podemos concluir que **MySQL** es un sistema gestor de bases de datos relacional cliente-servidor de coste mínimo que incluye un servidor SQL, programas cliente para acceder al servidor, herramientas administrativas y una interfaz de programación para escribir programas, es portable y se ejecuta en sistemas operativos comerciales [16].

Existen cuatro versiones de MySQL:

- **Estándar.** Incluye el motor estándar y la posibilidad de usar bases de datos
  InnoDB. Todo el potencial de MySQL, pero sin soporte completo para utilizar transacciones.

- **Max.** Para usuarios que quieran MySQL con herramientas de prueba para realizar opciones avanzadas de base de datos.

- **Pro.** Versión comercial del MySQL estándar.

- **Classic.** Igual que la estándar pero no dispone de soporte para **InnoDB** [17]**.**

El uso de MySQL (excepto en la versión Pro) está sujeto a licencia **GNU public license** (llamada GPL). Está licencia admite el uso de MySQL para crear cualquier tipo de aplicación.

1.13. LENGUAJES DE PROGRAMACION WEB

Actualmente existen diferentes lenguajes de programación para desarrollar en la web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas.

Desde los inicios de Internet, fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. A medida que paso el tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas a dar solución. Esto dió lugar a desarrollar lenguajes de programación para la web dinámicos, que permitieran interactuar con los usuarios y utilizaran sistemas de Bases de Datos. A continuación daremos una introducción a los diferentes lenguajes de programación para la Web.

1.13.1. LENGUAJE HTML

Desde el surgimiento de internet se han publicado sitios Web gracias al lenguaje HTML. Es un lenguaje estático para el desarrollo de sitios web (acrónimo en inglés de HyperText Markup Language, en español Lenguaje de Marcas Hipertextuales). Desarrollado por el World Wide Web Consortium (W3C). Los archivos pueden tener las extensiones (htm, html) [18]**.**

Ventajas:

- Sencillo que permite describir hipertexto.

- Texto presentado de forma estructurada y agradable.
- No necesita de grandes conocimientos cuando se cuenta con un editor de páginas web o WYSIWYG.
- Archivos pequeños.
- Despliegue rápido.
- Lenguaje de fácil aprendizaje.
- Lo admiten todos los exploradores.

Desventajas:

- Lenguaje estático.
- La interpretación de cada navegador puede ser diferente.
- Guarda muchas etiquetas que pueden convertirse en “basura” y dificultan la corrección.
- El diseño es más lento.
- Las etiquetas son muy limitadas.

1.13.2. LENGUAJE Javascript

Este es un lenguaje interpretado, no requiere compilación. Fue creado por Brendan Eich en la empresa Netscape Communications. Utilizado principalmente en páginas web. Es similar a Java, aunque no es un lenguaje orientado a objetos, el mismo no dispone de herencias. La mayoría de los navegadores en sus últimas versiones interpretan código Javascript [18].

El código Javascript puede ser integrado dentro de nuestras páginas web. Para evitar incompatibilidades el World Wide Web Consortium (W3C) diseño un estándar denominado DOM (en inglés Document Object Model, en su traducción al español Modelo de Objetos del Documento).

Ventajas:

- Lenguaje de scripting seguro y fiable.
- Los script tienen capacidades limitadas, por razones de seguridad.
- El código Javascript se ejecuta en el cliente.

Desventajas:

- Código visible por cualquier usuario.
- El código debe descargarse completamente.
- Puede poner en riesgo la seguridad del sitio, con el actual problema llamado XSS (significa en inglés Cross Site Scripting renombrado a XSS por su similitud con las hojas de estilo CSS).

1.13.3. PHP

PHP (Hypertext Pre-Processor) es un lenguaje de programación concebido principalmente como herramienta para el desarrollo de aplicaciones web. Nos permite diseñar páginas dinámicas de servidor, es decir, páginas capaces de responder de manera inteligente a las demandas del cliente y nos permiten la automatización de gran cantidad de tareas.

PHP lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor [18].

Comenzó siendo un conjunto de scripts escritos en Perl que permitían a su creador, Rasmus Lerdorf, el control de los accesos a sus páginas personales. A este conjunto de scripts se les denominó como *Personal Home Page Tools.* En 1995 nació PHP/FI y a finales de 1997 se libera PHP/FI 2.0, en Junio de 1998 se liberó oficialmente *PHP 3.0* siendo completamente reescrito por Andi Gutmans y Zeev Zuraski, en el *PHP 4.0* la evolución consistió en la reescritura de su núcleo, dando lugar a un nuevo motor denominado Zend fue liberado en Mayo de 2000; la ultima y actual versión ***PHP 5.0*** fue liberada en Junio de 2004 está basada en el nuevo motor *Zend 2*, que cuenta entre sus características el completo soporte para la programación orientada a objetos, incorpora la gestión de excepciones, una nueva librería de XML y un soporte nativo para el sistema gestor de bases de datos SQLite y una nueva ampliación de MySQL.

PHP está construido de forma modular, esto quiere decir que todos sus componentes se ejecutan en el mismo espacio de memoria PHP, lo cual hace que el código PHP puede ejecutarse más rápidamente al no sustituir la sobrecarga impuesta por la comunicación con los diferentes objetos.

Haciendo así a este lenguaje el más utilizado para en las aplicaciones Web, aunque debemos recordar que tiene otras aplicaciones [19].

Ventajas:

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objeto: Clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- Capacidad de expandir su potencial utilizando módulos.
- Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.

- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Desventajas:

- Se necesita instalar un servidor web.
- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- La programación orientada a objetos es aún muy deficiente para aplicaciones grandes.
- Dificulta la modularización.
- Dificulta la organización por capas de la aplicación.

Seguridad:

PHP es un poderoso lenguaje e intérprete, ya sea incluido como parte de un servidor web en forma de módulo o ejecutado como un binario CGI separado, es capaz de acceder a archivos, ejecutar comandos y abrir conexiones de red en el servidor. Estas propiedades hacen que cualquier cosa que sea ejecutada en un servidor web sea insegura por naturaleza.

PHP está diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI que Perl o C, y con la selección correcta de opciones de configuración en tiempos de compilación y ejecución, y siguiendo algunas prácticas correctas de programación.

2. ANÁLISIS DEL SISTEMA

2.1. INTRODUCCIÓN

Con el auge en los últimos años de la Computación ha habido un gran incremento en las personas interesadas en estudiar esta ciencia. Como está en constante crecimiento en todo momento se reportan avances científicos y/o tecnológicos. Por lo tanto es necesario que los usuarios tengan los conocimientos básicos bien definidos para poder continuar su proceso de actualización.

Existe una gran diversidad de diccionarios y glosarios de términos computacionales en línea, pero ninguno realmente enfocado a la materia de Introducción a la Disciplina Computacional. Lo anterior es muy importante ya que es necesario poder tener ideas claras acerca de lo que es la computación para que así los usuarios interesados en esta ciencia sean capaces de tener información confiable y fidedigna para poder elegir correctamente el perfil que desean tener. Esto nos ayudará a evitar que los alumnos interesados en la Ciencia de la Computación tomen una decisión errónea con respecto a su perfil de interés, disminuyendo de esta forma el número de bajas en las carreras computacionales.

2.1.1. DISCIPLINA COMPUTACIONAL

La computación al ser una ciencia joven provoca su muy rápida evolución, con acelerado ritmo surgen nuevas teorías, conceptos, métodos, circuitos, tipos de computadoras, lenguajes, programas, etc.

Por lo tanto es necesario que las personas interesadas en esta ciencia tengan una buena formación teórica, la cual le permitirá reaprender a lo largo de su vida profesional de tal manera que resista la obsolencia. De esta manera se forman egresados de computación capaces de hacer software, hablar de software y usar lo que otros hacen.

La introducción a la disciplina computacional nos permitirá tener una seria y razonada formación basada en dos aspectos básicos:

- La computadora y los sistemas de cómputo (Herramienta principal del estudioso de computación).

- Los lenguajes de programación (modo de usar o manejar la herramienta principal).

También representa un buen principio hacia otras fuentes de información más técnicas sobre arquitectura de computadoras, programación, teoría de la computación, software de base (*programación de sistemas*), y sistemas de información entre otros [2].

2.2. DEFINICIÓN DEL PROBLEMA

Como se mencionó anteriormente en la introducción, con la gran evolución que ha tenido la Computación es necesario tener reunido en un solo glosario los términos computacionales básicos de manera correcta y sencilla de entender, para así poder brindar los conocimientos necesarios a cualquier estudiante de la Ciencia de la Computación.

Este glosario se creará basado en la materia de Introducción a la Disciplina Computacional que es una materia que tiene como finalidad brindarles a los estudiantes la información relacionada con la ciencia de la computación para poder definir el perfil correcto de acuerdo al interés de cada alumno.

2.3. REQUERIMIENTOS FUNCIONALES

El Sistema nos permite hacer búsquedas de términos computacionales, estas búsquedas se pueden realizar de diferentes formas.

El usuario será capaz de escoger la búsqueda que más le convenga, ya sea buscar todas las palabras que tiene el glosario, buscar la palabra según la unidad a la que pertenece o buscar todas las palabras que empiecen con cierta letra del abecedario, así como también podrá dar sugerencias de términos faltantes.

2.4. REQUERIMIENTOS NO FUNCIONALES

El sistema al estar en Web tiene la dificultad que el tiempo de respuesta a una petición de usuario puede ser variable, y esto depende de circunstancias ajenas al sistema como lo sería la latencia y el ancho de banda de la red. Generalmente estas peticiones tardan segundos en realizarse, pero por lo mencionado anteriormente las operaciones requeridas por los usuarios pueden diferir en cuestión de algunos segundos.

2.5. MODELO DE CASOS DE USO

Una excelente técnica para mejorar la definición de los requisitos en la realización de programas, es la creación de casos de uso (descripción narrativa de los procesos). El modelo de uso es un aporte de Ivan Jacobson a UML. Fundamentalmente el modelo de uso, a través de los casos de uso, indica quien usa o interactúa con el sistema a desarrollar y como lo hace.

La figura 2.1 muestra el Diagrama de Casos de Uso del sistema, donde el actor usuario puede:

- Buscar Término.
- Sugerir términos.

Y el actor administrador puede realizar:

- Loggin.
- Altas.
- Bajas.
- Modificaciones.
- Revisar Sugerencias.

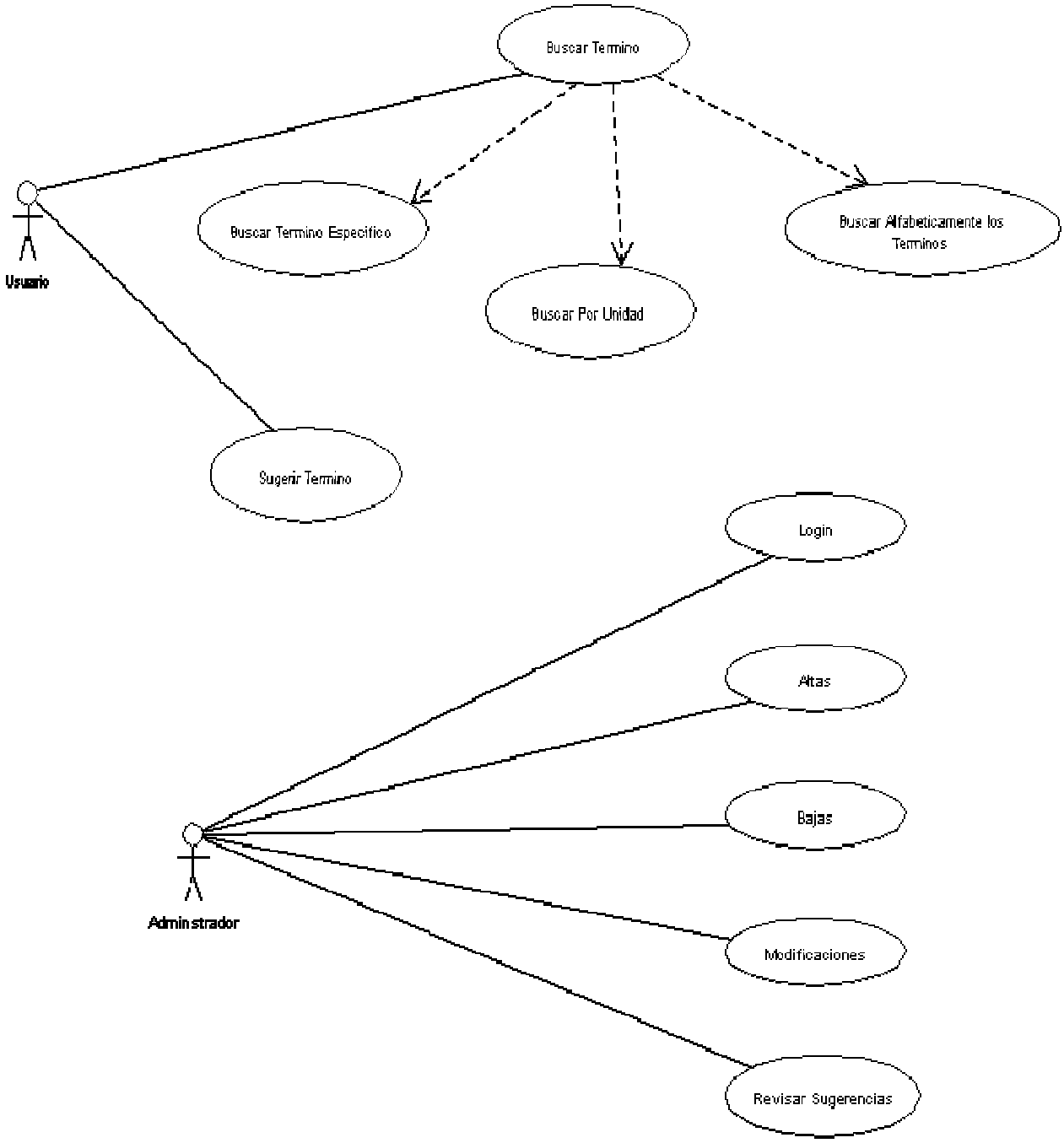


Fig. 3.1 Diagrama de Casos de Uso.

Lo que nos indica el diagrama de Casos de Uso, es que el caso de uso Buscar Término, incluye ya sea el caso de uso de: Buscar Término Específico o Buscar por Unidad o Buscar Alfabéticamente los Términos. Así que Buscar Término tiene en común, los 3 casos de uso ya mencionados.

2.5.1. ESPECIFICACIONES DE CASOS DE USO.

1. **Nombre del Caso de Uso:** Buscar Término. (Término Específico)

   Breve descripción: El usuario inicia el caso de uso. Este caso de uso se utiliza cuando se desea realizar la búsqueda de un término en específico.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**
     1. El sistema muestra los tipos de búsqueda por los que está conformado nuestro sistema.
     2. El actor selecciona el tipo de búsqueda que desea realizar, en este caso la búsqueda por término específico.
     3. El sistema despliega una pantalla donde se solicita que ingrese el término que desee buscar.
     4. El actor ingresa dicha información y oprime el botón buscar.
     5. El sistema muestra el resultado de la búsqueda.
        - 5.1. Si la palabra no fue encontrada, la pantalla mostrará dicha información.
        - 5.2. Si la palabra fue encontrada, el sistema mostrará la información requerida.
   - **2.2. Flujos Alternativos**
     No aplica.

3. **Requerimientos Especiales**
   No aplica

4. **Precondiciones**
   No aplica

1. **Nombre del Caso de Uso:** Buscar Término. (Búsqueda por Unidad)

   Breve descripción: El usuario inicia el caso de uso. Este caso de uso se utiliza cuando se desea realizar la búsqueda de los términos que pertenecen a una unidad en específico.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**
     1. El sistema muestra los tipos de búsqueda por los que está conformado nuestro sistema.
     2. El actor selecciona el tipo de búsqueda que desea realizar, en este caso la búsqueda por unidad.
     3. El sistema despliega una pantalla donde se solicita que ingrese la unidad de la cual desea saber los términos.
     4. El actor ingresa dicha información y oprime el botón buscar.
     5. El sistema muestra el resultado de la búsqueda.
        - 5.1. Si la unidad no contiene términos, la pantalla mostrará dicha información.
        - 5.2. Si la unidad tiene términos, el sistema mostrará la información requerida.
   - **2.2. Flujos Alternativos**
     No aplica.

3. **Requerimientos Especiales**
   No aplica

4. **Precondiciones**
   No aplica

1. **Nombre del Caso de Uso:** Buscar Término. (Alfabéticamente)

   Breve descripción: El usuario inicia el caso de uso. Este caso de uso se utiliza cuando se desea realizar la búsqueda de los términos alfabéticamente.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**

     1. El sistema muestra los tipos de búsqueda por los que está conformado nuestro sistema.
     2. El actor selecciona el tipo de búsqueda que desea realizar, en este caso la búsqueda alfabéticamente.
     3. El sistema despliega una pantalla donde se muestran las letras del abecedario.
     4. El actor oprime una letra.
     5. El sistema muestra el resultado de la búsqueda.
        - 5.1. Si no hay palabras que inicien con la letra seleccionada, la pantalla mostrará dicha información.
        - 5.2. Si hay palabras que inicien con la letra seleccionada, el sistema mostrara la información requerida.
   - **2.2. Flujos Alternativos**

     No aplica.

3. **Requerimientos Especiales**

   No aplica

4. **Precondiciones**

   No aplica

1. **Nombre del Caso de Uso:** Sugerir Término.

   Breve descripción: El usuario inicia el caso de uso. El sistema le muestra una pantalla donde debe de ingresar el término con el que desea sugerir, su significado y una liga donde se pueda ampliar la información la cual es opcional.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**
     1. El sistema desplegará una pantalla en donde se visualizará un formulario con los datos que deben de ingresarse.
     2. El actor ingresa el término, su definición y la liga y posteriormente pulsa el botón agregar.
     3. El sistema despliega una pantalla donde se muestra un mensaje de que el término fue agregado correctamente, así como también un link para agregar otra palabra o regresar a la página principal.
   - **2.2. Flujos Alternativos**
     No aplica.

3. **Requerimientos Especiales**
   No aplica

4. **Precondiciones**
   No aplica

1. **Nombre del Caso de Uso:** Loggin.

   Breve descripción: El administrador inicia el caso de uso. El sistema le muestra una pantalla donde debe de ingresar su nombre de usuario y su contraseña

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**
     1. El sistema desplegará una pantalla en donde se visualizará un formulario con los datos que deben de ingresarse.
     2. El actor ingresa su nombre de usuario y su contraseña y posteriormente pulsa el botón aceptar.
        - 2.1 Si alguno de los datos son incorrectos o si los datos no se encuentran registrados, se desplegara una pantalla con el mensaje de usuario no válido.
        - 2.2 Si el actor introduce correctamente los datos, se mostrará en pantalla las opciones para administrar el sistema.
   - **2.2. Flujos Alternativos**

     No aplica.

3. **Requerimientos Especiales**

   No aplica

4. **Precondiciones**

   No aplica

1. **Nombre del Caso de Uso:** Altas

   Breve descripción: El administrador del sistema da inicio al caso de uso. El sistema le desplegará una pantalla en donde dará de alta los términos. El administrador proporcionara el término en el campo correspondiente, así como también su definición y la liga correspondiente al término, la cual es opcional.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**
     1. El sistema muestra un formulario con los datos requeridos para ingresar un término.
     2. El actor ingresa los datos requeridos en el formulario: término, definición y liga. Posteriormente pulsa el botón agregar.
     3. El sistema despliega una pantalla un mensaje de que el término fue agregado satisfactoriamente.

   - **2.2. Flujos Alternativos**
     No aplica

3. **Requerimientos Especiales**
   No aplica

4. **Precondiciones**
   Que el actor haya realizado satisfactoriamente el caso de uso: Loggin.

1. **Nombre del Caso de Uso:** Bajas

   Breve descripción: El administrador del sistema da inicio al caso de uso. El sistema le desplegará una pantalla donde deberá de ingresar el término que desee eliminar.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**
     1. El sistema muestra una pantalla donde se solicitará el término que se desea eliminar.
     2. El actor ingresa la información requerida y pulsa el botón buscar.
     3. El sistema despliega en pantalla el resultado de la búsqueda.
        - 3.1 Si el término no fue encontrado, la pantalla mostrará dicha información.
        - 3.2 Si el término fue encontrado el sistema mostrará la opción para poder eliminarlo.
     4. El actor selecciona la opción de borrar asociada al término que desea eliminar.
     5. El sistema despliega un mensaje en la pantalla afirmando que la acción se realizó correctamente, es decir, que el término fue eliminado.

   - **2.2. Flujos Alternativos**
     No aplica

3. **Requerimientos Especiales**
   No aplica

4. **Precondiciones**
   Que el actor haya realizado satisfactoriamente el caso de uso: Loggin.

1. **Nombre del Caso de Uso:** Modificaciones

   Breve descripción: El administrador del sistema da inicio al caso de uso. El sistema le desplegara una pantalla donde deberá de ingresar el término que desee modificar después de esto se ingresaran los datos pertinentes para realizar la modificación.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**
     1. El sistema muestra una pantalla donde se solicitara el término que se desea modificar.
     2. El actor ingresa la información requerida y pulsa el botón buscar.
     3. El sistema despliega en pantalla el resultado de la búsqueda.
        - 3.1 Si el término no fue encontrado, la pantalla mostrara la dicha información.
        - 3.2 Si el término fue encontrado el sistema mostrara la opción para poder modificarlo.

     4. El actor selecciona la opción de modificar asociada al término que desea modificar.
     5. El sistema despliega en pantalla los datos actuales que tiene el término.
     6. El actor modificara los datos incorrectos, ya sea el término, su definición o su respectiva liga. Para posteriormente pulsar el botón modificar.
     7. Se desplegara un mensaje en la pantalla afirmando que la acción se realizo correctamente, es decir, que el termino fue modificado.

   - **2.2. Flujos Alternativos**
     No aplica

3. **Requerimientos Especiales**
   No aplica

4. **Precondiciones**
   Que el actor haya realizado satisfactoriamente el caso de uso: Loggin.

1. **Nombre del Caso de Uso:** Revisar Sugerencias

   Breve descripción: El administrador del sistema da inicio al caso de uso. El sistema le desplegara una pantalla que mostrara todas las palabras sugeridas por los usuarios.

2. **Flujo de eventos.**
   - **2.1. Flujo Básico**

     1. El sistema muestra una pantalla donde se desplegaran todas las palabras sugeridas por los distintos usuarios, así como también un botón para modificarlas, eliminarlas o poder darlas de alta en el sistema.
     2. El actor selecciona la opción que desea: modificar, borrar o cargar.
        - 2.1 Si el actor elige modificar, el sistema despliega en pantalla los datos actuales que tiene el término.
          - 2.1.1El actor modificará los datos incorrectos, ya sea el término, su definición o su respectiva liga. Para posteriormente pulsar el botón modificar.
          - 2.1.2. Se desplegará un mensaje en la pantalla afirmando que la acción se realizó correctamente, es decir, que el término fue modificado.
        - 2.2 Si el actor elige eliminar, el sistema despliega un mensaje en la pantalla afirmando que la acción se realizó correctamente, es decir, que el término fue eliminado.
        - 2.3 Si el actor elige cargar, el sistema despliega un mensaje en pantalla afirmando que la acción se realizo correctamente, es decir, que el término ha sido dado de alta correctamente.

   - **2.2. Flujos Alternativos**

     No aplica

3. **Requerimientos Especiales**

   No aplica

4. **Precondiciones**

   Que el actor haya realizado satisfactoriamente el caso de uso: Loggin.

3. DISEÑO DEL SISTEMA

3.1. DISEÑO CONCEPTUAL

Tras recabar los requerimientos antes mencionados, lo que procede es realizar el diseño de la base de datos para el sistema, utilizando el modelo Entidad – Relación, para así poder establecer los diferentes componentes de la base de datos, como lo son las diferentes entidades y las relaciones existentes entre ellas.

3.1.1. DIAGRAMA ENTIDAD – RELACIÓN

Las entidades que se presentarán a continuación son el resultado del análisis que se realizó a los requerimientos del sistema, se mostrara una descripción de cada una de ellas para un mejor entendimiento del diagrama Entidad – Relación.

Se utilizará la siguiente simbología para describir las entidades:

- "#" para el identificador único.
- "(FK)" para las llaves foráneas.
- "*" para los atributos obligatorios.
- "º" para los atributos opcionales.

Desc_Termino

\# * FK Id_Termino
\# * Id_Desc_Termino
* Def_Termino
º Liga_Termino

Nombre: DESC_ TERMINO
Campos obligatorios: Id_Termino, Id_Desc_Termino
Campos opcionales: Liga_Termino
Identificador único: Id_Termino, Id_Desc_Termino
Descripción: entidad que almacena la definición y la liga relacionadas a los términos.

Termino

* Id_Termino
* Termino

Nombre: Termino
Campos obligatorios: Id_Termino, Termino.
Campos opcionales:
Identificador único: Id_Termino.
Descripción: entidad que almacena todo los términos.

Term_Unidad

* FK Id_Unidad
* FK Id_Termino

Nombre: Term_Unidad
Campos obligatorios: Id_Unidad, Id_Termino
Campos opcionales:
Identificador único: Id_Unidad, Id_Termino
Descripción: entidad que almacena los términos con sus unidades.

Unidad

* Id_Unidad
* Nom_Unidad

Nombre: Unidad
Campos obligatorios: Id_Unidad, Nom_Unidad.
Campos opcionales:
Identificador único: Id_Unidad.
Descripción: entidad que almacena la información de las unidades.

Desc_Sugerencia

* FK Id_Sugerencia
* Id_Desc_Sugerencia
° Def_Term_Sugerencia
° Liga_Sugerencia

Nombre: Desc_Sugerencia
Campos obligatorios: Id_Sugerencia, Id_Desc_Sugerencia.
Campos opcionales: Def_Term_Sugerencia Liga_Sugerencia.
Identificador único: Id_Sugerencia, Id_Desc_Sugerencia.
Descripción: entidad que almacena la definición y la liga de los términos sugeridos por el usuario.

Sugerencia

* Id_Sugerencia
* Term_Sugerido

Nombre: Sugerencia
Campos obligatorios: Id_Sugerencia,
Term_Sugeriddo.
Campos opcionales:
Identificador único: Id_Sugerencia.
Descripción: entidad que almacena todo los términos sugeridos por el usuario.

Unid_Sugerencia

* FK Id_Sugerencia
* FK Id_Unidad

Nombre: Unid_Sugerencia
Campos obligatorios: Id_Sugerencia, Id_Unidad
Campos opcionales:
Identificador único: Id_Sugerencia, Id_Unidad
Descripción: entidad que almacena los términos sugeridos con su unidades.

Administrador

* Id_Administrador
* Nombre
* Clave_Acceso
* Activo

Nombre: Administrador
Campos obligatorios: Id_Administrador,
Nombre,
Clave_Acceso, Activo.
Campos opcionales:
Identificador único: Id_Administrador
Descripción: entidad que almacena los datos del administrador.

Después de que se han definido las entidades y sus atributos, se creará el diagrama Entidad – Relación a partir del cual se realizará el mapeo al modelo relacional.

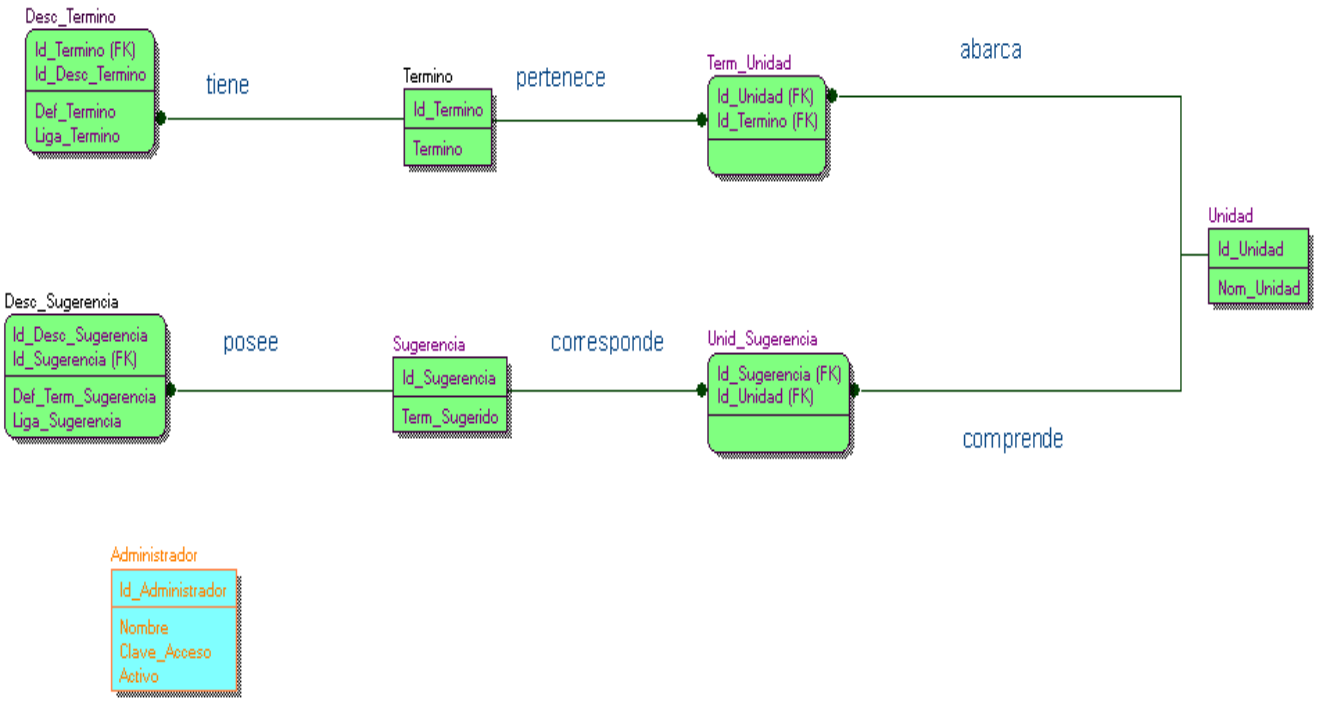


Figura 3.1 Diagrama Entidad – Relación.

Con el diagrama de la Figura 3.1 podemos realizar el mapeo al modelo relacional obteniendo así las tablas de la base de datos.

3.1.1.1. Descripción de Entidades

- Desc_Termino: Entidad que almacena la definición y la liga relacionada a los términos de la base de datos.
- Termino: Entidad que almacena los términos de la base de datos.
- Term_Unidad: Esta entidad consta de la relación entre que existe entre los términos y las unidades.

- Unidad: Entidad que almacena todas las unidades de la materia de Introducción a la Disciplina Computacional.

- Desc_Sugerencia: Entidad que almacena la definición y la liga relacionada a los términos sugeridos por los usuarios.

- Sugerencia: Esta Entidad está formada por todos los términos sugeridos por los usuarios.

- Unid_Sugerencia: Entidad formada de la relación que existe entre los términos sugeridos por los usuarios y las unidades de la materia de Introducción a la Disciplina Computacional.

- Administrador: En esta entidad se encuentran los datos relacionados con el administrador de la base de datos.

3.1.1.2 Descripción de Atributos.

- Desc_Termino (Id_Termino, Id_Desc_Termino, Def_Termino, Liga_Termino)

- Termino (Id_Termino, Termino)

- Term_Unidad (Id_Unidad, Id_Termino)

- Unidad (Id_Unidad, Nom_Unidad)

- Desc_Sugerencia (Id_Desc_Sugerencia, Id_Sugerencia, Def_Term_Sugerencia, Liga_Sugerencia)

- Sugerencia (Id_Sugerencia, Term_Sugerido)

- Unid_Sugerencia (Id_Sugerencia, Id_Unidad)

- Administrador (Id_Administrador, Nombre, Clave_Acceso, Activo)

Con la información anteriormente presentada podemos establecer las relaciones con atributos, llaves primarias (las cuales son los atributos con el signo “#” antepuesto) y las llaves foráneas (las cuales son atributos subrayados). Por lo tanto tenemos:

- Desc_Termino (#Termino, #Desc_Termino, Def_Termino, Liga_Termino)
- Termino (#Termino, Termino)
- Term_Unidad (#Unidad, #Termino)
- Unidad (#Unidad, Nom_Unidad)
- Desc_Sugerencia (#Desc_Sugerencia, #Sugerencia, Def_Term_Sugerencia, Liga_Sugerencia)
- Sugerencia (#Sugerencia, Term_Sugerido)
- Unid_Sugerencia (#Sugerencia, #Unidad)
- Administrador (#Administrador, Nombre, Clave_Acceso, Activo)

3.1.1.3. Definición de Relaciones.

tiene (Termino, Desc_Termino, 1:N)

pertenece (Termino, Term_Unidad, 1:N)

abarca (Unidad, Term_Unidad, 1:1)

tiene (Sugerencia, Desc_Sugerencia, 1:N)

pertenece (Sugerencia, Unid_Sugerencia, 1:N)

abarca (Unidad, Unid_Sugerencia, 1:1)

3.2 NORMALIZACIÓN

El objetivo de Normalizar una base de datos es para:

- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

Por lo que tenemos tres reglas básicas de normalización:

- Primera forma normal: Nos sirve para tener atributos atómicos, es decir, que son indivisibles.
- Segunda forma normal: Nos sirve para no tener dependencias parciales, es decir, que todos los atributos que no son parte de una clave dependen de forma completa de la clave primaria.
- Tercera forma normal: nos sirve para tener solo atributos que si no forman parte de ninguna clave, pertenecen directamente y no transitivamente de la clave primaria.

3.2.1. Tabla Desc_Termino

1FN

La primera forma normal, nos dice que los atributos de la tabla deben de ser atómicos y que no deben de existir repetidos.

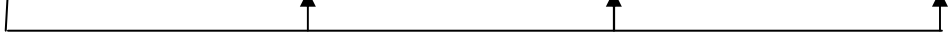
# Id_Desc_Termino	# Id_Termino	Def_Termino	Liga_Termino

Tras observar la tabla Desc_Termino podemos comprobar que se cumple con la definición por lo tanto la tabla Desc_Termino se encuentra en la primera forma normal.

2FN

La segunda forma normal, nos dice que la tabla debe de estar en1FN y además los atributos que no son llave primaria deben dependen funcional y totalmente de la llave primaria.

# Id_Desc_Termino	# Id_Termino	*Def_Termino*	*Liga_Termino*



En la tabla Desc_Termino, podemos comprobar que los atributos que no son llave primaria dependen total y funcionalmente de la llave primaria, por lo tanto se encuentran en 2FN.

3FN

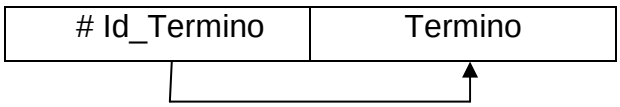
La tercera forma normal, nos dice que una tabla se encuentra en tercera forma normal si está en 2FN y que no haya atributos que no dependan transitivamente de la llave primaria. Como podemos observar los atributos de la tabla Desc_Termino dependen solo de la llave primaria, por lo tanto esta tabla esta en tercera forma normal.

3.2.2. Tabla Término

1FN

La tabla Término cuenta con atributos atómicos por lo tanto se encuentra en primera forma normal.

2FN



En la tabla Término, podemos comprobar que los atributos que no son llave primaria dependen total y funcionalmente de la llave primaria, por lo tanto se encuentran en 2FN.

3FN

Dado que todos los atributos de la tabla Término, dependen solo de la llave primaria, esta tabla se encuentra en tercera forma normal.

3.2.3. Tabla Term_Unidad

1FN

La tabla Term_Unidad cuenta con atributos atómicos por lo tanto se encuentra en primera forma normal.

2FN

#Id_Unidad	# Id_Termino

En la tabla Term_Unidad, encontramos que está formada solo por la llave primaria, por lo tanto no tiene otros atributos, esto quiere decir que esta en 2FN.

3FN

Dado que todos los atributos de la tabla Term_Unidad, son la llave primaria, esta tabla se encuentra en tercera forma normal.

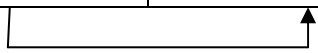
3.2.4. Tabla Unidad

1FN

La tabla Unidad cuenta con atributos atómicos por lo tanto se encuentra en primera forma normal.

2FN

# Id_Unidad	Nom_Unidad



En la tabla Unidad, podemos comprobar que los atributos que no son llave primaria dependen total y funcionalmente de la llave primaria, por lo tanto se encuentran en 2FN.

3FN

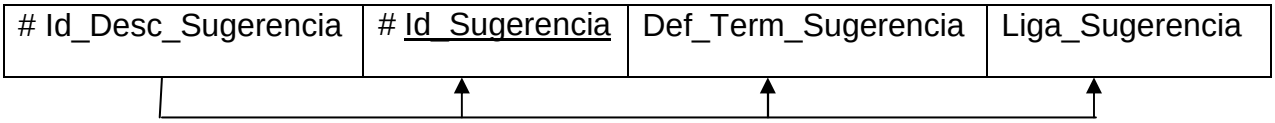
Dado que todos los atributos de la tabla Unidad, dependen solo de la llave primaria, esta tabla se encuentra en tercera forma normal.

3.2.5. Tabla Desc_Sugerencia

1FN

La tabla Desc_Sugerencia cuenta con atributos atómicos por lo tanto se encuentra en primera forma normal.

2FN



En la tabla Desc_Sugerencia, podemos comprobar que los atributos que no son llave primaria dependen total y funcionalmente de la llave primaria, por lo tanto se encuentran en 2FN.

3FN

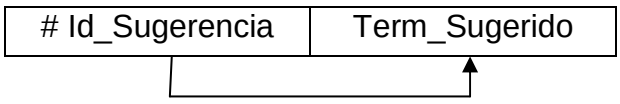
Dado que todos los atributos de la tabla Desc_Sugerencia, dependen solo de la llave primaria, está tabla se encuentra en tercera forma normal.

3.2.6. Tabla Sugerencia

1FN

La tabla Sugerencia cuenta con atributos atómicos por lo tanto se encuentra en primera forma normal.

2FN



En la tabla Sugerencia, podemos comprobar que los atributos que no son llave primaria dependen total y funcionalmente de la llave primaria, por lo tanto se encuentran en 2FN.

3FN

Dado que todos los atributos de la tabla Sugerencia, dependen solo de la llave primaria, está tabla se encuentra en tercera forma normal.

3.2.7. Tabla Unid_Sugerencia

1FN

La tabla Unid_Sugerencia cuenta con atributos atómicos por lo tanto se encuentra en primera forma normal.

2FN

# Id_Sugerencia	# Id_Unidad

En la tabla Unid_Sugerencia, encontramos que está formada solo por la llave primaria, por lo tanto no tiene otros atributos, esto quiere decir que esta en 2FN.

3FN

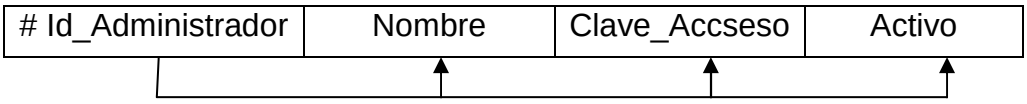
Dado que todos los atributos de la tabla Unid_Sugerencia, son la llave primaria, está tabla se encuentra en tercera forma normal.

3.2.8. Tabla Administrador

1FN

La tabla Administrador cuenta con atributos atómicos por lo tanto se encuentra en primera forma normal.

2FN



En la tabla Administrador, podemos comprobar que los atributos que no son llave primaria dependen total y funcionalmente de la llave primaria, por lo tanto se encuentran en 2FN.

3FN

Dado que todos los atributos de la tabla Administrador, dependen solo de la llave primaria, esta tabla se encuentra en tercera forma normal.

3.2.9. Esquema Relacional Normalizado

Después de efectuar la normalización el Esquema Relacional Normalizado se presenta en la Figura 3.2 tomando en cuenta la siguiente nomenclatura:

- "#" para el identificador único.
- "(FK)" para las llaves foráneas.

# Id_Desc_Termino	# Id_Termino (F.K.)	*Def_Termino*	*Liga_Termino*

# Id_Termino	Termino

# Id_Unidad (F.K.)	# Id_Termino (F.K.)

# Id_Unidad	Nom_Unidad

# Id_Desc_Sugerencia	# Id_Sugerencia (F.K.)	Def_Term_Sugerencia	Liga_Sugerencia

# Id_Sugerencia	Term_Sugerido

# Id_Sugerencia (F.K.)	# Id_Unidad (F.K.)

#Id_Administrador	Nombre	Clave_Accseso	Activo

Figura 3.2 Esquema Relacional Normalizado.

4. IMPLEMENTACIÓN Y PRUEBAS

4.1. HERRAMIENTAS UTILIZADAS

Por lograr que este sistema llegue a una gran cantidad de público se optó presentarlo a través de un sitio web. El uso de una herramienta para generar la base de datos antes diseñada, una plataforma para el desarrollo de la interfaz, y un servidor para el manejo de la información fueron necesarios. La utilización de PhpMyAdmin, PHP Editor y Wampserver, se considera herramientas de implantación.

A continuación se realizará una breve descripción del software utilizado para la realización de este Sistema.

4.1.1. PHPMYADMIN

PHPMyAdmin es un programa de libre distribución en PHP, creado por una comunidad sin ánimo de lucro. Es una herramienta muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz web muy intuitiva.

La aplicación en si no es más que un conjunto de archivos escritos en PHP que podemos copiar en un directorio de nuestro servidor web, de modo que, cuando accedemos a esos archivos, nos muestran unas páginas donde podemos encontrar las bases de datos a las que tenemos acceso en nuestro servidor de bases de datos y todas sus tablas. La herramienta nos permite crear tablas, insertar datos en las tablas existentes, navegar por los registros de las tablas, editarlos y borrarlos, borrar tablas y un largo etcétera, incluso ejecutar sentencias SQL y hacer un backup de la base de datos [20].

4.1.2. PHP EDITOR

PHP Editor es una herramienta perfecta para los programadores en lenguaje PHP, además puede ser utilizado para HTML, SQL, Java, JavaScript, C++, Python, etc. Con PHP Editor se podrá trabajar con varios documentos al mismo tiempo, éste incorpora un visualizador de páginas y un navegador interno, explorador de códigos y archivos, entre otras funciones.

El programa presenta una interfaz muy confortable y bastante intuitiva que hace muy fácil su manejo [21].

4.1.3. WAMPSERVER

(Windows-Apache-MySQL- PHP/Python/PERL). El término hace referencia al sistema creado por la conjunción de esas aplicaciones libres (de código abierto) y el sistema operativo Windows. Este grupo de aplicaciones generalmente son usados para crear servidores web[22].

WAMP es una forma de mini-servidor que puede ejecutarse en casi cualquier sistema operativo Windows. WAMP incluye Apache 2, PHP 5 (SMTP puertos son discapacitados), y MySQL (phpMyAdmin y SQLitemanager se instalan para administrar sus bases de datos) preinstalado. Un icono en la bandeja de la barra de tareas muestra el estado de WAMP, lo que le permitirá saber si:

a) WAMP está funcionando pero no se abren los servicios (el icono aparecerá de color rojo),

b) WAMP está funcionando y es un servicio abierto (el icono aparecerá amarillo) o

c) WAMP está funcionando con todos los servicios abiertos (el icono aparecerá de color blanco). Apache y MySQL se consideran servicios (que puede ser desactivado por la izquierda clic en el icono de la barra de tareas, orientando su cursor sobre el servicio que desea deshabilitar y seleccionando [23].

4.2 IMPLEMENTACIÓN

Como se mencionó anteriormente para generar las tablas de la base de datos y las tablas de la misma se utilizó MySQL. Una vez terminada la creación de la base de datos se hará una vinculación al sitio web realizado con PHP Editor a través del servidor WAMP.

4.2.1 INTERFACES DEL SISTEMA

En Sistema contará con diferentes tipos de búsquedas para un término y además el usuario podrá sugerir algún término que no se encuentre dado de alta dentro del glosario.

Por otra parte el administrador de la página, podrá realizar las funciones de: altas, bajas, modificaciones de términos, así como también podrá revisar los términos sugeridos por los usuarios y decidir eliminarlos o cargarlos a la tabla principal de los términos de la base de datos.

4.2.1.1. MENÚ USUARIO

Al ejecutar el inicio del Sistema, se mostrará la página principal del usuario(Fig. 4.1), la cual muestra la bienvenida para el usuario, así como también las diferentes búsquedas que se pueden realizar. Por último ésta le permite al administrador oprimiendo el botón correspondiente ir a sus opciones.

Figura 4.1 Página Principal del Sistema.

Después de que el usuario haya seleccionado el tipo de búsqueda deseado, se mostrará una nueva pantalla que corresponda al tipo de búsqueda (Fig.4.2).

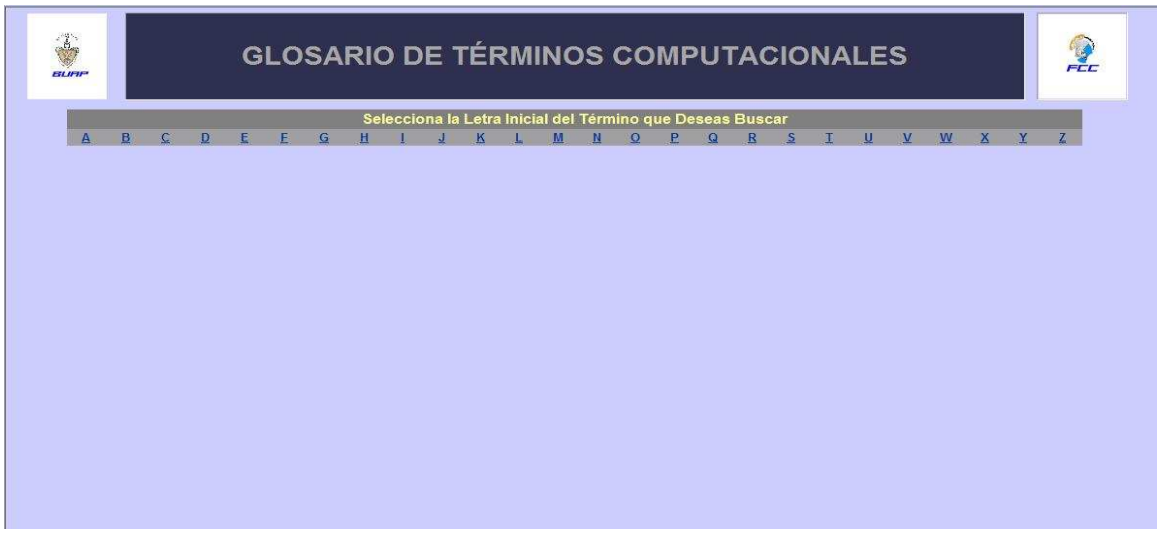


Figura 4.2 Página según la búsqueda.

En esta página, el usuario seleccionará la letra inicial de los términos que desea buscar, y al hacer clic sobre ella se desplegará la información. (Ver Fig. 4.3.)

Figura 4.3 Resultado de la Búsqueda deseada.

En esta misma página se muestran tres opciones de las cuales dos de ellas nos permiten realizar dos tipos diferentes de búsquedas al de la página actual y uno tercero que nos lleva a sugerir un término.

En el link de sugerir un término, se despliega un formulario (ver Figura 4.4) que nos permitirá mandar esta información al administrador de la página para su alta en el sistema.

Figura 4.4 Formulario de Sugerencias

4.2.1.2. MENÚ ADMINISTRADOR

La pantalla principal del administrador, nos muestra las diferentes opciones que pueden realizar (Fig. 4.5)

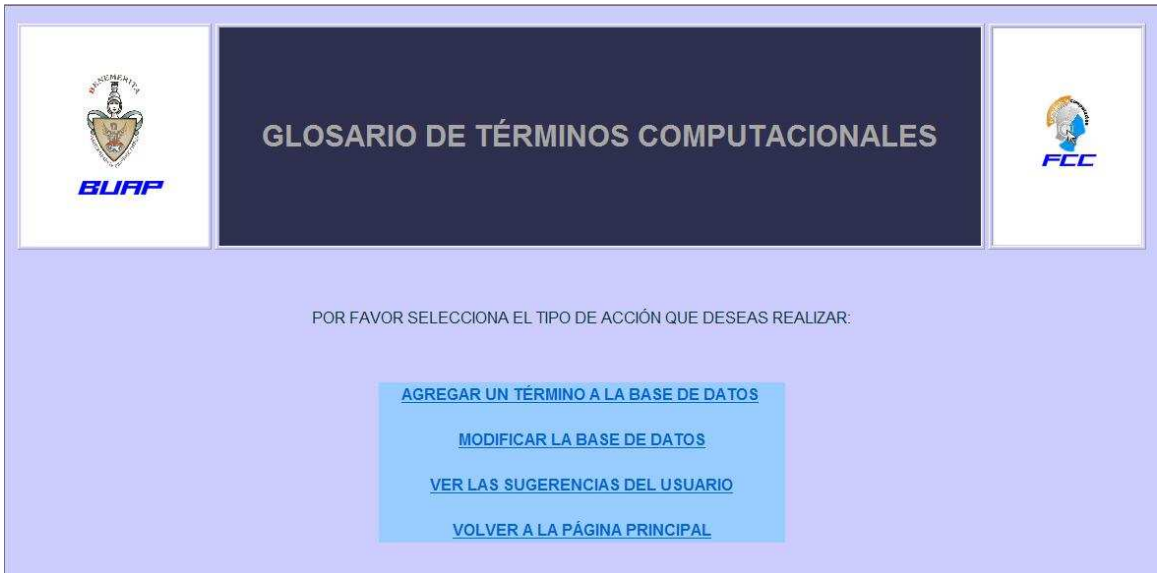


Figura 4.5 Página Principal del Administrador

En la primera opción se puede agregar un término a la base de datos a través de un formulario como se muestra en la Fig. 4.6.

Figura 4.6 Formulario para Agregar un Término

Para poder realizar este movimiento, es necesario proporcionar la información requerida en los diferentes campos, como lo son el término, su definición(es), su liga(s) siendo esta opcional, así como a la unidad(es) que pertenece dicho término y por último, dar clic en el botón Guardar (Fig. 4.7)

Figura 4.7 Llenado del Formulario para Agregar un Término

Cada vez que se tienen que llenar campos, se corre el riesgo de cometer algún error de ortografía, por lo cual, existe la alternativa de poder modificar los datos para su corrección.

Esta opción en el sistema, solo le corresponde al administrador, el cual tiene que buscar la palabra deseada en el glosario y después podrá corregir el error. Por ejemplo, si al agregar la palabra C que significa: El lenguaje C es una herramienta de programación de tipo general, utilizada para el desarrollo del sistema operativo Unix “Nosotros agregamos” El lenguaje C es una herramienta de programa de tipo general, utilizada para el desarrollo del sistema operativo Unix, primero tenemos que buscar el término en el glosario (ver Fig. 4.8) y una vez que encontremos el término, se seleccionará el link correspondiente como se muestra en la figura 4.9.

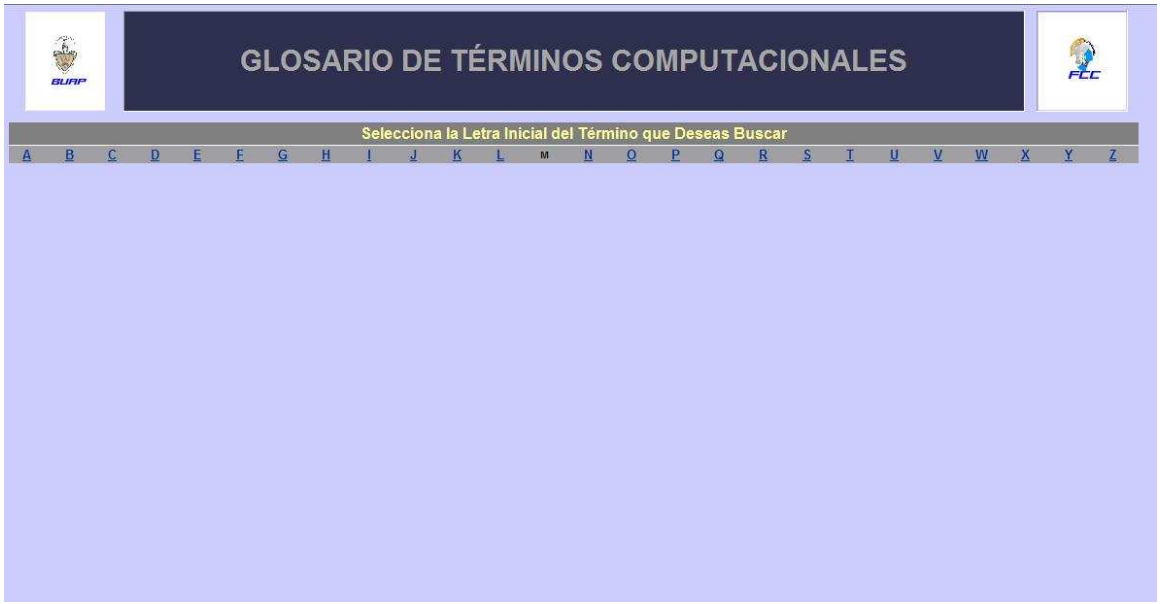


Figura 4.8 Búsqueda de un Término para Modificar

Figura 4.9 Resultado de la Búsqueda del Término para la Modificación

Una vez que se ha seleccionado la opción de modificar, en la siguiente pantalla se mostrará un formulario con los datos actuales como se muestra en la Figura 4.10, para que una vez realizada la corrección de la información se almacene nuevamente en nuestra base de datos, después de darle clic al botón modificar.

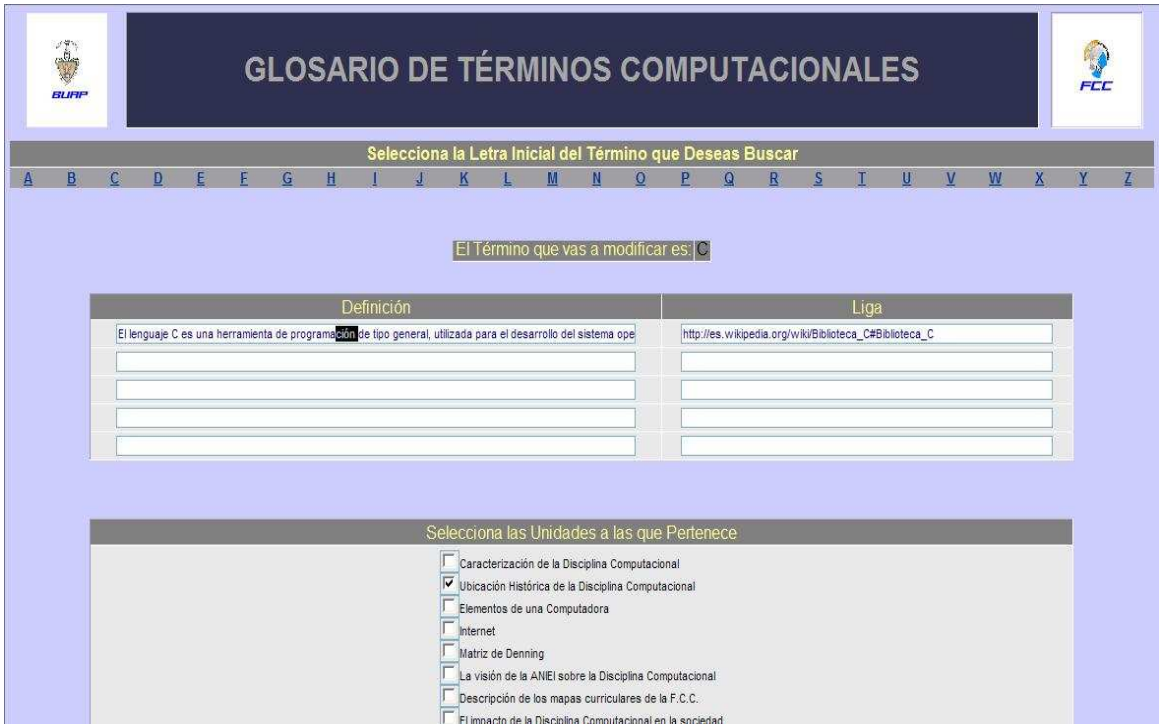


Figura 4.10 Datos Actuales del Término a Modificar

Otra de las opciones que tiene el administrador, es el poder realizar la eliminación de los términos de la base de datos.

Para poder realizar el borrado de algún término, primero se debe realizar una búsqueda en la base de datos del término y una vez que se muestran los términos encontrados se selecciona el término(s) que se desea eliminar y posteriormente se da clic en el botón borrar. (Ver Figura 4.11)

Figura 4.11 Lista de Términos para Borrar

Por último el administrador puede revisar las sugerencias realizadas por los usuarios, y decidir eliminarlas o cargarlas directamente a la base de datos. Cabe recordar que hasta que las sugerencias no hayan sido agregadas a la base de datos por el administrador, estas no podrán ser vistas por los usuarios (ver Figura 4.12).

Figura 4.12 Lista de Sugerencias

4.3 PRUEBAS

En este proceso se trata de encontrar errores que son el resultado de procedimientos no previstos y de verificar el correcto cumplimiento de los requerimientos funcionales.

En la página del usuario, tras realizar varias pruebas, se obtuvieron los siguientes resultados. El primero de ellos, es cuando al realizar la búsqueda por unidad, la unidad seleccionada no tiene ningún término, por lo que el Sistema mandará un mensaje como se muestra en la Figura 4.13.

Figura 4.13 No hay Términos en la Unidad Seleccionada

Otra prueba realizada, es cuando el término no se encuentra dentro del glosario, por lo tanto mandará un mensaje a pantalla como podemos apreciar en la figura 4.14.

Figura 4.14 El Término no Existe

En cuanto a la página del administrador, también se realizaron diferentes pruebas. Por ejemplo:

Para poder ingresar a la página donde le muestra las opciones, se debe validar el administrador, ya que al ser un Glosario de Términos con información real debemos de tener información verídica, por lo tanto el control de la base de datos es restringido. Para que esto suceda se valida un usuario a través de un formulario como se muestra en la Fig. 4.15.

Figura 4.15 Validación de Usuario

Una de las pruebas consistió al proporcionar el nombre de usuario incorrecto, después se realizó otra con un nombre correcto pero una contraseña diferente obteniendo como resultado la pantalla que se muestra en la Figura 4.16. Posteriormente se realizó otra con un nombre de usuario correcto y una contraseña correcta pero el usuario ya no está activo (ver Fig. 4.17).

Figura 4.16 Usuario y/o Contraseña no válidos.

Figura 4.17 Administrador Inactivo.

Una vez que el usuario ha ingresado correctamente su información y el sistema lo verifica, se le permite el acceso al mismo para poder hacer las funciones pertinentes y necesarias.

CONCLUSIONES Y PERSPECTIVAS

Podemos concluir que gracias a las metodologías aprendidas a través de la Ingeniería de Software obtuvimos un resultado exitoso en la creación del Sistema. De esta forma se comprobó que al implementar los modelos de desarrollo del software de una manera correcta podemos tener Sistemas confiables y funcionales.

Se ha conseguido la realización de un sistema que es fácil de utilizar y entender, confiable y de bajo costo, diseñado en línea para su fácil y rápido uso.

Este Sistema beneficiará a una comunidad de alumnos al permitirle a ellos encontrar información verídica bien organizada y concentrada, ahorrándoles tiempo para estudiar o investigar acerca de la Ciencia de la Computación, ya que actualmente dicha información se encuentra toda dispersa y en ocasiones no es muy adecuada o correcta.

El Sistema cuenta dentro de su funcionamiento básico con las consultas, las cuales se realizan escogiendo primero la búsqueda que más se adapte a las necesidades del usuario y posteriormente introduciendo el dato con el que se cuenta. En la sección de las sugerencias el usuario podrá colaborar con algún término que no haya encontrado en el glosario, aportando el término, su definición y una liga de donde obtuvo la información. Estas sugerencias se guardan en una tabla especial la cual posteriormente puede ser modificada por el administrador. Para poder navegar dentro de las diferentes pantallas del Sistema, se utilizaron diferentes ligas y botones que realizan acciones determinadas.

Durante la realización de este Sistema se aprendieron lenguajes de programación como MySql que es el Sistema Gestor de Base de Datos utilizado y el lenguaje PHP para la elaboración del entorno Web.

Se creó la base de datos de este Sistema denominado “Glosario de Términos Computacionales para Introducción a la Disciplina Computacional”, la cual está formada por 8 tablas, las cuales reciben su nombre, basándose en las características de las funciones que desempeñan dentro de la base de datos.

El Sistema está diseñado para una sola materia del plan de estudios de la Licenciatura e Ingeniería en Ciencias de la Computación ofrecidas en la Facultad de Ciencias de la Computación, pero este Sistema puede adaptarse para otras materias.

El Sistema presentado en este documento, se propone como material de apoyo para los alumnos de nuevo ingreso a las dos carreras de Computación que tiene nuestra Universidad, para ofrecer bases fuertes y una guía sobre Computación durante su carrera. Que contribuyen a evitar la deserción escolar.

BIBLIOGRAFÍA

1.- http://www.translationbureau.gc.ca/index.php?lang=francais&cont=s_700

2.- Levine, G. Guillermo “Computación y Programación moderna. Perspectiva integral de la informática”, Addison Wesley, 2001.

3.- http://www.monografias.com/trabajos/histocomp/histocomp.shtml

4.- http://es.wikipedia.org/wiki/Historia_de_la_inform%C3%A1tica

5.- http://www.bibliodgsca.unam.mx/tesis/tes4enal/sec_7.htm

6.- Stephen R. Schach “Ingeniería de Software Clasica y Orientada a Objetos”, McGraw Hill, 2006.

7.- http://zarza.usal.es/~fgarcia/docencia/isoftware/rincon.htm

8.- http://es.wikipedia.org/wiki/Desarrollo_de_software

9.- Pressman. S. Roger “Ingeniería de Software. Un enfoque práctico”, McGraw Hill, 2002.

10.- http://es.wikipedia.org/wiki/Base_de_datos

11.- http://www.monografias.com/trabajos11/basda/basda.shtml

12.- Elmasri Y. Navathe. “Fundamentos de Sistemas de Bases de Datos”, Adisson Wesley, 2000.

13.- Silberschatz Abraham, F. Korth Henry , Sudarshan S. “Fundamentos de Bases de Datos”, McGraw Hill, 2002.

14.- http://www.mysql-hispano.org/page.php?id=16

15.- http://es.wikipedia.org/wiki/Clave_for%C3%A1nea

16.- Pérez César. “MySQL para Windows y Linux”, Alfaomega Ra-MA, 2008.

17.- Sánchez Jorge. “MySQL Guía Rápida”, 2004.

18.-http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/

19.- Gutiérrez Abraham, Bravo Ginés. “PHP5 a través de ejemplos”, Alfaomega Ra-Ma, 2008.

20 .- http://www.desarrolloweb.com/articulos/844.php

21.- http://es.kioskea.net/telecharger/telecharger-303-php-editor

22.- http://www.alegsa.com.ar/Dic/wamp.php

23.- http://www.techfaq.com/lang/es/wamp.shtml&usg=AlkjrhhOn8padQAnJ0fcO13_7Jqwfj0tYA