JURECU

EDWIN DUVAN PATIÑO CARREÑO
FABIAN ORLANDO PINZON CASTILLO
FELIPE ANTONIO GARZON CELIS

COORPORACION UNIVERSITARIA MINUTO DE DIOS
FACULTAD INGENIERIA
TECNOLOGIA EN INFORMATICA
SOACHA
2012

JURECU

EDWIN DUVAN PATIÑO CARREÑO
FABIAN ORLANDO PINZON CASTILLO
FELIPE ANTONIO GARZON CELIS

Juguemos con regletas Cuisenaire

Director Del Proyecto
Ing. GABRIEL PUERTA

COORPORACION UNIVERSITARIA MINUTO DE DIOS
FACULTAD INGENIERIA
TECNOLOGIA EN INFORMATICA
SOACHA
2012

Nota de aceptación:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Bogotá 06 de Diciembre del 2012

AGRADECIMIENTOS

A Dios por su inmensa colaboración en la inspiración y consecución de este trabajo de grado.

A mi señor padre Fabián Enrique Pinzón y a mi señora madre Ligia Castillo por el respaldo económico y apoyo emocional que m brindaron para lograr esta meta.

A mi señor padre Luis José Patiño y a mi señora madre Olga Carreño por el apoyo económico y moral que me han dado durante la carrera. De la misma manera a mi hermano Edison Patiño por su corta pero valerosa colaboración en el desarrollo de este proyecto.

A mi señor padre Gilberto Garzón Bolívar y a mi señora madre Mayibe Celis por el apoyo económico y emocional que me han brindado durante el transcurso de la carrera y mi vida. Además a mis hermanos por el apoyo moral y por compartir conmigo sus conocimientos.

A Xiomara Ivonne Ruiz coordinadora del primer ciclo del colegio Cafam La Esperanza por la colaboración del tema del proyecto de grado.

Al Director de proyecto Gabriel Puerta por guiarnos en el proceso de la elaboración del proyecto de grado.

Al asesor metodológico Julio Eduardo Jejen por colaborarnos en el desarrollo del documento realizado para el proyecto de grado.

A la docente Ana María Obando por ayudarnos con el diseño del entorno grafico del software del proyecto de grado.
A los docentes de la universidad por aportarnos sus conocimientos durante la carrera.

Y a todas las demás personas que de una u otra forma formaron parte para la elaboración de este proyecto.

Atentamente,

Edwin Duvan Patiño Carreño
Fabián Orlando Pinzón Castillo
Felipe Antonio Garzón Celis

DEDICATORIA

Dedico este trabajo principalmente a Dios por haberme dado la vida. A mis padres y toda mi familia les doy las gracias por el acompañamiento que han tenido conmigo durante toda mi vida.

Les dedico este trabajo a mis padres y a mi hermano por el apoyo moral que me han brindado durante la carrera y mi existencia.

Le dedico este trabajo a Dios porque gracias a él es q puedo cumplir mis sueños.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ILUSTRACIONES

LISTA DE ANEXOS

Pág.

GLOSARIO

ANTECEDENTES: toda investigación tiene antecedentes. Consisten en la presentación de la información más relevante y directamente relacionada con nuestro tema de investigación y que podamos considerar aportes en referencia a este.

ATRIBUTOS: son los que representan los datos asociados al objeto, o lo que es lo mismo sus propiedades o características. Los atributos y sus valores en un momento dado, determinan el estado de un objeto.

BASE DE DATOS: las bases de datos almacenan datos, permitiendo manipularlos fácilmente y mostrarlos de diversas formas

CICLO DE VIDA: el ciclo de vida del software es una sucesión de estados o fases por los cuales pasa un software a lo largo de su "vida".

CLASES: una clase es una construcción que se utiliza como un modelo (o plantilla) para crear objetos de ese tipo. El modelo describe el estado y el comportamiento que todos los objetos de la clase comparten.

COLORES: es la impresión producida al incidir en la retina los rayos luminosos difundidos o reflejados por los cuerpos.

CONSULTAS: una consulta es el método para acceder a los datos en las bases de datos. Con las consultas se puede modificar, borrar, mostrar y agregar datos en una base de datos.

DATO: representación simbólica (numérica, alfabética, etc.) de un atributo de una entidad. Un dato no tiene valor semántico (sentido) en sí mismo, pero al ser procesado puede servir para realizar cálculos o tomar decisiones

DIAGRAMA: representación gráfica de una sucesión de hechos, pasos u operaciones en un procedimiento.

DIAGRAMA DE FLUJO: el diagrama de flujo o diagrama de actividades es la representación gráfica del algoritmo o proceso.

EJECUTABLE: en programación, como resultado de la fase de enlace, el enlazador guardará, en disco, un archivo ejecutable. Dicho archivo será "el ejecutable".

ESTADO: el estado de un objeto se refiere al conjunto de los valores de sus atributos en un instante de tiempo dado.

FACTIBILIDAD: factibilidad se refiere a la disponibilidad de los recursos necesarios para llevar a cabo los objetivos o metas señalados. Generalmente la factibilidad se determina sobre un proyecto.

FORMULARIO: los formularios presentan una visión ordenada de múltiple información sobre algo, y son útiles para llenar bases de datos

IDENTIDAD: la identidad es la propiedad que permite a un objeto diferenciarse de otros.

IMÁGENES: es una representación visual, que manifiesta la apariencia visual de un objeto real o imaginario.

INSTANCIA: la instancia de una clase es el objeto generado a partir de esa clase. Puede haber muchas instancias de una clase, en otras palabras, varios objetos generados a partir de esa clase.

LENGUAJE: el concepto de lenguaje puede ser entendido como un recurso que hace posible la comunicación. En el caso de los seres humanos, esta herramienta se encuentra extremadamente desarrollada y es mucho más avanzada que en otras especies animales, ya que se trata de un proceso de raíces fisiológicas y psíquicas.

LICENCIAS: es una especie de contrato, en donde se especifican todas las normas y cláusulas que rigen el uso de un determinado programa.

LOGIN: el login es el momento de autentificación al ingresar a un servicio o sistema.

MATEMÁTICAS: el término matemáticas viene del griego "máthema", que quiere decir aprendizaje, estudio y ciencia. Y justamente las matemáticas son una disciplina académica que estudia conceptos como la cantidad, el espacio, la estructura y el cambio.

METODOLOGÍA: la rama de la metodología, dentro de la ingeniería de software, se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

METODOLOGÍA XP: la programación extrema es una metodología de desarrollo ligero (o ágil) basada en una serie de valores y de prácticas de buenas maneras

que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.

METODOLOGÍA RUP: el Proceso Unificado Racional, Rational Unified Process en inglés, y sus siglas RUP, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

MÉTODOS: son los que acceden a los atributos de una manera predefinida e implementan el comportamiento del objeto

MÓDULOS: en programación, un módulo es un software que agrupa un conjunto de subprogramas y estructuras de datos.
OBJETOS: un objeto se define como la unidad que en tiempo de ejecución realiza las tareas de un programa. También a un nivel más básico se define como la instancia de una clase.

PARÁMETROS: un parámetro es un tipo de variable que es recibida por una función, procedimiento

REGLETAS CUISENAIRE: son un juego de manipulación matemática utilizado para enseñar una amplia variedad de temas matemáticos como las operaciones básicas.

SOFTWARE: en computación, el software -en sentido estricto- es todo programa o aplicación programada para realizar tareas específicas. El término "software" fue usado por primera vez por John W. Tukey en 1957.

SQL SERVER: SQL Server es el Sistema de gestión de bases de datos de Microsoft. Aunque evidentemente no se trata de un producto de código abierto, en Open Alfa hemos considerado interesante investigar y comentar sus características, comparando su funcionalidad y rendimiento con la que ofrecen PostgreSQL y MySQL

TABLAS: unidad donde se crea el conjunto de datos de una base de datos. Estos datos estarán ordenados en columnas verticales.

TEST: las pruebas tipo test, son denominadas también pruebas objetivas. Con ellas se puede deducir de forma bastante fiable los conocimientos sobre una materia.

UML: es un popular lenguaje de modelado de sistemas de software. Se trata de un lenguaje gráfico para construir, documentar, visualizar y especificar un sistema de software. Entre otras palabras, UML se utiliza para definir un sistema de software.

USUARIO: en informática, un usuario es un individuo que utiliza una computadora, sistema operativo, servicio o cualquier sistema informático. Por lo general es una única persona.

VISUAL STUDIO: Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML

WINDOWS: es la Familia de sistemas operativos gráficos (GUI) para computadoras desarrollada por la empresa Microsoft. Su traducción literal al español es Ventanas, pues su interfaz se basa en ellas. Microsoft Windows es el sistema operativo más usado del mundo con un 90% de penetración en el mercado.

RESUMEN

Teniendo como base la investigación realizada en el colegio Cafam La Esperanza se observaron fallas en temas relacionados con las asignaturas de matemáticas y lenguaje en los grados 1 y 2 de este colegio, junto con la metodología de las regletas Cuisenaire creadas George Cuisenaire surge la idea de analizar y diseñar un software didáctico – educativo con esta metodología que sirva como herramienta de aprendizaje en el colegio específicamente en los estudiantes de grado 1 y 2. Por medio de las regletas Cuisenaire se quiere lograr que los estudiantes comprendan la matemática de forma divertida interactuando con el software. A demás el software JURECU aportará al estudiante la asignatura de castellano formando palabras, frases y oraciones para que los estudiantes del colegio puedan tener una buena interpretación y comprensión de imágenes.

Palabras claves: regletas, colegio, matemáticas, estudiantes.

ABSTRACT

Taking as a basis the research conducted in the school Cafam Hope failures were observed in subjects related to the subjects of math and language arts in grades 1 and 2 of this college, along with the methodology of the strips Cuisenaire George created the idea of analyze and design a learning software - educational with this methodology that serves as a learning tool specifically at school students in grades 1 and 2. Using Cuisenaire strips is to make students understand mathematics in a fun interacting with the software. A JURECU other software provide the student the subject of Castilian forming words, phrases and sentences for college students to have a good performance and image understanding.

INTRODUCCION

Las regletas cuisenaire creadas por Emile George Cuisenaire quien nace en cuaregnon (Bélgica) a finales del siglo XIX en el año 1891 y murió en Thuin 1975, En 1952 fueron creadas como un material matemático destinado para que los niños aprendan matemáticas. El material se compone de 10 regletas de diferentes colores y distintos tamaños. En 1953 Cuisenaire conoce al profesor Caleb Gateño quien con ayuda de este señor difundieron la metodología de las regletas cuisenaire. Al ser manipulativo permite que los niños resuelvan los diferentes problemas que se plantean gracias a su propia experiencia. Así irán adquiriendo el concepto de número más fácilmente que con la representación numérica aprendida de memoria.

Tomando en cuenta el uso de las regletas Cuisenaire y la falta de un software que permita que los estudiantes amplíen sus conocimientos se optó por el diseño de un software, aprovechando los recursos informáticos que posee el colegio Cafam la esperanza.

Este software didáctico-educativo se realiza con el fin de mejorar el uso actual de las herramientas usadas como lo son las guías, el tablero, y demás elementos físicos en el colegio Cafam la esperanza. El software consiste en desarrollar e implementar las regletas Cuisenaire para la asignatura de matemáticas especialmente en las operaciones básicas (suma, resta, multiplicación, división) y para la asignatura de lenguaje tendrá los temas de: textos con imágenes, creación de frases y palabras

1 FASE DE INICIO

Se utiliza la metodología en XP para el desarrollo de este proyecto.

1.1 TITULO DEL PROYECTO

JURECU

1.2 TEMA

“Juguemos con regletas Cuisenaire”

1.3 PLANTEAMIENTO DEL PROBLEMA

En la actualidad, los estudiantes de primer y segundo grado del colegio Cafam La Esperanza presentan dificultad en las asignaturas de matemáticas y lenguaje.

1.4 DESCRIPCIÓN Y FORMULACIÓN DEL PROBLEMA

De acuerdo a la investigación que se realizó en el colegio Cafam ubicado en el sector de Bosa La Esperanza, se determinó que los alumnos de los grados 1° y 2°, evidencian dificultad en la asignatura de matemáticas, en el tema de las operaciones básicas específicamente en suma, resta, multiplicación y división; existe la posibilidad de que el docente piense desarrollar algunas guías con los estudiantes, y él no puede debido a que los estudiantes no comprenden la asignatura y se le dificulta al docente tener que repetir lo mismo varias veces debido a que no puede avanzar tema logrando así un retraso en los temas a desarrollar durante el periodo y el año. Del mismo modo los alumnos de este colegio no sólo presentan dificultad en matemáticas sino que también presentan dificultad en la asignatura de lenguaje, al momento de realizar talleres con los docentes formando palabras, frases y en la comprensión de imágenes con textos. La posible solución a estas dificultades que se presenta en el colegio, es diseñar un software didáctico-educativo que contengan las operaciones básicas (sumas restas multiplicación y división) también los respectivos temas de lenguaje como los son formando palabras, frases y la comprensión de imágenes con textos. Y lo anterior se hace en tres módulos y estos a su vez se dividan en formularios.

1.5 ALCANCES Y DELIMITACIONES

Para la ejecución de este proyecto se han establecido los alcances y delimitaciones que van a tomar en diseño e implementación del software.

- El software está diseñado bajo la plataforma de Windows 7
- El software en la versión 1.0 solo tendrá los módulos de matemáticas, lenguaje y regletas.
- las pruebas que se realizarán en este módulo de evaluación serán sobre los temas de regletas, castellano y matemáticas trabajados en el software JURECU
- los requerimientos mínimos para usar este software son:
  Sistema Operativo Windows xp
  * Procesador: Dual Core de 2.1 GHz
  * RAM: 2 GB
  * Disco: 250GB
  * Tarjeta gráfica: 512MB

- el software funciona en la plataforma de Windows xp y 7
- El software está dirigido únicamente a estudiantes de grado primero y segundo del colegio Cafam la esperanza.
- El proyecto solo tendrá en sus actividades de desarrollo, en las asignaturas de matemáticas y lenguaje.
- El proyecto tendrá en la asignatura de matemáticas los temas de suma, resta, multiplicación y división en sus actividades de desarrollo, y en la asignatura de lenguaje tendrá los temas de formando palabras, frases y en la comprensión de imágenes con textos.

2 OBJETIVOS

Para este proyecto se plantearon los siguientes objetivos.

2.1 GENERAL

Análisis y diseño de un software didáctico-educativo utilizado la metodología de las regletas Cuisenaire que sirva de refuerzo en las asignaturas de matemáticas y castellano para los estudiantes de ciclo 1(grado 1,2) de básica primaria.

2.2 ESPECÍFICOS

-Diseñar un formulario que genere preguntas con sus respectivas respuestas cerradas, que permita al docente reconocer el desarrollo del estudiante con el manejo del software educativo.

-Reconocer el manejo de las regletas Cuisenaire, mediante el diseño de un formulario que permita el uso de las operaciones básicas.

-Crear un módulo que permita a los estudiantes interactuar con el software mediante el uso de distintos juegos didácticos.

-Elaborar un formulario que permita resolver actividades como la comprensión de textos con ayuda de imágenes, la formación de palabras o frases para colaborar con la lectura y escritura.

-Evaluar la retentiva visual y agilidad mental que el estudiante puede llegar a tener, con la ayuda de un formulario que contenga imágenes iguales en forma aleatoria y pueda mostrar un resultado por medio de un test.

- Crear una base de datos en SQL que guarde los puntajes obtenidos por los estudiantes en los diferentes módulos, para que los docentes observen el progreso del estudiante.

2.3 JUSTIFICACION

Este software didáctico-educativo se realiza con el fin de mejorar el uso actual de las herramientas usadas como lo son las guías, el tablero, y demás elementos físicos en el colegio Cafam la esperanza El software consiste en desarrollar e implementar las regletas Cuisenaire para la asignatura de matemáticas

especialmente en las operaciones básicas (suma, resta, multiplicación, división) y para la asignatura de lenguaje tendrá los temas de: textos con imágenes, creación de frases y palabras, ya que estos procesos se llevan de modo físico lo cual genera ciertas dificultades como la perdida de los materiales y el desorden de la información.
De este modo se ayudará a mejorar estos procedimientos en el colegio Cafam la esperanza, aumentando su rendimiento, eficacia y colaborar al medio ambiente evitando la tala de árboles para la construcción de las regletas, la utilización de los químicos o pintura que pueden afectar la capa de ozono. Este software didáctico-educativo también reduce los gastos económicos en el colegio ya que al tener el software no se requiere tener en modo físico las regletas Cuisenaire y además evita imprimir la gran cantidad de guías o textos puesto que esto genera un alto gasto económico y un gran deterioro del medio ambiente

2.4 HIPOTESIS

2.4.1 GENERAL

¿Es posible construir un software didáctico-educativo o herramienta que use la metodología de las regletas Cuisenaire que pueda aportar a los estudiantes del colegio Cafam la esperanza un refuerzo en la asignatura de matemáticas?

2.5 MISION EL PROYECTO

Diseñar un software didáctico-educativo que ayude a generar “aprendizaje atractivo, eficaz y colectivo” en los estudiantes de 1 y 2 grado del colegio Cafam la Esperanza.

2.6 VISION DEL PROYECTO

La visión de JURECU es lograr en el 2015 se lleve a cabo la implementación del software didáctico-educativo en todos los colegios de Bogotá.

3 MARCO DE REFERENCIA

3.1 ANTECEDENTES

El desarrollo del software didáctico-educativo juguemos con regletas Cuisenaire (JURECU) propone adaptar las regletas Cuisenaire creadas por Georges Cuisenaire quien nace en Cuaregnon (Bélgica) a finales del siglo XIX. Formado como músico (primer premio de violín en el conservatorio de Mons en 1907), y posteriormente como maestro en la Escuela Normal de Mons en 1911, obtiene el diploma de profesor de música en 1920, aunque desde 1912 ejerce de maestro de primaria en una escuela rural en Thuin en Bélgica. Especialista en Música, debe aguzar su ingenio para enseñar matemática en su escuela.

Las Regletas Cuisenaire son un material matemático destinado para que los niños aprendan matemáticas. Este método pedagógico que se utiliza frecuentemente en las aulas de educación infantil gracias a Georges Cuisenaire, en donde la utilización de las regletas se realiza con la pregunta como soporte didáctico. Éste material matemático destinado básicamente a que los niños aprendan la composición y descomposición de los números e iniciarles en actividades de cálculo, todo ello sobre una base manipulativa. El material se compone de 10 regletas de diferentes colores y tamaños. Al ser manipulativo permite que los niños resuelvan los diferentes problemas que se plantean gracias a su propia experiencia. Así irán adquiriendo el concepto de número más fácilmente que con la representación numérica aprendida de memoria.

Tomando en cuenta el uso de las regletas Cuisenaire y la falta de un software que permita que los estudiantes amplíen sus conocimientos se optó por el diseño de dicho software, aprovechando los recursos informáticos que posee el colegio Cafam la esperanza.

3.2 MARCO CONCEPTUAL

Conceptos fundamentales que se tomaron en cuenta para la solución al problema planteado.

BASE DE DATOS: Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente. A continuación te presentamos una guía que te explicará el concepto y características de las bases de datos.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una base de datos se puede definir como

un conjunto de información relacionada que se encuentra agrupada o estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

CARACTERÍSTICAS
Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar

CICLO DE VIDA: se refiere al período de tiempo que comienza cuando se concibe la idea de generar el programa hasta que finalmente se retira.

Waterfall (en cascada): Se denomina modelo en cascada porque su característica principal es que no se comienza con un paso hasta que no se ha terminado el anterior.
El principal problema de esta aproximación es el que no podemos esperar el que las especificaciones iniciales sean correctas y completas y que el usuario puede cambiar de opinión sobre una u otra característica. Además los resultados no se pueden ver hasta muy avanzado el proyecto por lo que cualquier cambio debido a un error puede suponer un gran retraso además de un alto coste de desarrollo.

Como es evidente esto es solo un modelo teórico, si el usuario cambia de opinión en algún aspecto tendremos que volver hacia atrás en el ciclo de vida.

Prototipos: Consiste en iterar en la fase de análisis tantas veces como sea necesario, mostrando prototipos al usuario para que pueda indicarnos de forma más eficiente los requisitos del sistema. La iteración finalizará cuando el usuario dé el visto bueno al prototipo.

Evolutivo: Se diferencia del modelo por prototipos en que en prototipos se da por hecho que aunque se necesiten varias iteraciones para lograrlo al final se llegará a tener una serie de requisitos completos y sin errores, que no vayan a cambiar más.
En el modelo evolutivo se asume que los requisitos pueden cambiar en cualquier momento del ciclo de vida y no solo en la etapa de análisis.

Incremental: Es una aproximación muy parecida a la evolutiva. En este modelo se desarrolla el sistema para satisfacer un subconjunto de los requisitos especificados y en posteriores versiones se incrementa el programa con nuevas funcionalidades que satisfagan más requisitos.
En el caso del modelo evolutivo se desarrollaría una nueva versión de todo el sistema, en el incremental se parte de la versión anterior sin cambios y le añadimos las nuevas funciones.

En espiral: Toma las ventajas del modelo de desarrollo en cascada y el de prototipos añadiéndole el concepto de análisis de riesgo.
Se definen cuatro actividades:
Planificación, en la que se recolectan los requisitos iniciales o nuevos requisitos a añadir en esta iteración.
Análisis de riesgo; basándonos en los requisitos decidimos si somos capaces o no de desarrollar el software y se toma la decisión de continuar o no continuar.
Ingeniería, en el que se desarrolla un prototipo basado en los requisitos obtenidos en la fase de planificación.

Evaluación del cliente: el cliente comenta el prototipo. Si está conforme con él se acaba el proceso, si no se añaden los nuevos requisitos en la siguiente iteración.
Basada en transformaciones: Derivado del modelo en cascada, en él se considera que partiendo de las especificaciones y gracias a las herramientas CASE estas se transforman en diseño lógico del software, este se transforma en un diseño físico (un diseño dependiente de la tecnología) y éste en el código final.

DATO: la definición del concepto de Dato como toda asignación aislada de Cifras**,** Conceptos e Instrucciones que no tienen ni contexto ni correlación entre sí, sino que son meras representaciones simbólicas de algo, o bien una unidad mínima de lo que posteriormente puede ser una Información.
Pero para que estos datos puedan constituir una información específica, es necesario que exista un debido Procesamiento, que permita una organización básica por lo menos, que permita asignar un contexto y un orden determinado, siendo lo contrario solamente un Conjunto de Datos Aislados que por sí mismos no pueden valer más que una representación única de sí mismos.
Esta tarea de ordenamiento es también aplicable a la informática, en la que los Datos Aislados por sí mismos no tienen valor alguno, sino que deben tener un Contexto determinado y un Orden específico, el cual se da mediante el debido Proceso, en el cual lógicamente interviene la Unidad Central de

Procesamiento (también conocida por sus siglas en inglés, CPU, o bien sus equivalentes de Procesador o Microprocesador)
Cuando ya hemos ordenado y contextualizado esto, ya no tendremos simples Datos, sino que habremos obtenido como producto una Información, siendo ésta factible de ser retransmitida hacia un nuevo rumbo, interpretada por un usuario y a su vez compartida con otros usuarios, a través del uso de distintos dispositivos.

En lo que respecta a la Informática, podemos definir entonces como Dato a todas las instrucciones que son en su unidad procesadas por uso del CPU, siendo representaciones de la Energía Eléctrica que circula a través de los distintos componentes empleando el Código Binario (Que consiste en la transmisión o no-transmisión de impulsos eléctricos)
La información que ha pasado por el Procesador es entonces enviada a los distintos Periféricos de Salida para que un usuario pueda interpretarlos, leerlos, percibirlos y a su vez compartirlo con otros usuarios, o bien interactuar con ellos mediante un Periférico de Entrada, que estará enviando nuevos datos al procesador y estableciendo una Comunicación entre los mismos.
Por este motivo es que si bien se suelen utilizar como sinónimos, el trato ambiguo que se le da a los Datos y a la Información es erróneo, siendo para estas últimas la necesidad de contar con un contexto determinado y la asignación de un orden, un entorno y una aptitud para poder ser leídas, interpretadas y compartidas.

METODOLOGÍA XP: La programación extrema es una metodología de desarrollo ligero (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.
Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.
Una de las características principales de este método de programación, es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El resultado de esta selección ha sido esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, sí que el resultado es una nueva manera de ver el desarrollo de software.
El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos. Aplicando el sentido común.

Principios básicos
La Programación Extrema se basa en 12 principios básicos agrupados en cuatro categorías:
Retroalimentación a escala fina

1. El principio de pruebas*:* se tiene que establecer un período de pruebas de aceptación del programa (llamado también *período de caja negra*) donde se definirán las entradas al sistema y los resultados esperados de estas entradas. Es muy recomendable automatizar estas pruebas para poder hacer varias simulaciones del sistema en funcionamiento. Para hacer estas simulaciones automatizadas, se pueden utilizar *Ambientes de Prueba* (Unit testing frameworks). Un buen ejemplo de un ambiente de prueba es el JUnit para Java Otros ambientes de pruebas para otros lenguajes como C, C++, JavaScript, XML y servicios Web.
2. Proceso de planificación: en esta fase, el usuario tendrá que escribir sus necesidades, definiendo las actividades que realizará el sistema. Se creará un documento llamado *Historias del usuario* (User Stories). Entre 20 y 80 historias (todo dependiendo de la complejidad del problema) se consideran suficientes para formar el llamado *Plan de Liberación*, el cual define de forma específica los tiempos de entrega de la aplicación para recibir retroalimentación por parte del usuario. Por regla general, cada una de las Historias del usuario suelen necesitar de una a tres semanas de desarrollo. Son muy importantes y tienen que ser una constante las reuniones periódicas durante esta fase de planificación. Estas pueden ser a diario, con todo el equipo de desarrollo para identificar problemas, proponer soluciones y señalar aquellos puntos a los que se les ha de dar más importancia por su dificultad o por su punto crítico.
3. El cliente en el sitio: se le dará poder para determinar los requerimientos, definir la funcionalidad, señalar las prioridades y responder las preguntas de los programadores. Esta fuerte interacción cara a cara con el programador disminuye el tiempo de comunicación y la cantidad de documentación, junto con los altos costes de su creación y mantenimiento. Este representante del cliente estará con el equipo de trabajo durante toda la realización del proyecto.
4. Programación en parejas: uno de los principios más radicales y en el que la mayoría de gerentes de desarrollo pone sus dudas. Requiere que todos los programadores XP escriban su código en parejas, compartiendo una sola máquina. De acuerdo con los experimentos, este principio puede producir aplicaciones más buenas, de manera consistente, a iguales o menores costes. Aunque el *pair-programming* puede no ser para todo el mundo.
Proceso continuo en lugar de por lotes
1. Integración continua: permite al equipo hacer un rápido progreso implementando las nuevas características del software. En lugar de crear *builds* (o versiones) estables de acuerdo a un cronograma establecido, los equipos de programadores XP pueden reunir su código y reconstruir el sistema varias veces al día. Esto reduce los problemas de integración comunes en proyectos largos y estilo cascada.
2. Refactorización: permite a los equipos de programadores XP mejorar el diseño del sistema a través de todo el proceso de desarrollo. Los programadores evalúan continuamente el diseño y recodifican lo necesario. La finalidad es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente.
3. Entregas pequeñas: colocan un sistema sencillo en producción rápidamente

que se actualiza de forma rápida y constante permitiendo que el verdadero valor de negocio del producto sea evaluado en un ambiente real. Estas entregas no pueden pasar las 2 o 3 semanas como máximo.

Entendimiento compartido

1. Diseño simple*:* se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos. *Simple Design* se enfoca en proporcionar un sistema que cubra las necesidades inmediatas del cliente, ni más ni menos. Este proceso permite eliminar redundancias y rejuvenecer los diseños obsoletos de forma sencilla.

2. Metáfora*:* desarrollada por los programadores al inicio del proyecto, define una historia de cómo funciona el sistema completo. XP estimula historias, que son breves descripciones de un trabajo de un sistema en lugar de los tradicionales diagramas y modelos UML (*Unified Modeling Language*). La metáfora expresa la visión evolutiva del proyecto que define el alcance y propósito del sistema. Las tarjetas CRC (Clase, Responsabilidad y Colaboración) también ayudarán al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos y define sus responsabilidades (lo que ha de hacer) y las colaboraciones con las otras clases (cómo se comunica con ellas).

3. Propiedad colectiva del código*:* un código con propiedad compartida. Nadie es el propietario de nada, todos son el propietario de todo. Este método difiere en mucho a los métodos tradicionales en los que un simple programador posee un conjunto de código. Los defensores de XP argumentan que mientras haya más gente trabajando en una pieza, menos errores aparecerán.

4. Estándar de codificación: define la propiedad del código compartido así como las reglas para escribir y documentar el código y la comunicación entre diferentes piezas de código desarrolladas por diferentes equipos. Los programadores las han de seguir de tal manera que el código en el sistema se vea como si hubiera estado escrito por una sola persona.

Bienestar del programador.

1. La semana de 40 horas: la programación extrema sostiene que los programadores cansados escriben código de menor cualidad. Minimizar las horas extras y mantener los programadores frescos, generará código de mayor calidad. Como dice *Beck*, está bien trabajar tiempos extra cuando es necesario, pero no se ha de hacer durante dos semanas seguidas.

3.3 MARCO REFERENCIAL

Las páginas web o software relacionados con las regletas Cuisenaire se encuentran a continuación junto con algunas de las ventajas y desventajas de cada una de ellas:

El software se llama (NÚMEROS DE COLORES)

Ventajas de este software.

- ✓ Que el software es dinámico
- ✓ Permite interactuar el software con el usuario
- ✓ Los niños aprenden las operaciones básicas

Desventajas de este software.

- ✓ La combinación de los colores no es factible para los niños
- ✓ El software no dan una buena explicación de lo que son las regletas
- ✓ Las aplicaciones que contiene el software no son las más adecuadas para los niños

La página web se llama (REGLETAS CUISSENAIRE EN INFANTIL DE 5 AÑOS)

Ventajas de esta página

- ✓ En la página web explican el color y numero respectivo de cada regleta
- ✓ Que tiene los objetivos propuestos que desea cumplir con ese página web
- ✓ Explican que son las regletas Cuisenaire.

Desventajas de esta página

- ✓ La página web no testa diseñada para interactuar con el niño
- ✓ La página web tiene mucho texto y pocas imágenes
- ✓ La letra es muy pequeña y a los niños no le gusta leer ese tipo de letra
- ✓ El texto está mal ubicado debería estar centrado

La otra página web se llama (REGLETAS CUISSENAIRE (números en color))

Ventajas de esta página web

- ✓ Dicen las ventajas y limitaciones que tiene el uso de recursos para las regletas
- ✓ Colocaron actividades relacionadas con las regletas para los niños
- ✓ Explican las operaciones básicas(suma, resta, división, multiplicación, división) una por una usando las regletas

Desventajas de esta página

- ✓ Las operaciones que hacen son muy complejas para los niños
- ✓ La página web contiene mucho texto

3.4 MARCO LEGAL

La licencia utilizada en los equipos del colegio Cafam La Esperanza es:

WINDOWS 7 HOME BASIC Y ULTIMATE

Los presentes términos de licencia constituyen un contrato entre Microsoft Corporación (o, en función de donde resida, una de sus filiales) y usted. Le rogamos que los lea atentamente. Son de aplicación al software arriba mencionado, el cual incluye, en su caso, los soportes físicos en los que lo haya recibido. Los términos de la licencia en papel impreso, que pueden venir con el software, podrán modificar o sustituir cualquier término de la licencia que aparezca en pantalla. Estos términos también se aplicarán a los siguientes elementos de Microsoft

· Actualizaciones
· Complementos
· Servicios basados en Internet
· Servicios de soporte

Todos ellos deben corresponder a este software, a excepción de que existan otros términos aplicables a dichos elementos. En tal caso, se aplicarán esos otros términos.

El uso del software implica la aceptación de estos términos. Si no los acepta, no utilice el software. En lugar de ello, devuélvalo al distribuidor para obtener un reembolso o crédito. Si no puede obtener un reembolso de este modo, póngase en contacto con Microsoft o con la filial de Microsoft de su país para obtener información sobre la política de reembolsos de Microsoft. Consulte www.microsoft.com/worldwide. Para México, llame al (011)(91) (55) 5267-2000, o bien visite el sitio Web www.microsoft.com/mexico/default.asp.

Tal como se describe más adelante, el uso del software implica un consentimiento por su parte para la transmisión de determinada información durante la activación y la validación, y para los servicios basados en Internet.

Si cumple los presentes términos de licencia, tendrá los siguientes derechos por cada licencia que adquiera.

1. INTRODUCCIÓN.

a. Software. El software incluye el software del sistema operativo de escritorio. Este software no incluye servicios Windows Live. Windows Live es un servicio que ofrece Microsoft en virtud de un contrato independiente.

b. Modelo de Licencia. Se otorga una licencia de software por copia y por equipo. Un equipo es un sistema de hardware físico con un dispositivo de almacenamiento interno capaz de ejecutar el software. Una partición o división de hardware se considera un equipo independiente.

2. DERECHOS DE INSTALACIÓN Y USO.

a. Una Copia por Equipo. Podrá instalar una copia del software en un equipo. Ese equipo será el "equipo licenciado".

b. Equipo Licenciado. Podrá utilizar simultáneamente el software en hasta dos procesadores del equipo licenciado. Salvo dispuesto de otro modo en estos términos de licencia, no podrá utilizar el software en ningún otro equipo.

c. Número de Usuarios. Salvo dispuesto de otro modo en estos términos de licencia, el software no podrá ser utilizado por más de un usuario a la vez.

d. Versiones Alternativas. El software puede incluir más de una versión, por ejemplo de 32 bits y de 64 bits. Sólo podrá instalar y utilizar una versión en un momento dado.

3. REQUISITOS DE LICENCIA Y/O DERECHOS DE USO ADICIONALES.

a. Multiplexado. El hardware o software que usted utilice para:

· agrupar conexiones, o bien

· reducir el número de dispositivos o usuarios que utilizan el software o tienen acceso directo al mismo (Operación que suele denominarse "multiplexado" o "agrupación"), no reduce el número de licencias necesarias.

b. Componentes de Fuente. Mientras se ejecuta el software, podrá utilizar sus fuentes para mostrar e imprimir el contenido. Solamente podrá:

· incrustar las fuentes en el contenido de acuerdo con las restricciones de incrustación de fuentes y

· descargarlas temporalmente en una impresora u otro dispositivo de salida para imprimir el contenido.

c. Iconos, Imágenes y Sonidos. Mientras se ejecuta el software, podrá utilizar pero no compartir sus iconos, imágenes, sonidos y soportes físicos. Los ejemplos de imágenes, sonidos y soportes físicos que se proporcionan con el software están destinados exclusivamente a un uso no comercial.

d. Uso con Tecnologías de Virtualización. En lugar de utilizar el software directamente en el equipo licenciado, podrá instalar y utilizar el software solamente en un sistema de hardware virtual (o emulado de cualquier otro modo) en el equipo licenciado. Cuando se utilice en un entorno virtualizado, es posible que el contenido protegido mediante una tecnología de administración de derechos digitales, BitLocker o cualquier tecnología de cifrado del volumen completo de la unidad de disco no sea tan seguro como el contenido protegido que no se encuentre en un entorno virtualizado. Debe cumplir con todas las leyes y disposiciones internacionales y nacionales que sean de aplicación a dicho contenido protegido.

e. Conexiones de Dispositivos. Podrá autorizar como máximo a otros 20 dispositivos a tener acceso al software instalado en el equipo licenciado para utilizar solamente Servicios de Archivo, Servicios de Impresión, Internet Information Server, Servicios de Conexión Compartida a Internet y Servicios de Telefonía.

f. Tecnologías de Acceso Remoto. Podrá obtener acceso de forma remota al software instalado en el equipo licenciado, y utilizarlo, desde cualquier otro equipo para compartir una sesión mediante la Asistencia Remota o tecnologías similares. Por “sesión” se entiende la experiencia de interactuar, directa o indirectamente, con el software a través de cualquier combinación de periféricos de entrada, salida y visualización.

4. ACTIVACIÓN OBLIGATORIA.

La activación asocia el uso del software a un equipo concreto. Durante la activación, el software enviará a Microsoft información sobre el propio software y el equipo. Esta información incluye la versión, el idioma y la clave de producto del software, la dirección de protocolo de Internet del equipo y la información derivada de la configuración del hardware del equipo. Para obtener más información, consulte go.microsoft.com/fwlink/?Linkid=104609. El uso del software constituirá un consentimiento por su parte para la transmisión de esta información. Si tiene la licencia apropiada, tiene derecho a utilizar la versión del software que se haya instalado durante el proceso de instalación hasta agotar el plazo permitido para la activación. Salvo que el software se active, no tendrá derecho a utilizar el software una vez transcurrido al plazo permitido para la activación. Se ha diseñado así para evitar su uso sin licencia. No está permitido evitar o eludir la activación. Si el equipo está conectado a Internet, el software podrá conectarse automáticamente con Microsoft para llevar a cabo la activación. También puede activar el software manualmente por teléfono o a través de Internet. En tal caso, es posible que deba abonar algún cargo por los servicios de Internet y telefónicos. Algunas de las modificaciones que pueda realizar en los componentes del equipo o del software podrían requerir la reactivación del software. El software le recordará la necesidad de activarlo mientras usted no lo haga**.**

5. VALIDACIÓN.

a. La validación comprueba que se ha activado el software y que cuenta con una licencia adecuada. También comprueba que no se han efectuado cambios no autorizados en las características de validación, licencia o activación del software. Asimismo, la validación puede comprobar la existencia de determinado software malintencionado o no autorizado en relación con esos cambios no autorizados. La validación de la licencia apropiada le permite seguir utilizando el software o algunas de sus características u obtener ventajas adicionales. **No está permitido eludir la validación.** Se ha diseñado así para evitar el uso del software sin licencia. Para obtener más información, consulte go.microsoft.com/fwlink/?Linkid=104610.

b. El software realizará de vez en cuando una comprobación de validación del propio software. Microsoft o el propio software podrán iniciar la comprobación. Para habilitar la característica de activación y las comprobaciones de validación, el software podrá requerir cuando estime oportuno actualizaciones o descargas adicionales de las funciones de validación, licencia o activación del software. Las actualizaciones o descargas son necesarias para que el software funcione

correctamente y se podrán descargar e instalar sin previo aviso. Durante o después de una comprobación de validación, el software podrá enviar a Microsoft información sobre el propio software, el equipo y los resultados de la comprobación de validación. Esta información incluye, por ejemplo, la versión y la clave de producto del software, las modificaciones no autorizadas en las funciones de validación, licencia o activación del software, cualquier software malintencionado o no autorizado relacionado que se haya encontrado y la dirección de protocolo de Internet del equipo. Microsoft no utilizará esta información para identificarle ni para ponerse en contacto con usted. El uso del software constituirá un consentimiento por su parte para la transmisión de esta información. Para obtener más información acerca de la validación y de los datos que se envían durante o tras una comprobación de validación, consulte go.microsoft.com/fwlink/?Linkid=104611.

c. Si, tras la comprobación de la validación, se descubre que se trata de software falsificado, que no dispone de la licencia apropiada, que no es un producto Windows original o que incluye modificaciones no autorizadas, la funcionalidad y el uso del software se verán afectados, por ejemplo:

Microsoft podrá:

· reparar el software, quitar, poner en cuarentena o deshabilitar las modificaciones no autorizadas susceptibles de interferir en el uso adecuado del software, lo que incluye eludir las funciones de activación o validación del software;

· comprobar y quitar el software malintencionado o no autorizado que esté relacionado con dichas modificaciones no autorizadas o

· notificar que el software no tiene la licencia apropiada o no es un producto Windows original.

Usted podrá:

· recibir avisos para obtener una copia licenciada adecuada del software o

· seguir las instrucciones de Microsoft a fin de obtener la licencia para utilizar el software y reactivarlo.

Usted no podrá:

· utilizar o seguir utilizando el software o algunas de sus características o

· obtener determinadas actualizaciones o mejoras de Microsoft.

d. Sólo se pueden obtener actualizaciones del software de Microsoft u otras fuentes autorizadas. Para obtener más información sobre cómo obtener actualizaciones de fuentes autorizadas, consulte go.microsoft.com/fwlink/?Linkid=104612.

Para el uso del software JURECU el colegio debe utilizar la licencia de SQL server 2008

SQL Server 2008 –Esquemas de Licenciamiento

SQL Server 2008 está disponible bajo dos modelos de licenciamiento:

LICENCIAMIENTO DE SERVIDOR SQL SERVER 2008 Y CAL

SQL Server 2008 ofrece diferentes licencias de Servidor (en Ediciones Workgroup, Standard y Enterprise) en conjunto con

Licencias de Acceso del Cliente (CALs) por dispositivo o por usuario.

Licencia de servidor

Se requiere una licencia de servidor (para ediciones Workgroup, Standard, o Enterprise) para cada ambiente de sistema operativo en el cual se ejecute dicha edición del software SQL Server o cualquiera de sus componentes (por ejemplo, Servicios de Análisis).).

CAL por dispositivo.

Se requiere una CAL de Dispositivo de SQL Server para que un dispositivo (por ejemplo, PC, estación de trabajo, terminal, PDA, teléfono celular, etcétera) pueda acceder o usar los servicios o la funcionalidad de Microsoft SQL Server. El modelo de Servidor más CAL de dispositivo probablemente es la opción más rentable si existen múltiples usuarios por dispositivo (por ejemplo, un centro de atención telefónica).

CAL por usuario.

Se requiere una CAL de Usuario de SQL Server para que un usuario (empleado, cliente, socio, etcétera) pueda acceder o usar los servicios o la funcionalidad de Microsoft SQL Server. El modelo de Servidor más CAL de Usuario probablemente es la opción más rentable si existen múltiples dispositivos por usuario (por ejemplo, un usuario que tiene un PC de escritorio, laptop, PDA, etcétera).

Una CAL no es un software; es un documento legal que otorga acceso al software del servidor a un dispositivo o un usuario.

Una CAL de dispositivo único otorga acceso a múltiples servidores para un dispositivo (la CAL debe ser la misma versión que la versión más reciente de cualquiera de los servidores). Una CAL de usuario único otorga acceso a múltiples servidores para el usuario.

LICENCIAMIENTO DE PROCESADOR SQL SERVER 2008

Microsoft ofrece un modelo de licenciamiento basado en procesador para ayudar a aligerar la complejidad o cuando el número de usuarios o dispositivos es alto

Licencia de procesador. Se requiere una Licencia de Procesador para cada procesador instalado en cada ambiente de sistema operativo que ejecute SQL Server o cualquier de sus componentes (por ejemplo, Servicios de Análisis). Esto incluye acceso para que un número ilimitado de usuarios o dispositivos se conecten ya sea desde dentro o fuera del firewall.

Los clientes no necesitan comprar Licencias de Servidor adicionales o Licencias de Acceso del Cliente (CALs) cuando obtienen el modelo de licenciamiento por procesador.

MÚLTIPLES NÚCLEOS

Los procesadores de múltiples núcleos, consistentes en múltiples unidades de ejecución de procesamiento o “núcleos” (cores) en un chip, se consideran una promisoria forma de elevar la potencia de cómputo. Microsoft ha sido el líder en esta área al cobrar la misma cantidad por procesador, sin importar cuántos núcleos haya en él.

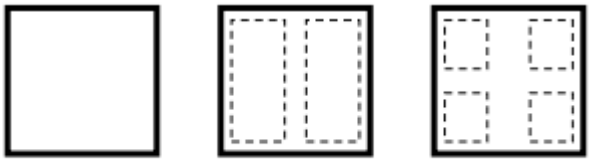


Ilustración 1 Múltiples Núcleos

En cada uno de estos escenarios se requiere una sola licencia de procesador para SQL Server, a diferencia de los requisitos de licenciamiento de Oracle e IBM.

Tabla 1 Licenciamiento de Oracle

Edicion	Descripción	Precio por procesador	Precio Server mas CALs
Enterprise Edition	Una plataforma completa de administración de datos e inteligencia de negocios la cual provee escalabilidad a nivel empresarial, alta disponibilidad y seguridad para correr las aplicaciones criticas de negocio	**Retail*** $24,999 **Ejemplo**** $23,911	**Retail*** $13,969 con 25 CALs **Ejemplo** $8,487, $162 por CAL adicional
Standard Edition	Una plataforma completa para administración de de datos e inteligencia de negocios la cual brinda la mayor facilidad de uso y de administración existentes en el mercado para correr aplicaciones departamentales..	**Retail*** $5,999 **Ejemplo**** $5,737	**Retail*** $1,849 con 5 CALs **Ejemplo **** $885 server, $162 por CAL adicional
Workgroup Edition	Una plataforma confiable de administración de datos y de reporteo que brinda capacidades de sincronización y administración remota segura para correr aplicaciones en sucursales.	**Retail*** $3,899 **Ejemplo **** $3,700	**Retail*** $739 con 5 CALs **Ejemplo **** $730 por servidor $146 por CAL adicional
Developer Edition	Puede ser instalada y utilizada por un usuario para diseñar, desarrollar, probar y demostrar programas en tantos sistemas como sea requeridos.	**Retail*** $50	**No aplica**
Web Edition	Una opción en bases de datos baja en TCO (costo total de propiedad), escalable y administrable para web hosters y consumidores finales que buscan implementar aplicaciones web y servicios de cara al público.	**$15 por proc mensual (SPLA) Ejemplo **** **$2862**	**No aplica**
Express Edition	Una edición gratuita de SQL-Server ideal para aprender y construir aplicaciones pequeñas para servidores y de escritorio, así como para la redistribución por Vendedores Independientes de Software (ISVs).	**No aplica**	**No aplica**
Compact Edition	Una base de datos gratuita de SQL Server embebida, ideal para construir aplicaciones stand-alone y ocasionalmente conectadas para dispositivos móviles, de escritorio y clientes web.	**No aplica**	**No aplica**
Evaluation Edition	Esta edición puede ser instalada con propósitos de evaluación y demostración hasta la fecha de expiración de 180 días..	**No aplica**	**No aplica**

*Todos los precios reflejan los precios de compra dentro de Estados Unidos y en U.S. dollars. Los precios listados son precios de retail estimados; todos los precios de reventa pueden variar.
** Estos son precios representativos para una compañía que compra una cantidad pequeña de Procesadores o Licencias de Servidores a través de las licencias por Volumen de Microsoft.SQL Server 2008 – Nuevas Funcionalidades
1. Administración basada en Políticas para simplificar la administración de las Bases de datos.
2. Encriptación transparente de datos y Auditoría avanzada para proteger datos sensibles.
3. Database Mirroring o Espejeo de Base de datos mejorado para brindar mayor disponibilidad.
4. Resource Governor para administrar cargas de trabajo simultaneas proveyendo un desempeño predecible
5. Performance Studio para el diagnóstico de problemas (troubleshoot), monitoreo y puesta a punto de instancias de SQL Server.
6. Nuevos tipos de datos para soporte de documentos, imágenes e información espacial.
7. Integración con Office para modificar y renderear reportes de SQL Server directamente en Office y Share Point.
8. Compresión de datos para mejorar el desempeño de consultas y reducir el espacio de almacenamiento.
9. Visualización de datos para obtener una visión rápida de conjuntos de datos complejos.
10. Synch Framework para la sincronización de datos online/offline.

3.5 MARCO TEÓRICO

Las herramientas metodológicas utilizadas para el desarrollo de este proyecto son:

ESTUDIO DE CAMPO

El día 14 de agosto de 2012 se hizo la visita al colegio Cafam La esperanza ubicado en el sector de Bosa donde se habló con el rector del colegio para la posibilidad de implementar un software educativo en este colegio donde se plantearon las siguientes preguntas a los docentes y coordinadora académica del colegio.

FORMATO ENCUESTA:

1- ¿De qué le gustaría que tratara el software educativo?
2- ¿A qué cursos le gustaría que fuera enfocado el software?
3- ¿Piensa que hay necesidad de implementar un software para los estudiantes?

FOTMATO ENTREVISTA:
RECTOR:

1- Se habló sobre la posibilidad de implementar un software educativo en el colegio.
2- De la misma manera se habló sobre cual ciclo del colegio debería estar enfocado el software.

COORDINADORA ACADEMICA:

1- Con base en la encuesta realizada a los docentes de (primer ciclo) se observó las falencias de los estudiantes en las asignaturas de castellano y matemáticas.
2- La coordinadora presento una sugerencia de diseñar un software educativo que implementara las regletas Cuisenaire en los estudiantes de primer ciclo.

DOCENTES:

1. Los docentes de primer ciclo a los que se les preguntaron sobre la temática que debería llevar el software educativo, contestaron las siguientes posibilidades:
    - Formando palabras
    - Oraciones
    - Operaciones básicas
        - Suma
        - Resta
        - Multiplicación
        - División
    - Textos con imágenes

4 MODELOS DE DATOS

4.1 MODELO ENTIDAD RELACION

El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad, mediante un conjunto de representaciones gráficas y lingüísticas.
El este modelo es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades

A continuación se muestra el modelo entidad-relación establecido en el proyecto.

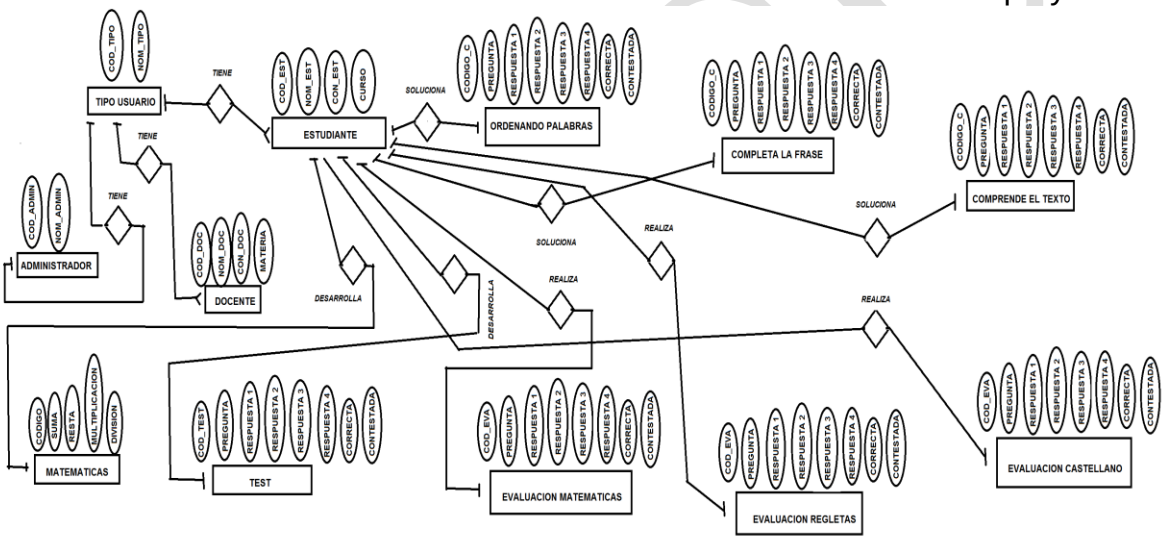


Ilustración 2 modelo entidad relación
Se muestra los atributos correspondientes a las tablas y la relación de una tabla con la otra.

4.2 MODELO TABULAR

Codigo	Nombre	Contraseña	Curso

Codigo	Suma	Resta	Multiplicacion	Division

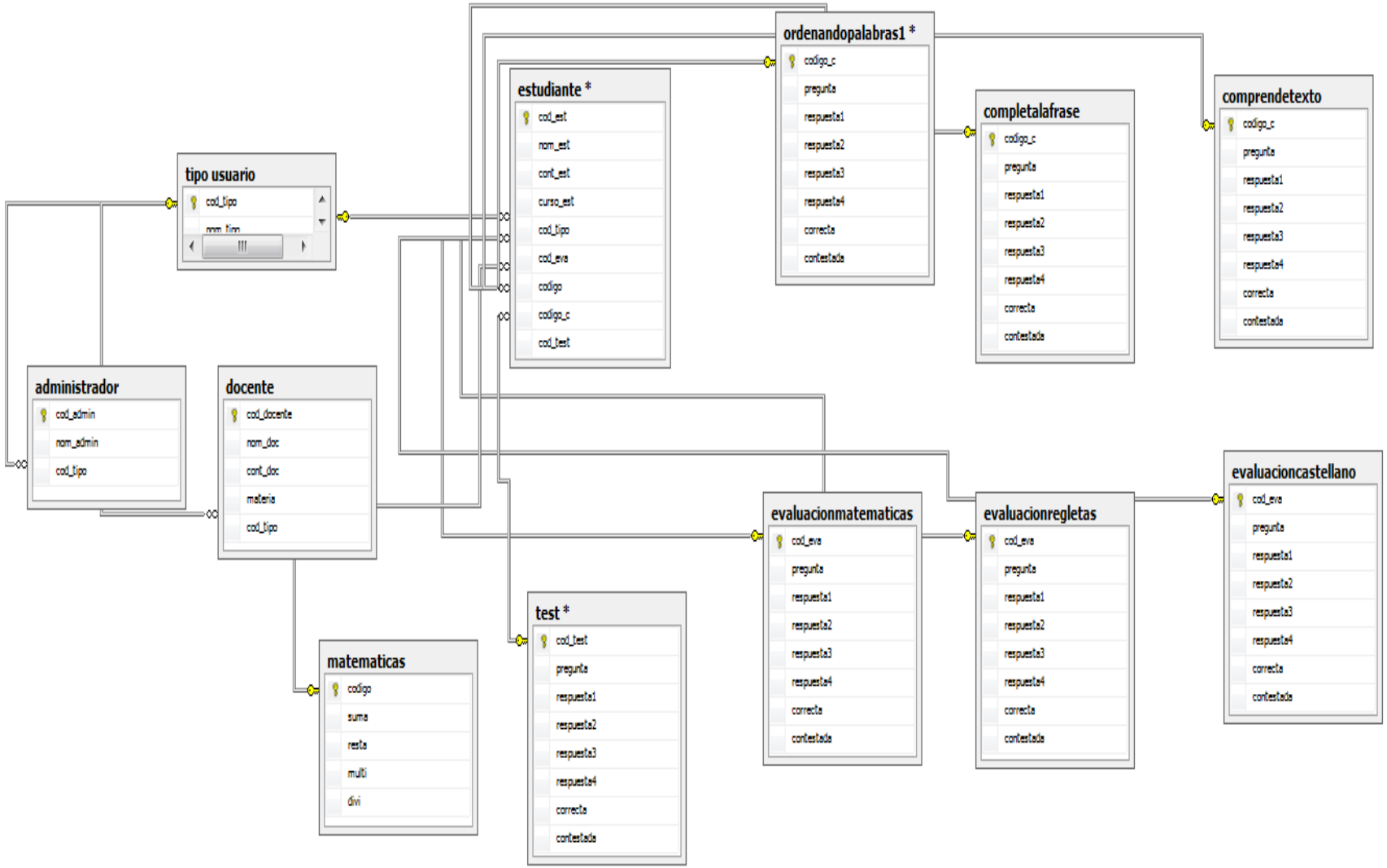
Codigo	Pregunta	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Correcta
Codigo	Pregunta	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Correcta
Codigo	Pregunta	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Correcta
Codigo	Pregunta	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Correcta
Codigo	Pregunta	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Correcta
Codigo	Pregunta	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Correcta
Codigo	Pregunta	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Correcta

Ilustración 3 Modelo Tabular

4.3 MODELO RELACIONAL

Se trata de un modelo lógico [Irene Luque Ruiz- Ed. Ra-ma], que establece una estructura sobre los datos, aunque posteriormente éstos puedan ser almacenados de múltiples formas para aprovechar características físicas concretas de la máquina sobre la que se implante la base de datos realmente

Ilustración 4 modelo relacional
Se encuentra el modelo relacional


tipo usuario
cod_tipo
administrador
cod_admin
nom_admin
cod_tipo
docente
cod_docente
nom_doc
cont_doc
materia
cod_tipo
estudiante *
cod_est
nom_est
cont_est
curso_est
cod_tipo
cod_eva
codigo
codigo_c
cod_test
ordenandopalabras1 *
codigo_c
pregunta
respuesta1
respuesta2
respuesta3
respuesta4
correcta
contestada
completalafrase
codigo_c
pregunta
respuesta1
respuesta2
respuesta3
respuesta4
correcta
contestada
comprendetexto
codigo_c
pregunta
respuesta1
respuesta2
respuesta3
respuesta4
correcta
contestada
matematicas
codigo
suma
resta
multi
divi
test *
cod_test
pregunta
respuesta1
respuesta2
respuesta3
respuesta4
correcta
contestada
evaluacionmatematicas
cod_eva
pregunta
respuesta1
respuesta2
respuesta3
respuesta4
correcta
contestada
evaluacionregletas
cod_eva
pregunta
respuesta1
respuesta2
respuesta3
respuesta4
correcta
contestada
evaluacioncastellano
cod_eva
pregunta
respuesta1
respuesta2
respuesta3
respuesta4
correcta
contestada

5 CICLO DE VIDA DEL SOFTWARE

5.1 MODELO EN CASCADA

Es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo se ve fluyendo hacia abajo (como una cascada) sobre las fases que componen el ciclo de vida.

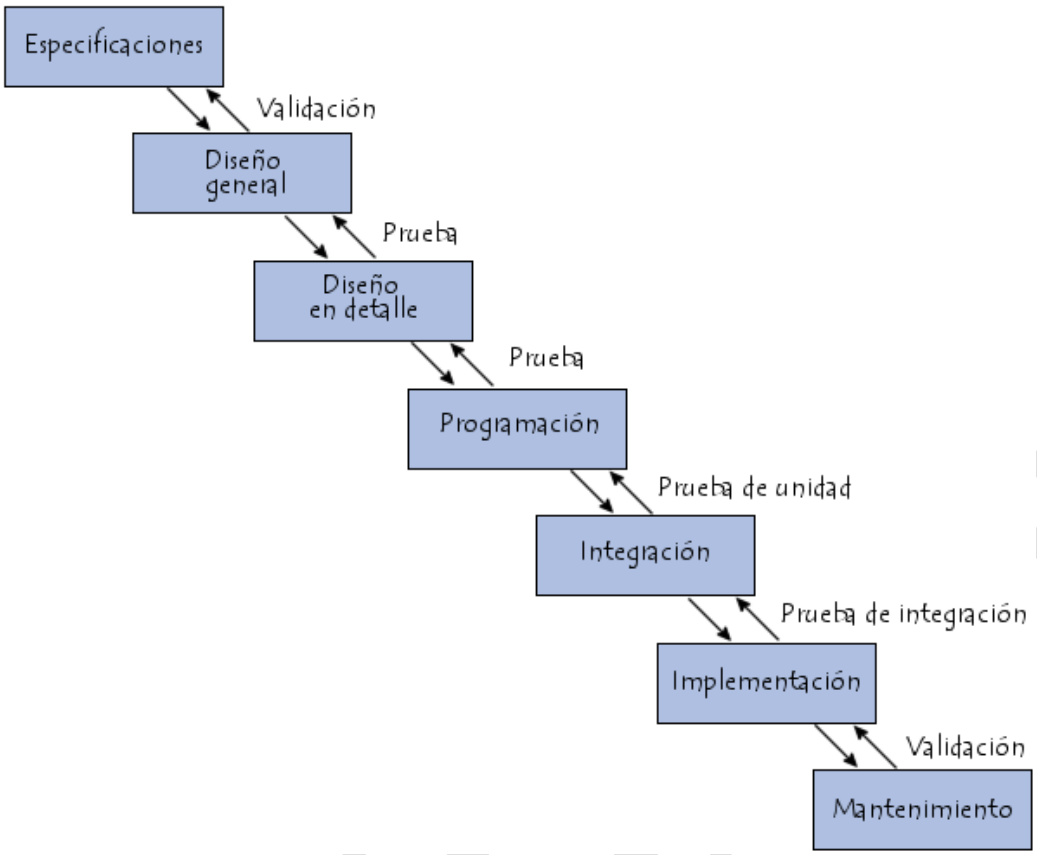
Se cree que el modelo en cascada fue el primer modelo de proceso introducido y seguido ampliamente en la ingeniería el software. La innovación estuvo en la primera vez que la ingeniería del software fue dividida en fases separadas.

La primera descripción formal del modelo en cascada se cree que fue en un artículo publicado en 1970 por Winston W. Royce, aunque Royce no usó el término cascada en este artículo. Irónicamente, Royce estaba presentando este modelo como un ejemplo de modelo que no funcionaba, defectuoso.

En el modelo original de Royce, existían las siguientes fases:

1. Especificación de requisitos
2. Diseño
3. Construcción (o codificación)
4. Implementación
5. Pruebas
6. Instalación
7. Mantenimiento

Para seguir el modelo en cascada, se avanza de una fase a la siguiente en una forma puramente secuencial.



Si bien ha sido ampliamente criticado desde el ámbito académico y la industria, sigue siendo el paradigma más seguido a día de hoy.

Ventajas

El modelo en cascada puede ser apropiado, en general, para proyectos estables (especialmente los proyectos con requisitos no cambiantes) y donde es posible y probable que los diseñadores predigan totalmente áreas de problema del sistema y produzcan un diseño correcto antes de que empiece la implementación. Funciona bien para proyectos pequeños donde los requisitos están bien entendidos.

Es un modelo en el que todo está bien organizado y no se mezclan las fases. Es simple y fácil de usar.

Debido a la rigidez del modelo es fácil de gestionar ya que cada fase tiene entregables específicos y un proceso de revisión. Las fases son procesadas y completadas de una vez.

Inconvenientes

En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.

Difícilmente un cliente va a establecer al principio todos los requisitos necesarios, por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite movilizarse entre fases.

Los resultados y/o mejoras no son visibles progresivamente, el producto se ve cuando ya está finalizado, lo cual provoca una gran inseguridad por parte del cliente que quiere ir viendo los avances en el producto. Esto también implica el tener que tratar con requisitos que no se habían tomado en cuenta desde el principio, y que surgieron al momento de la implementación, lo cual provocará que haya que volver de nuevo a la fase de requisitos.
Hay muchas personas que argumentan que es una mala idea en la práctica, principalmente a causa de su creencia de que es imposible, para un proyecto no trivial, conseguir tener una fase del ciclo de vida del producto software perfecta antes de moverse a las siguientes fases. Por ejemplo, los clientes pueden no ser conscientes exactamente de los requisitos que quieren antes de ver un prototipo del trabajo; pueden cambiar los requisitos constantemente, y los diseñadores e implementadores pueden tener poco control sobre esto. Si los clientes cambian sus requisitos después de que el diseño está terminado, este diseño deberá ser modificado para acomodarse a los nuevos requisitos, invalidando una buena parte del esfuerzo.
Muchas veces se considera un modelo pobre para proyectos complejos, largos, orientados a objetos y por supuesto en aquellos en los que los requisitos tengan un riesgo de moderado a alto de cambiar. Genera altas cantidades de riesgos e incertidumbres.

Variantes

Existen muchas variantes de este modelo. En respuesta a los problemas percibidos con el modelo en cascada puro, se introdujeron muchos modelos de cascada modificados. Estos modelos pueden solventar algunas o todas las críticas del modelo en cascada puro.
De hecho muchos de los modelos utilizados tienen su base en el modelo en cascada.
Uno de estos modelos modificados es el modelo **Sashimi.**
El modelo sashimi (llamado así porque tiene fases solapadas, como el pescado japonés sashimi) fue creado originalmente por Peter DeGrace. A veces se hace referencia a él como el modelo en cascada con fases superpuestas o el modelo en cascada con retroalimentación. Ya que las fases en el modelo sashimi se superponen, lo que implica que se puede actuar durante las etapas anteriores. Por ejemplo, ya que las fases de diseño e implementación se superpondrán en el modelo sashimi, los problemas de implementación se pueden descubrir durante las fases de diseño e implementación del proceso de desarrollo. Esto ayuda a aliviar muchos de los problemas asociadas con la filosofía del modelo en cascada.

5.2 MODELO EN ESPIRAL

El desarrollo en espiral es un modelo de ciclo de vida desarrollado por Barry Boehm en 1985, utilizado de forma generalizada en la ingeniería del software. Las

actividades de este modelo se conforman en una espiral, cada bucle representa un conjunto de actividades. Las actividades no están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgos, comenzando por el bucle anterior.

Boehm, autor de diversos artículos de ingeniería del software; modelos de estimación de esfuerzos y tiempo que se consume en hacer productos software; y modelos de ciclo de vida, ideó y promulgó un modelo desde un enfoque distinto al tradicional en Cascada: el Modelo Evolutivo Espiral. Su modelo de ciclo de vida en espiral tiene en cuenta fuertemente el riesgo que aparece a la hora de desarrollar software. Para ello, se comienza mirando las posibles alternativas de desarrollo, se opta por la de riesgos más asumibles y se hace un ciclo de la espiral. Si el cliente quiere seguir haciendo mejoras en el software, se vuelven a evaluar las nuevas alternativas y riesgos y se realiza otra vuelta de la espiral, así hasta que llegue un momento en el que el producto software desarrollado sea aceptado y no necesite seguir mejorándose con otro nuevo ciclo.

Este modelo de desarrollo combina las características del modelo de prototipos y el modelo en cascada. El modelo en espiral está pensado para proyectos largos, caros y complicados.

Esto modelo no fue el primero en tratar el desarrollo iterativo, pero fue el primer modelo en explicar las iteraciones.

Este modelo fue propuesto por Boehm en 1988 en su artículo “A Spiral Model of Software Development and Enhancement”. Básicamente consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. Se suele interpretar como que dentro de cada ciclo de la espiral se sigue un modelo en cascada, pero no necesariamente ha de ser así.

Este sistema es muy utilizado en proyectos grandes y complejos como puede ser, por ejemplo, la creación de un sistema operativo.

Al ser un modelo de ciclo de vida orientado a la gestión de riesgos se dice que uno de los aspectos fundamentales de su éxito radica en que el equipo que lo aplique tenga la necesaria experiencia y habilidad para detectar y catalogar correctamente riesgos.

Tareas:

Para cada ciclo habrá cuatro actividades:

1.Determinar o fijar objetivos:
   a.Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
   b.Fijar las restricciones
   c.Identificación de riesgos del proyecto y estrategias alternativas para evitarlos
   d.Hay una cosa que solo se hace una vez: planificación inicial o previa
2.Análisis del riesgo:
   a.Se estudian todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos
3.Desarrollar, verificar y validar (probar):
   a.Tareas de la actividad propia y de prueba
   b.Análisis de alternativas e identificación de resolución de riesgos

c.Dependiendo del resultado de la evaluación de riesgos, se elige un modelo para el desarrollo, que puede ser cualquiera de los otros existentes, como formal, evolutivo, cascada, etc. Así, por ejemplo, si los riesgos de la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos.

4.Planificar:

a.Revisamos todo lo que hemos hecho, evaluándolo y con ello decidimos si continuamos con las fases siguientes y planificamos la próxima actividad.

El proceso empieza en la posición central. Desde allí se mueve en el sentido de las agujas del reloj.

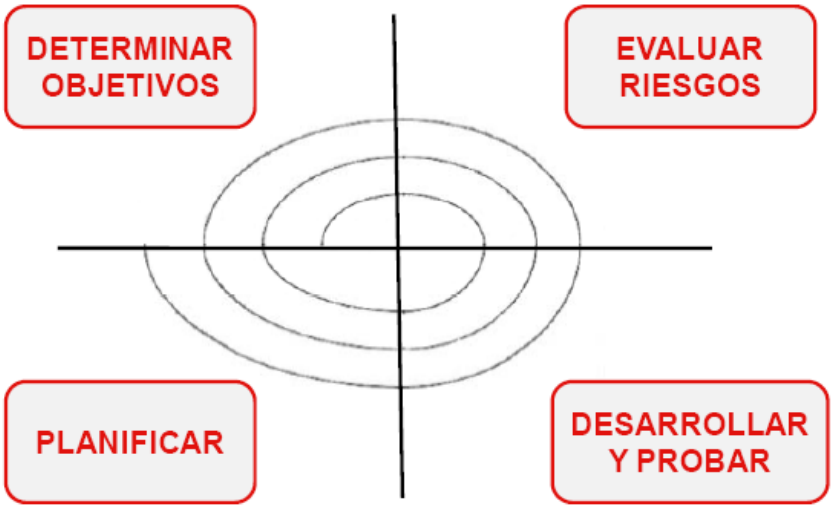


Ilustración 5 modelo en espiral

Las cuatro regiones del gráfico son:

-La tarea de planificación: para definir recursos, responsabilidades, hitos y planificaciones

-La tarea de determinación de objetivos: para definir los requisitos y las restricciones para el producto y definir las posibles alternativas

-La tarea de análisis de riesgos: para evaluar riesgos tanto técnicos como de gestión

-La tarea de ingeniería: para diseñar e implementar uno o más prototipos o ejemplos de la aplicación

Ventajas
El análisis de riesgos se hace de forma explícita y clara. Une los mejores elementos de los restantes modelos. Entre ellos:
-Reduce riesgos del proyecto
-Incorpora objetivos de calidad
-Integra el desarrollo con el mantenimiento
Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con el modelo, ya que el ciclo de vida no es rígido ni estático.
Mediante este modelo se produce software en etapas tempranas del ciclo de vida y suele ser adecuado para proyectos largos de misión crítica.

Inconvenientes
Es un modelo que genera mucho trabajo adicional. Al ser el análisis de riesgos una de las tareas principales exige un alto nivel de experiencia y cierta habilidad en los analistas de riesgos (es bastante difícil).

Comparación con otros modelos
La distinción más destacada entre el modelo en espiral y otros modelos de software es la tarea explícita de evaluación de riesgos. Aunque la gestión de riesgos es parte de otros procesos también, no tiene una representación propia en el modelo de proceso. Para otros modelos la evaluación de riesgos es una sub-tarea, por ejemplo, de la planificación y gestión global. Además no hay fases fijadas para la especificación de requisitos, diseño y pruebas en el modelo en espiral. Se puede usar prototipo para encontrar y definir los requisitos.
La diferencia entre este modelo y el modelo de ciclo incremental es que en el incremental se parte de que no hay incertidumbre en los requisitos iniciales; en este, en cambio, se es consciente de que se comienza con un alto grado de incertidumbre. En el incremental suponemos que conocemos el problema y lo dividimos. Este modelo gestiona la incertidumbre.

5.3 MODELO ORIENTADO A OBJETOS

Los tipos de ciclos de vida que se han visto hasta ahora son relativos al análisis y diseño estructurados, pero los objetos tienen una particularidad, y es que están basados en componentes que se relacionan entre ellos a través de interfaces, o lo que es lo mismo, son más modulares y por lo tanto el trabajo se puede dividir en un conjunto de mini-proyectos. Además, hoy en día la tendencia es a reducir los riesgos, y en este sentido, el ciclo de vida en cascada no proporciona muchas facilidades. Debido a todo esto, el ciclo de vida típico en una metodología de diseño orientado a objetos es iterativo e incremental.
En este texto sólo veremos un tipo de ciclo de vida orientado a objetos, que es además el más representativo, el modelo fuente.

Modelo fuente

Fue creado por Henderson-Sellers y Edwards en 1990. Es un tipo de ciclo de vida pensado para la orientación a objetos y posiblemente el más seguido. Un proyecto se divide en las fases:

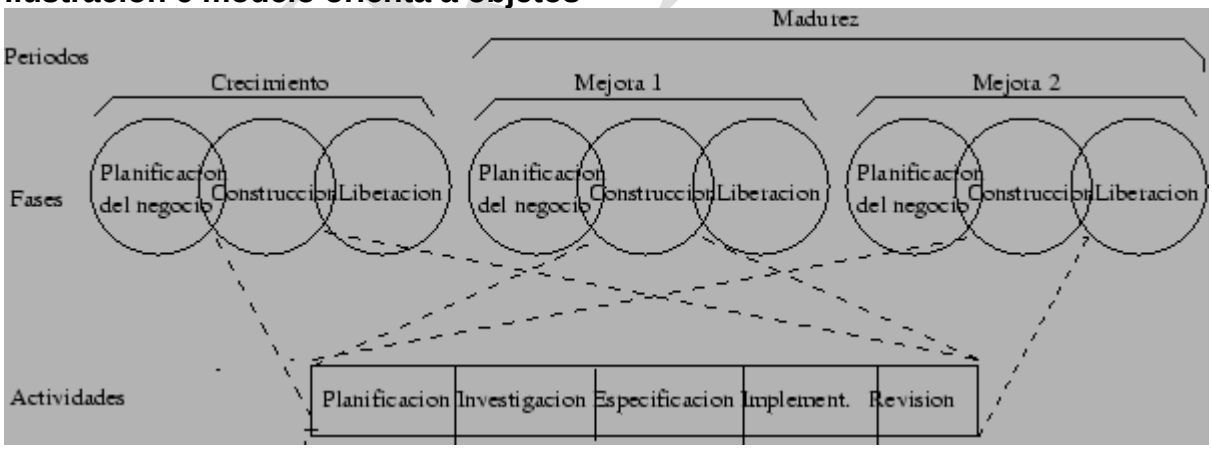
1. Planificación del negocio
2. Construcción: Es la más importante y se divide a su vez en otras cinco actividades
   - Planificación
   - Investigación
   - Especificación
   - Implementación
   - Revisión
3. Entrega

La primera y la tercera fase son independientes de la metodología de desarrollo orientado a objetos. Además de las tres fases, existen dos periodos:

1. Crecimiento: Es el tiempo durante el cual se construye el sistema
2. Madurez: Es el periodo de mantenimiento del producto. Cada mejora se planifica igual que el periodo anterior, es decir, con las fases de Planificación del negocio, Construcción y Entrega.

Cada clase puede tener un ciclo de vida sólo para ella debido a que cada una puede estar en una fase diferente en un momento cualquiera. La ventaja es que permite un desarrollo solapado e iterativo. En la siguiente figura se muestra un esquema de este tipo de ciclo de vida.

Ilustración 6 modelo orienta a objetos



Para el desarrollo del software "JURECU" se va utilizar e implementar el ciclo de vida en cascada

En el cual el enfoque metodológico ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior. Este ciclo de vida está acorde con el desarrollo de dicho software presentado, donde desde el inicio de la planeación del software se han empezado con las distintas etapas que nos menciona el método en cascada, este modelo de ciclo de vida está enfocado en general, para proyectos estables y donde es posible que los diseñadores predigan totalmente áreas de problema del sistema y produzcan un diseño correcto antes de que empiece la implementación. Este ciclo funciona bien con proyectos en los que desde el inicio se establecen los requisitos para su funcionamiento y diseño.

METODOLOGIA DESARROLLO DEL PROYECTO

UML

(Unified Modeling Language - Lenguaje Unificado de Modelado). UML es un popular lenguaje de modelado de sistemas de software. Se trata de un lenguaje gráfico para construir, documentar, visualizar y especificar un sistema de software. Entre otras palabras, UML se utiliza para definir un sistema de software.

Posee la riqueza suficiente como para crear un modelo del sistema, pudiendo modelar los procesos de negocios, funciones, esquemas de bases de datos, expresiones de lenguajes de programación, etc. Para ello utiliza varios tipos diferentes de diagramas, por ejemplo, en UML 2.0 hay 13 tipos de diagramas. Estos diagramas se pueden diferenciar en tres categorías:
- Diagramas de estructura:
Diagrama de clases
Diagrama de componentes
Diagrama de objetos
Diagrama de estructura compuesta (UML 2.0)
Diagrama de despliegue
Diagrama de paquetes

-Diagramas de comportamiento:

Diagrama de actividades
Diagrama de casos de uso
Diagrama de estados

-Diagramas de interacción:
Diagrama de secuencia
Diagrama de comunicación
Diagrama de tiempos (UML 2.0)
Diagrama de vista de interacción (UML 2.0)

RUP (Proceso Unificado de Rational)

RUP es un proceso para el desarrollo de un proyecto de software que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto, con 3 características esenciales, está dirigido por los **casos de Uso:** que orientan el proyecto a la importancia para el usuario y lo que este quiere, está centrado en **la arquitectura**: que Relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden, y es **iterativo e incremental**: dividiéndose el proyecto en mini proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada.

También se conoce por este nombre al software desarrollado por Rational, hoy propiedad de IBM, el cual incluye información entrelazada de diversos artefactos y descripciones de las diversas actividades. Está incluido en el Rational Method Composer (RMC), que permite la personalización de acuerdo a necesidades.

Originalmente se diseñó un proceso genérico y de dominio público, el Proceso Unificado, y una especificación más detallada, el Rational Unified Process, que luego se vendiera como producto independiente. Este maneja 6 principios clave:

1) **Adaptación del proceso**
   El proceso deberá adaptarse a las características propias de la organización. El tamaño del mismo, así como las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.
2) **Balancear prioridades**
   Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.
3) **Colaboración entre equipos**
   El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, etc.
4) **Demostrar valor iterativamente**

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

5) **Elevar el nivel de abstracción**
   Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o esquemas (frameworks) por nombrar algunos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.
6) **Enfocarse en la calidad**
   El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

CICLO DE VIDA DE RUP

RUP divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

FASE DE INICIO

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar que recursos deben ser asignados al proyecto. Los objetivos de esta fase son:
•Establecer el ámbito del proyecto y sus límites.
•Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
•Mostrar al menos una arquitectura candidata para los escenarios principales.
•Estimar el coste en recursos y tiempo de todo el proyecto.
•Estimar los riesgos, las fuentes de incertidumbre.

Los resultados de la fase de inicio deben ser:

•Un documento de visión: Una visión general de los requerimientos del proyecto, características clave y restricciones principales.
•Modelo inicial de Casos de Uso (10-20% completado).
•Un glosario inicial: Terminología clave del dominio.
•El caso de negocio.
•Lista de riesgos y plan de contingencia.
•Plan del proyecto, mostrando fases e iteraciones.
•Modelo de negocio, si es necesario

•Prototipos exploratorios para probar conceptos o la arquitectura candidata.

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

•Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.
•Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales.
•Las estimaciones de tiempo, coste y riesgo son creíbles.
•Comprensión total de cualquier prototipo de la arquitectura desarrollado.
•Los gastos hasta el momento se asemejan a los planeados

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente

FASE DE ELABORACION

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto ye liminar los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe de mostrarse que se han evitado los riesgos más graves. Los objetivos de esta fase son:
•Definir, validar y cimentar la arquitectura.
•Completar la visión.
•Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.
•Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable. Al terminar deben obtenerse los siguientes resultados:
•Un modelo de Casos de Uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
•Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un Caso de Uso específico.
•Descripción de la arquitectura software.
•Un prototipo ejecutable de la arquitectura.
•Lista de riesgos y caso de negocio revisados.
•Plan de desarrollo para el proyecto.
•Un caso de desarrollo actualizado que especifica el proceso a seguir.
•Un manual de usuario preliminar (opcional).En esta fase se debe tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes.

En la fase de elaboración se actualizan todos los productos de la fase de inicio. Los criterios de evaluación de esta fase son los siguientes:
•La visión del producto es estable.
•La arquitectura es estable.
•Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
•El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
•Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.
•Los gastos hasta ahora son aceptables, comparados con los previstos
Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

FASE DE CONSTRUCCION

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto. Los objetivos concretos según [KRU00] incluyen:
•Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
•Conseguir una calidad adecuada tan rápido como sea práctico.
•Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba)tan rápido como sea práctico. Los resultados de la fase de construcción deben ser [RSC98]:
•Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
•Arquitectura íntegra (mantenida y mínimamente actualizada)
•Riesgos Presentados Mitigados
•Plan del Proyecto para la fase de Transición.
•Manual Inicial de Usuario (con suficiente detalle)
•Prototipo Operacional – beta
•Caso del Negocio Actualizado

Los criterios de evaluación de esta fase son los siguientes:

•El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
•Todos los usuarios expertos están listos para la transición en la comunidadde usuarios.
•Son aceptables los gastos actuales versus los gastos planeados.

FASE DE TRANSICION

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto. Algunas de las cosas que puede incluir esta fase:
•Prueba de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios
•Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por nuestro proyecto.
•Conversión de las bases de datos operacionales.
•Entrenamiento de los usuarios y técnicos de mantenimiento.
•Traspaso del producto a los equipos de marketing, distribución y venta. Los principales objetivos de esta fase son:
•Conseguir que el usuario se valga por si mismo.
•Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario. Los resultados de la fase de transición son
•Prototipo Operacional
•Documentos Legales
•Caso del Negocio Completo
•Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
•Descripción de la Arquitectura completa y corregida
•Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión. Los criterios de evaluación de esta fase son los siguientes:
•El usuario se encuentra satisfecho.
Son aceptables los gastos actuales versus los gastos planificados.

XP (programación extrema)

Metodología ágil

Las metodologías ágiles (como por ejemplo XP, SCRUM, DSDM, Crystal, etc.) forman parte del movimiento de desarrollo ágil de software, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto.
De forma que una metodología ágil es la que tiene como principios que:
• Los individuos y sus interacciones son más importantes que los procesos y las herramientas.
• El software que funciona es más importante que la documentación exhaustiva.
• La colaboración con el cliente en lugar de la negociación de contratos.
• La respuesta delante del cambio en lugar de seguir un plan cerrado.
Se puede decir que, este movimiento empezó a existir a partir de febrero de 2001, cuando se reunieron los representantes de cada una de estas metodologías y terminaron poniendo en común sus ideas en una declaración conjunta.

Definición de XP

La programación extrema es una metodología de desarrollo ligera (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.
Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.
Una de las características principales de este método de programación, es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El resultado de esta selección ha sido esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, sí que el resultado es una nueva manera de ver el desarrollo de software.
El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos. Aplicando el sentido común.

Planificación del proyecto.

En este punto se tendrá que elaborar la planificación por etapas, donde se aplicarán diferentes iteraciones. Para hacerlo será necesaria la existencia de reglas que se han de seguir por las partes implicadas en el proyecto para que todas las partes tengan voz y se sientan realmente partícipes de la decisión tomada.
Las entregas se tienen que hacer cuanto antes mejor, y con cada iteración, el cliente ha de recibir una nueva versión. Cuanto más tiempo se tarde en introducir una parte esencial, menos tiempo se tendrá para trabajar con ella después. Se aconseja muchas entregas y muy frecuentes. De esta manera un error en la parte inicial del sistema tiene más posibilidades de detectarse rápidamente. Una de las máximas a aplicar es, los cambios, no han de suponer más horas de programación para el programador, ya que el que no se termina en un día, se deja para el día siguiente.
Se ha de tener asumido que en el proceso de planificación habrán errores, es más, serán comunes, y por esto esta metodología ya los tiene previstos, por lo tanto se establecerán mecanismos de revisión. Cada tres o cinco iteraciones es normal revisar las historias de los usuarios, y renegociar la planificación.
Cada iteración necesita también ser planificada, es lo que se llama planificación iterativa, en la que se anotarán las historias de usuarios que se consideren esenciales y las que no han pasado las pruebas de aceptación. Estas planificaciones también se harán en tarjetas, en las que se escribirán los trabajos que durarán entre uno y tres días.

Es por esto que el diseño se puede clasificar como continuo. Añade agilidad al proceso de desarrollo y evita que se mire demasiado hacia delante, desarrollando trabajos que aún no han estado programados.
Este tipo de planificación en iteraciones y el diseño iterativo, hace que aparezca una práctica que no existía en la programación tradicional. Se trata de las discusiones diarias informales, para fomentar la comunicación, y para hacer que los desarrolladores tengan tiempo de hablar de los problemas a los que se enfrentan y de ver cómo van con sus trabajos.

Diseño, desarrollo y pruebas.

El desarrollo es la parte más importante en el proceso de la programación extrema. Todos los trabajos tienen como objetivo que se programen lo más rápidamente posible, sin interrupciones y en dirección correcta.
También es muy importante el diseño, y se establecen los mecanismos, para que éste sea revisado y mejorado de manera continuada a lo largo del proyecto, según se van añadiendo funcionalidades al mismo.
La clave del proceso de desarrollar XP es la comunicación. La mayoría de los problemas en los proyectos son por falta de comunicación en el equipo.
En XP, aparece un nuevo concepto llamado Metáfora. Su principal objetivo es mejorar la comunicación entre todos los integrantes del equipo, al crear una visión global y común de lo que se quiere desarrollar. La metáfora tiene que ser expresada en términos conocidos por los integrantes del equipo, por ejemplo comparando el sistema que se desarrollará con alguna cosa de la vida real.
Antes de empezar a codificar se tienen que hacer pruebas unitarias, es decir:
Cada vez que se quiere implementar una parte de código, en XP, se tiene que escribir una prueba sencilla, y después escribir el código para que la pase. Una vez pasada se amplía y se continúa. En XP hay una máxima que dice "Todo el código que puede fallar tiene que tener una prueba".
Con estas normas se obtiene un código simple y funcional de manera bastante rápida. Por esto es importante pasar las pruebas al 100%.
Respecto a la integración del software, en XP se ha de hacer una integración continua, es decir, cada vez se tienen que ir integrando pequeños fragmentos de código, para evitar que al finalizar el proyecto se tenga que invertir grandes esfuerzos en la integración final. En todo buen proyecto de XP, tendría que existir una versión al día integrada, de manera que los cambios siempre se realicen en esta última versión.
Otra peculiaridad de XP es que cada programador puede trabajar en cualquier parte del programa. De esta manera se evita que haya partes "propietarias de cada programador". Por esto es tan importante la integración diaria.
Para terminar, otra peculiaridad que tiene la XP. La de fomentar la programación en parejas, es decir, hacer que los programadores no trabajen en solitario, sino que siempre estarán con otra persona. Una pareja de programadores ha de compartir el teclado, el monitor y el ratón. El principio fundamental de este hecho

es realizar de manera continua y sin parar el desarrollo de código. Las parejas tienen que ir cambiando de manera periódica, para hacer que el conocimiento se difunda en el grupo. Está demostrado que de esta manera el trabajo es más eficaz y también se consigue más y mejor código.

CICLO DE VIDA JURECU

- **Especificación de requisitos**

- Tener una herramienta digital que permita el uso de las regletas Cuisenaire
-El software contiene las operaciones básicas (suma, resta, multiplicación, división)
-El software contiene temas relacionados con la asignatura castellano.
-El software permite evaluar al estudiante para que el docente pueda observar el desempeño del alumno después de haber realizado el test

- **Diseño:** Se realizaron los todos los diagramas de casos de uso , secuencia y clases sobre la investigación.

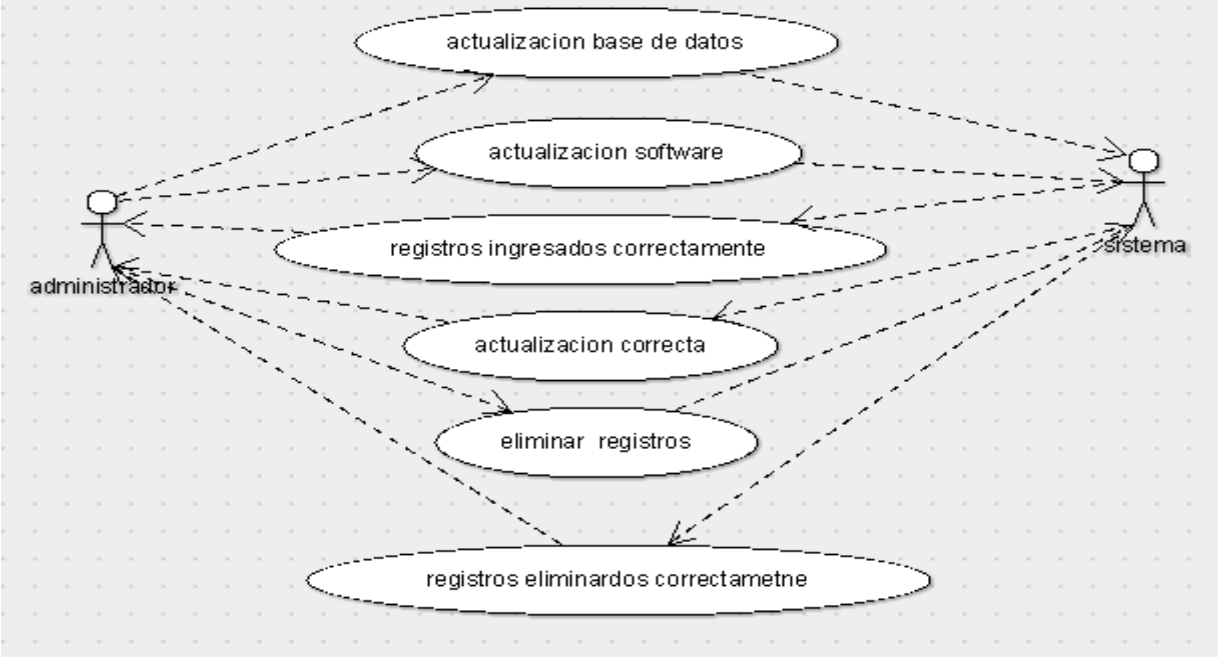
- **Construcción o codificación**
  La construcción del software se realizó sobre visual studio .net 2008 en lenguaje orientado a objetos, el diseño de la base de datos que usa el software esta realizado en sql server 2008.
- **Integración o implementación**
  La implementación será realizada en las diferentes salas informáticas que se encuentren disponibles en el colegio Cafam
- **Pruebas**
  El software jurecu fue sometido a distintas pruebas, una de ellas fue la de someter a las distintas aplicaciones que posee el software usándolas en varias ocasiones permitiendo verificar que el sistema no sufra ningún colapso, en el ingreso de los respectivos datos a la base de datos como también para verificar los resultados que arrojan sus aplicaciones.
- **Instalación**
  La instalación respectiva del software es de una manera rápida y eficaz la cual permite que el usuario pueda realizarla sin ningún problema con

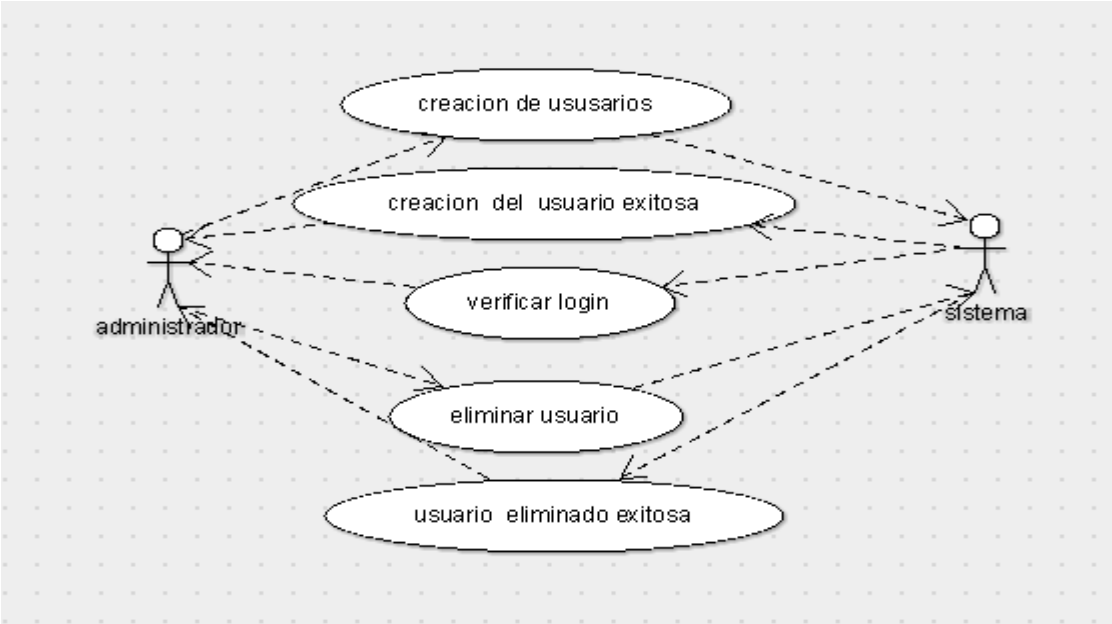
ayuda del manual de usuario en el que se explicará paso a paso su respectiva instalación.

- **Mantenimiento**
  El mantenimiento que será sometido el software JURECU se realizara en sus distintos módulos que lo integran, para asi poder optimar su rendimiento, de la misma manera la base de datos que usa el software sufrirá un mantenimiento en sus respectivos registros manejos para que no sufra ningún colapso en el ingreso de los datos.

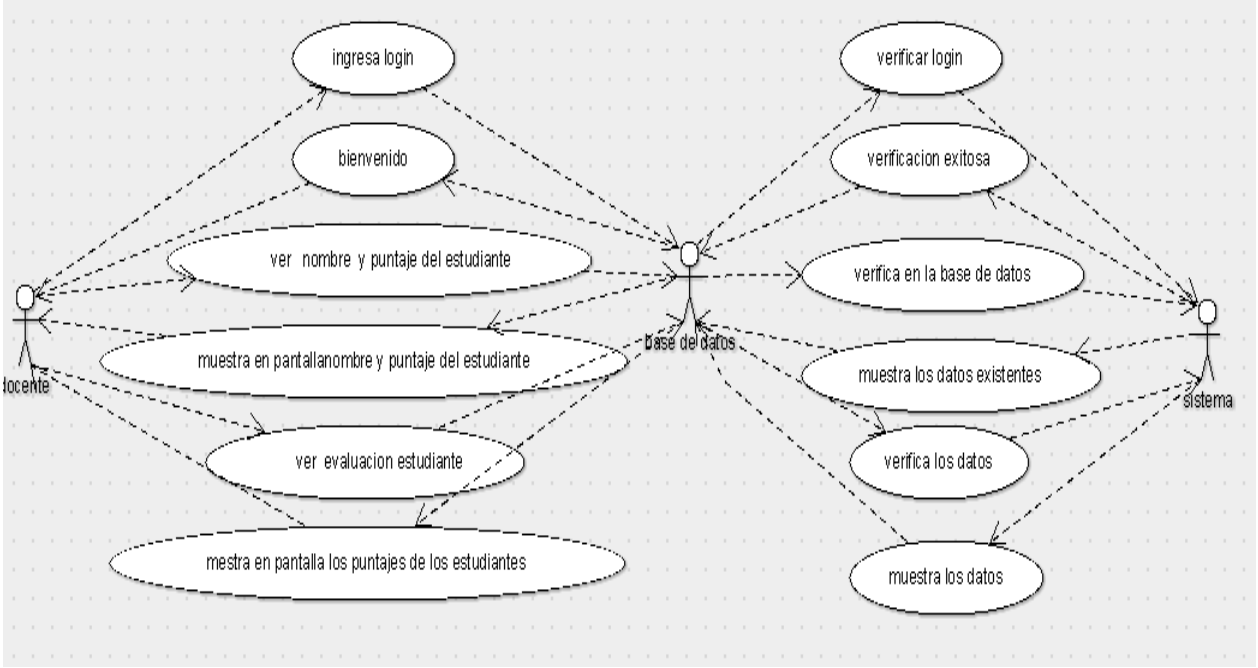
Diagramas de casos de uso
Casos de uso del administrador

Se puede observar la interacción del **administrador** con el sistema
Ilustración 7 diagramas de casos de uso



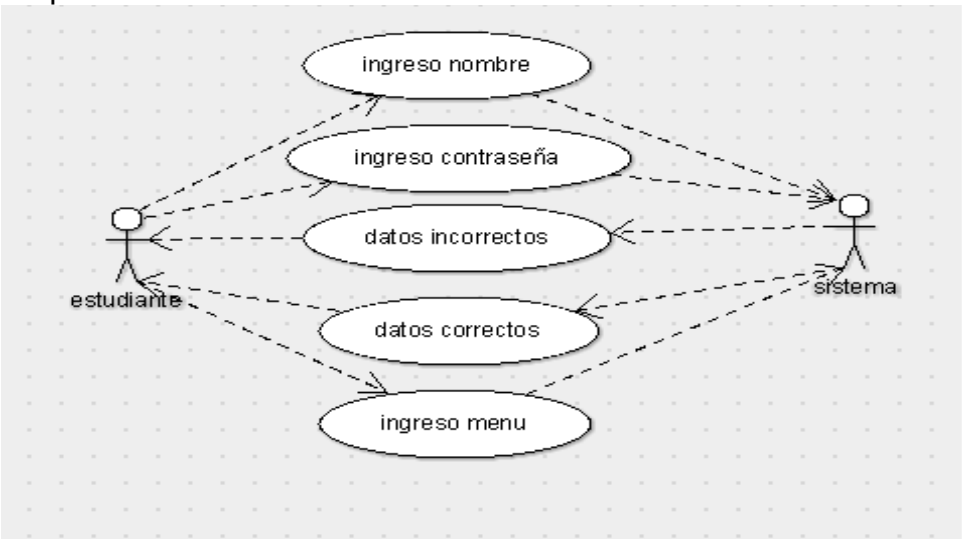


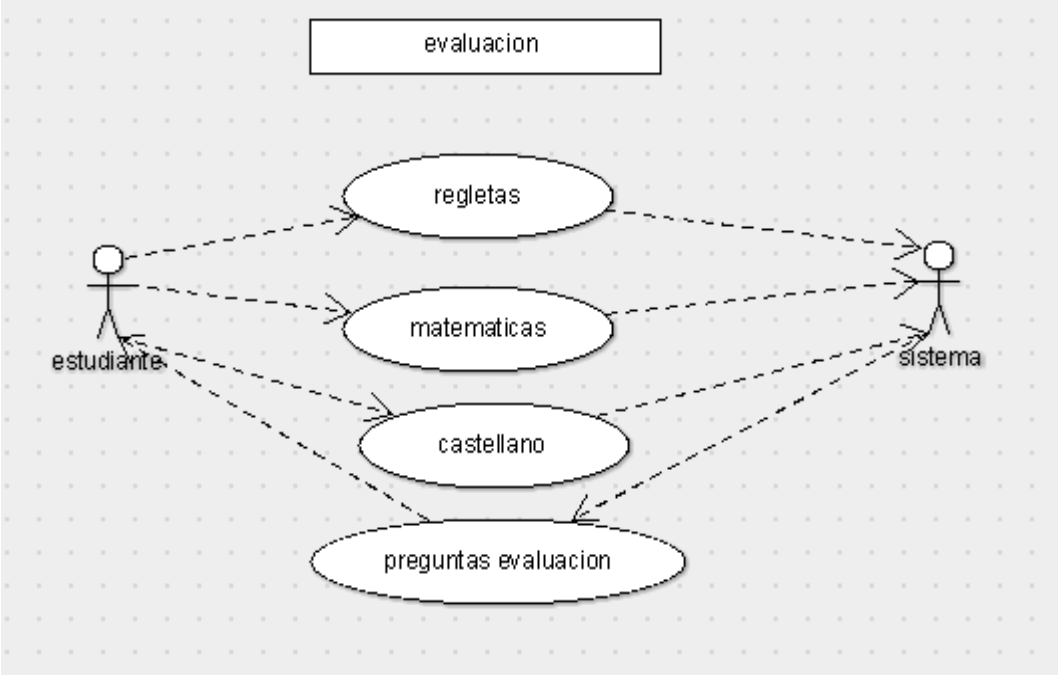
Caso de uso del docente



Casos de uso del estudiante

Se puede observar la interacción del **estudiante** con el sistema





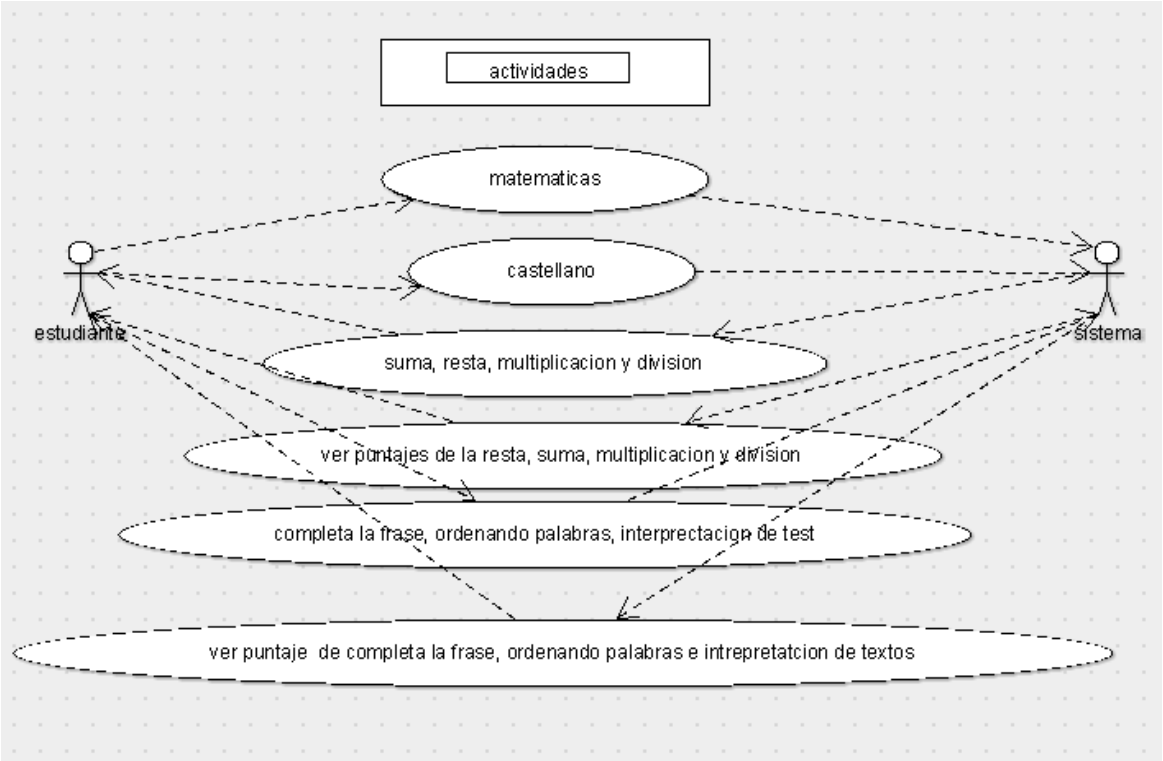

actividades
matematicas
castellano
estudiante
sistema
suma, resta, multiplicacion y division
ver puntajes de la resta, suma, multiplicacion y division
completa la frase, ordenando palabras, interprectacion de test
ver puntaje de completa la frase, ordenando palabras e intrepretatcion de textos

Diagrama de secuencia
Ilustración 8 diagrama de secuencia

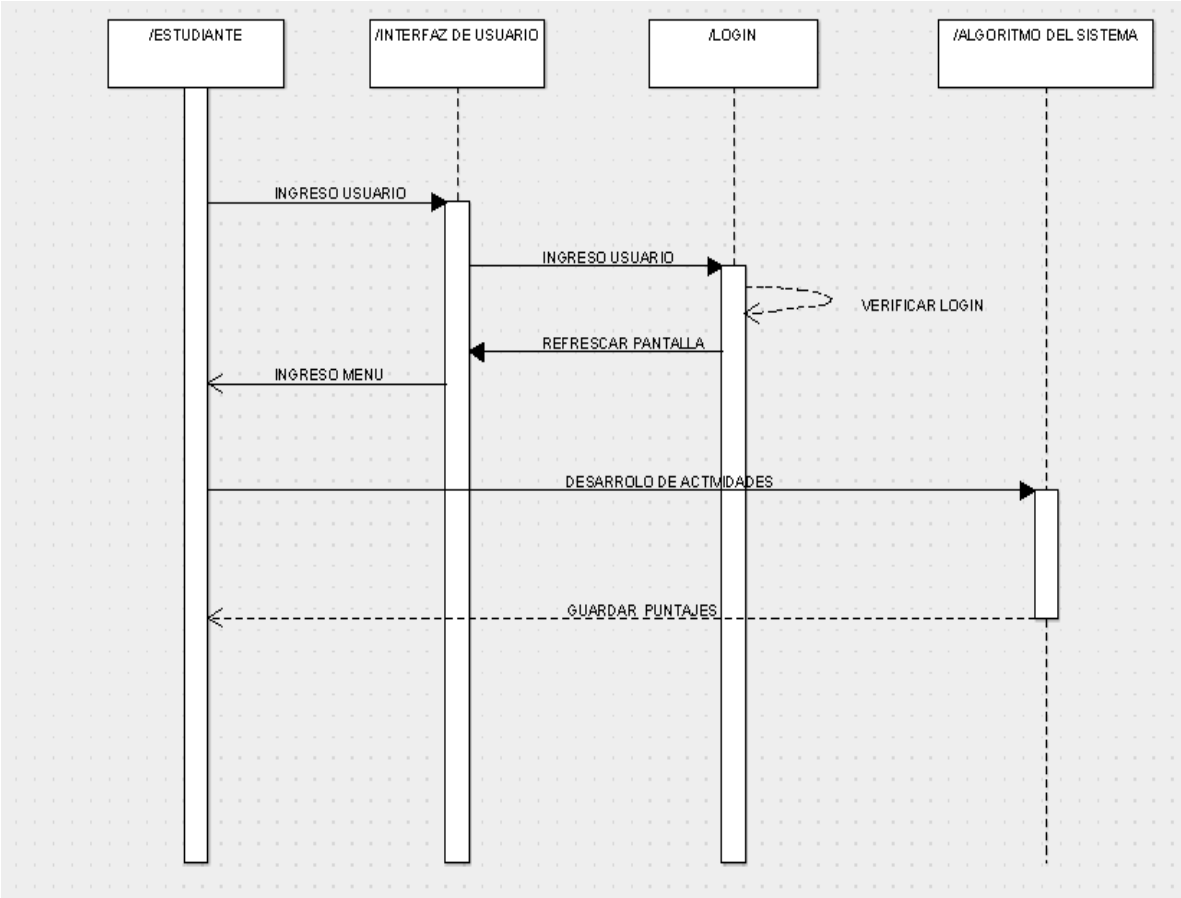
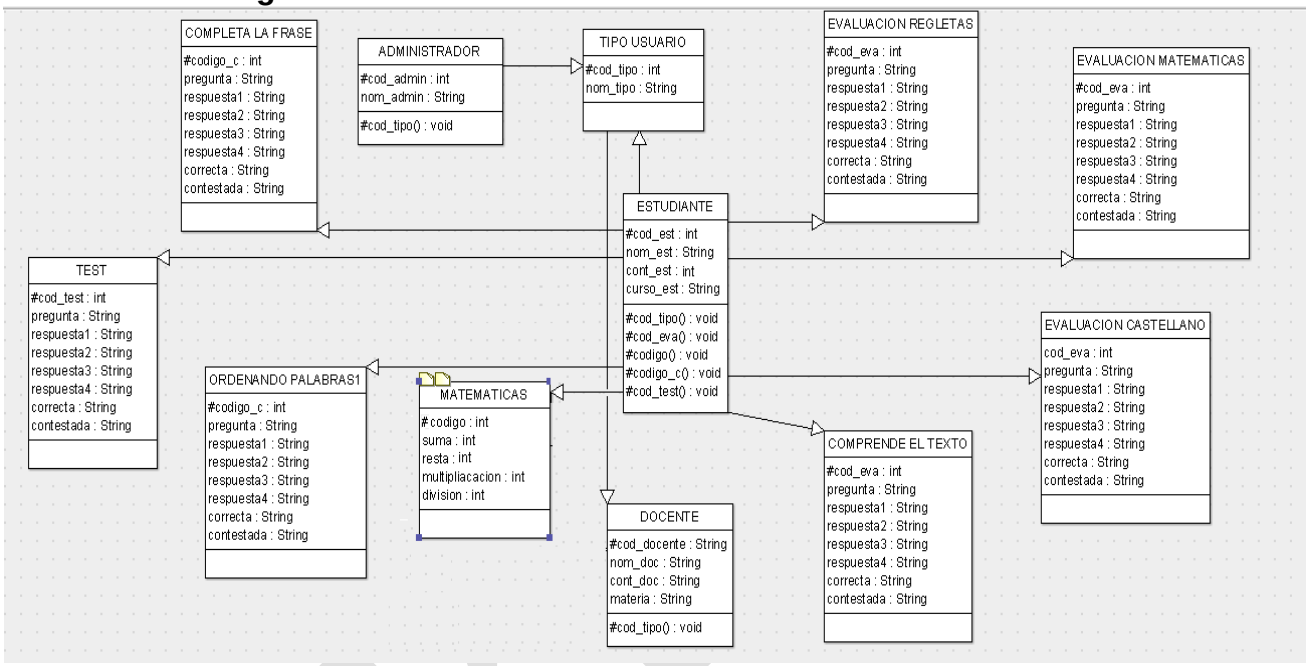


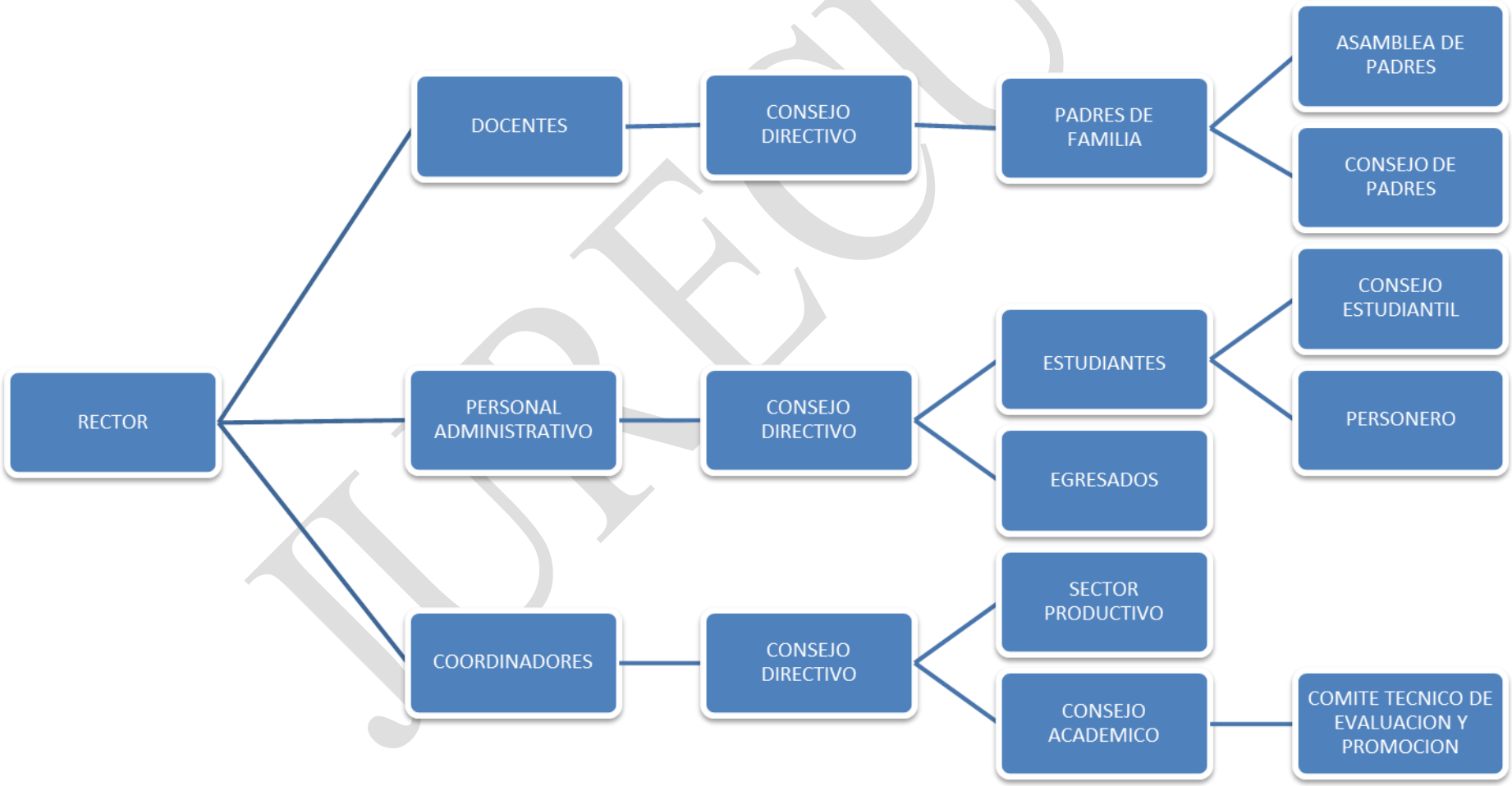
Diagrama de clases

Ilustración 9 diagrama de clases



ORGANIGRAMA DE LA EMPRESA

Ilustración 10 organigrama de la empresa



6 FASE DE EJECUCION

En la fase de ejecución se plantearon las distintas actividades definidas en el plan del proyecto.
Las actividades que durante esta etapa se realizan son:

- **Ejecución del Plan del Proyecto:** para la ejecución del proyecto se tuvieron en cuenta los costos fijos y variables de la misma manera se tuvieron en cuenta los requerimientos de hardware y software para dicha ejecución del producto.

- **Distribución de la información:** mantener la información respectiva y organizada del proyecto para que cualquier ente externo o interno este en la capacidad de manipularla para el beneficio del software.

- **Aseguramiento de la Calidad:** las normas que se establecieron en la etapa de planeación son lo primordial para que el software tenga un desarrollo de alta calidad en su implementación

- **Verificación del Alcance:** los alcances establecidos en la etapa de planeación se cumplieron en el desarrollo del producto por parte de los participantes del proyecto. Es necesario realizar una revisión técnica al producto para descartar los posibles colapsos del sistema.

REQUERIMIENTOS UTILIZADOS PARA EL DESARROLLO DEL SOFTWARE

HARDWARE
Procesador dual core de 1.60 GHz
Disco de 320 Gb
RAM 3Gb
Tarjeta gráfica Intel 256 MHz

SOFTWARE
Sistema operativo Windows 7 ultímate
Visual studio 2008. Net
SQL server 2008
SQL manager 2008
Gantt proyect

7 FASE DE CIERRE

Metodología xp

Como última etapa de la metodología, se tiene el CIERRE del proyecto, el cual contempla la terminación de las actividades relacionadas con la evaluación de los resultados obtenidos.

La etapa de cierre contendrá los siguientes módulos:

- **Evaluación de Resultados:** se realizaran distintas pruebas tanto de hardware como de software para determinar el desempeño y la calidad del sistema dando a conocer un informe final sobre los resultados obtenidos de dichas pruebas

8 FACTIBILIDAD

8.1 TÉCNICA

La Factibilidad Técnica consiste en realizar una evaluación de la tecnología existente en la organización, este estudio estuvo destinado a recolectar información y la posibilidad de hacer uso de los mismos en el desarrollo e implementación del sistema propuesto y de ser necesario, los requerimientos tecnológicos que deben ser adquiridos para el desarrollo y puesta en marcha del sistema en cuestión
De acuerdo a la tecnología necesaria, se debe evaluar bajo dos enfoques hardware y software

HARDWARE

En cuanto a hardware, específicamente las maquinas donde debe estar instalado el sistema propuesto este debe cubrir con los siguientes requerimientos mininos

Sistema Operativo Windows 7
* Procesador: Dual Core de 2.1 GHz
* RAM: 2 GB
* Disco: 250GB
* Tarjeta grafica: 512MB

SOFTWARE

En cuanto al software, la institución cuenta con todas las aplicaciones que se emplearán para el desarrollo del proyecto y funcionamiento del sistema lo cual no amerita inversión alguna para la adquisición de los mismos.

8.2 FACTIBILIDAD ECONÓMICA

COSTOS

A continuación se muestra los costos del diseño del software

FIJOS		VARIABLES	
TRANSPORTE:	$350.000	LUZ:	$30.000
ALIMENTACION:	$480.000	GAS:	$2.500
INTERNET:	$35.000	AGUA:	$10.000
IMPRESIONES:	$30.000		
	TOTAL:		$937.500

Tabla 2. Costos Factibilidad Económica.

VALOR DEL PROYECTO

El costo del proyecto es: $12. 937.500
El sueldo mensual de un programador es de $ 1.000.000
Más los costos para el diseño de este proyecto son: $937.500

8.3 FACTIBILIDAD HUMANA

Se debe capacitar a los docentes en el manejo de los módulos del software (JURECU) para que pueda determinar sin ningún inconveniente los resultados o datos guardados por los estudiantes de los respectivos cursos (1 y 2 grado) dándole un buen uso y manejo al software didáctico-educativo

9 CONCLUSIONES

Las conclusiones para este proyecto son:

- Fue exitoso el diseño de un formulario que se llamada evaluación, el cual permite al docente observar el desarrollo del estudiante con el manejo del software educativo.

- Se diseñó un formulario con las operaciones básicas, en el que por medio de un aplicativo permite el uso de las regletas cuisenaire con operaciones matemáticas.

- Se diseñó un módulo que permite a los estudiantes interactuar con el software por medio de distintos juegos descargados de la web.

- Se diseñó un formulario llamado castellano, compuesto por los subtemas formación de palabras, comprensión de textos y completa la frase para colaborar con la lectura y escritura.

- Se descargó un juego de la web que permite por medio de un test evaluar la retentiva visual y agilidad mental que el estudiante puede llegar a tener.

- Se está realizando las pruebas respectivas de conexión con la base de datos ya elaborada, con el fin de guardar los puntajes obtenidos por los estudiantes en los diferentes módulos del software.

10 BIBLIOGRAFIA E INFOGRAFIA

- DML EDU-CAR, S.A. DE C.V. DML EDU-CAR, S.A. DE C.V. definición las regletas cuisenaire, una breve descripción de que son las regleras cuisenaire, 2010.
  Disponible en: http://www.busquedazonal.com/mx/classified/REGLETAS-CUISENAIRE-56.html

- Wikipedia. Regletas cuisenaire. descripción sobre las regletas cuisenaire, presentando los tamaños, colores y numeración básica sobre las regletas, 2012.
  Disponible en: http://es.wikipedia.org/wiki/Regletas_de_Cuisenaire

- Wikipedia. definición de clase en informática, tipos de clases, componentes, métodos y propiedades-2012.
  Disponible en: http://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)

- Juan Pavón Mestras,Facultad de Informática UCM. Fundamentos de la Programación Orientada a Objetos, 2007.
  Disponible en: https://docs.google.com/viewer?a=v&q=cache:85JHWL9C7ToJ:www.fdi.ucm.es/profesor/jpavon/poo/1.1.Objetos%2520y%2520Clases.pdf+objeto+informatica&hl=es&gl=co&pid=bl&srcid=ADGEESgzupwWQ4BT9xEfr7-M256t45aaInqhqpP_0fT-Q50xc6JA4Yz8Y8Q31A6O_-OUJZOYhviYB3kYHzKK4KMfBv5a4Bm9AsTSuhOypuPrSBoz_ZF7feOcQOW-rFiiH48CyuBPipmu&sig=AHIEtbQGUGPzP2294QjjFr8aNR98JOiwfg

- Wikipedia. objeto. definición de objeto 2012.
  Disponible en:
  http://es.wikipedia.org/wiki/Objeto_(programaci%C3%B3n)

- ALEGSA - Santa Fe, Argentina. términos de programación, 2009.
  Disponible en: http://www.alegsa.com.ar/Dic/modulo.php

- Anónimo, WorDPress el lenguaje, 2008-2012
  Disponible en: http://definicion.de/lenguaje/

- Anónimo, las matematicas.2012
  Disponible en: http://www.misrespuestas.com/que-son-las-matematicas.html

- Anomino, metodología Xp, 2000
  http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP

- Autores:
  Araujo, Yuriana C.I 16.330.755Lòpez Hilda C.I. 17.648.545Mendoza, Alexander C.I 16.555.928Torrealba, Luis C.I 17.815.311Ortiz, Germán C.I.14.216.559Prof: Lic. Guerra Roberto. Metodología rup, 2010
  Disponible en: http://es.scribd.com/doc/31440864/Metodologia-RUP

- Anónimo, los antecedentes, 2008
  Disponible en: http://www.slideshare.net/contactofaum/antecedentes-524448

- ISC. Gabriel g. ruiz galvan. Diagramas, 2012
  Disponible en: http://www.slideshare.net/GRmatik7/informatica-ii-3-diagramas

- Anónimo , introducción visual studio, 2012
  Disponible en: http://msdn.microsoft.com/es-es/library/fx6bk1f4(v=vs.80).aspx

- ALEGSA - Santa Fe, Argentina, definición UML,1998 -2012
  Disponible en: http://www.alegsa.com.ar/Dic/uml.php

- Jacaboson, I., Booch, G., Rumbaugh J., El Proceso Unificado de Desarrollo de Software, Addison Wesley, conceptos de rup. 2000
  Disponible en: http://es.scribd.com/doc/7844685/CONCEPTOS-DE-RUP

- Janus45, imagen bienvenida del software. 2012
  Disponible en: http://imagenes-tiernas.net/imagenes-tiernas-de-bienvenida.html

- Anónimo, juegos usados en el software, 2012.
  Disponible en: http://juegosflash.dibujos.net/educativos/mas-jugados/

- Boyce-Codd, modelo relacional, 2008
  Disponible en: http://www.uazuay.edu.ec/analisis/El%20modelo%20relacional.pdf

- Damian perez valdes, bases de datos. 2007
  Disponible en: http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/

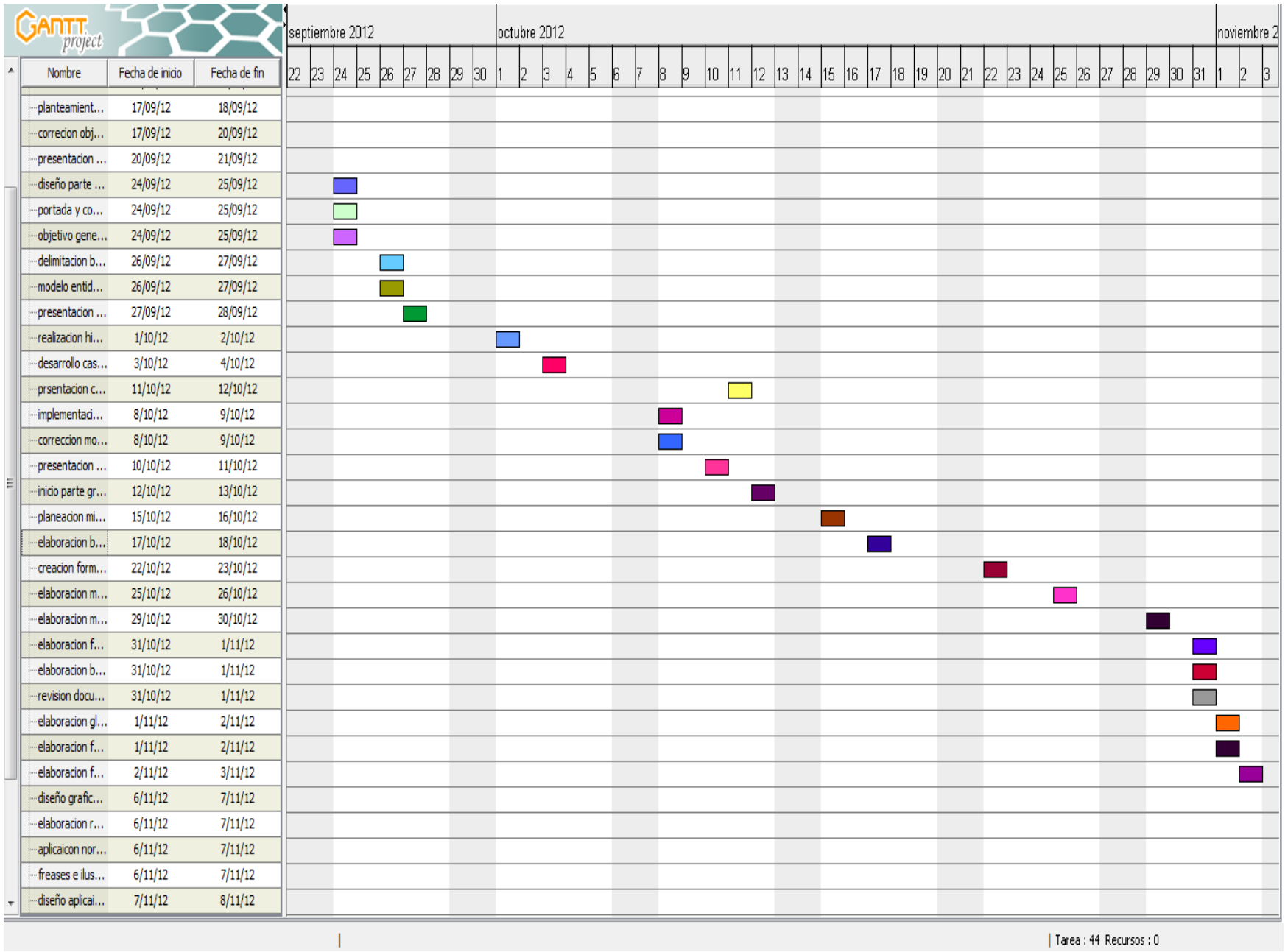
- Raúl González Duque, ciclo de vida.
  Disponible en: http://mundogeek.net/archivos/2004/05/20/ciclos-de-vida-del-software/
- Anónimo, definición de dato, 2012
  Disponible en: http://www.mastermagazine.info/termino/4530.php

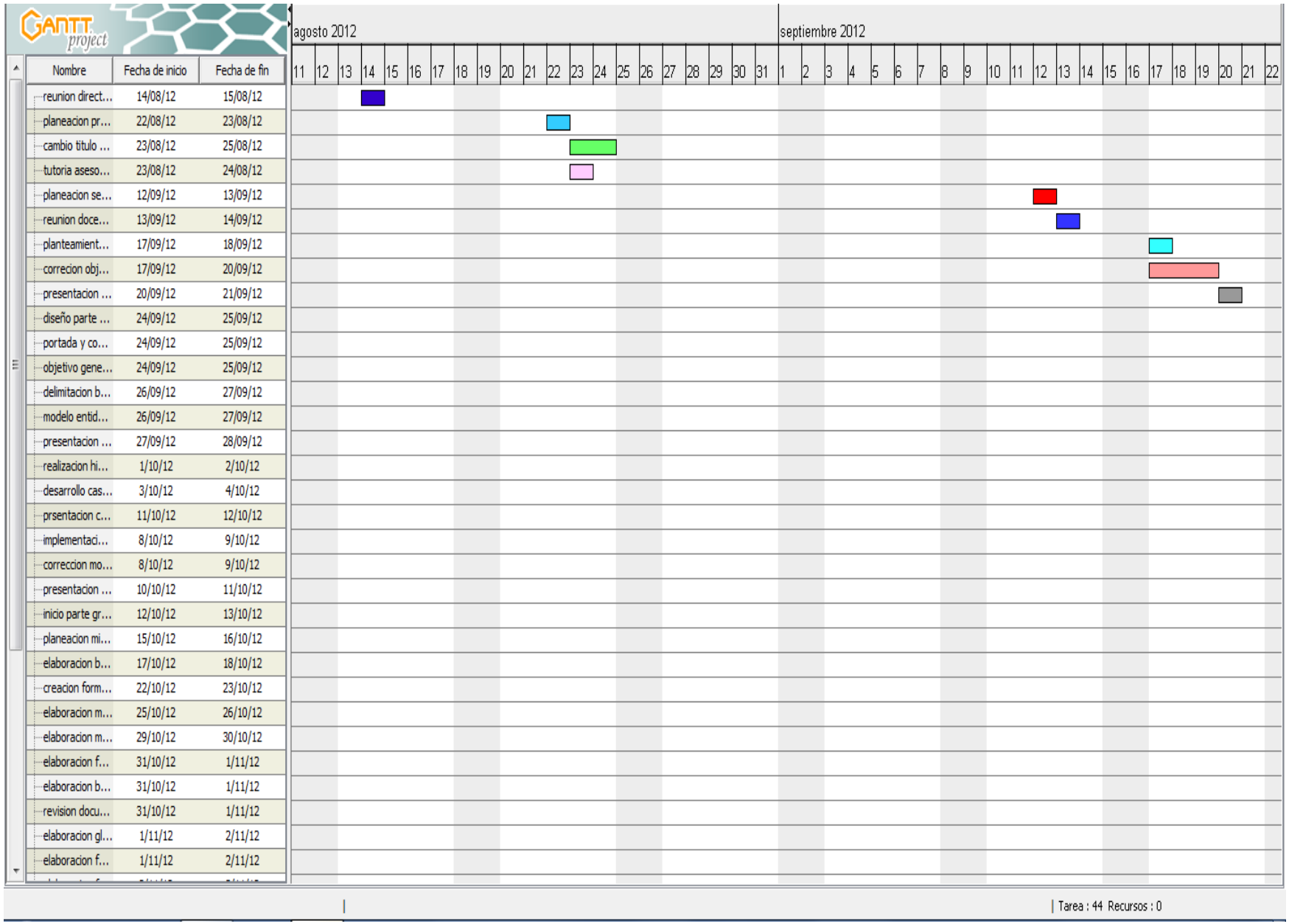
- Anónimo, imagen modelo en cascada, 2012
  Disponible en: http://www.google.com.co/imgres?imgurl=http://static.commentcamarche.net/es.kioskea.net/pictures/genie-logiciel-images-cycle-vie-cascade.png&imgrefurl=http://es.kioskea.net/contents/genie-logiciel/cycle-de-vie.php3&h=493&w=599&sz=20&tbnid=Apgyr6vroOw6MM:&tbnh=90&tbnw=109&prev=/search%3Fq%3Dmodelo%2Ben%2Bcascada%26tbm%3Disch%26tbo%3Du&zoom=1&q=modelo+en+cascada&usg=__QOtC_I5pxXl047yEED8f5ndM6L0=&docid=18D0f3bmxaWR_M&sa=X&ei=llOdUOJ-k7z1BMLJgcAC&ved=0CCoQ9QEwAw&dur=1241

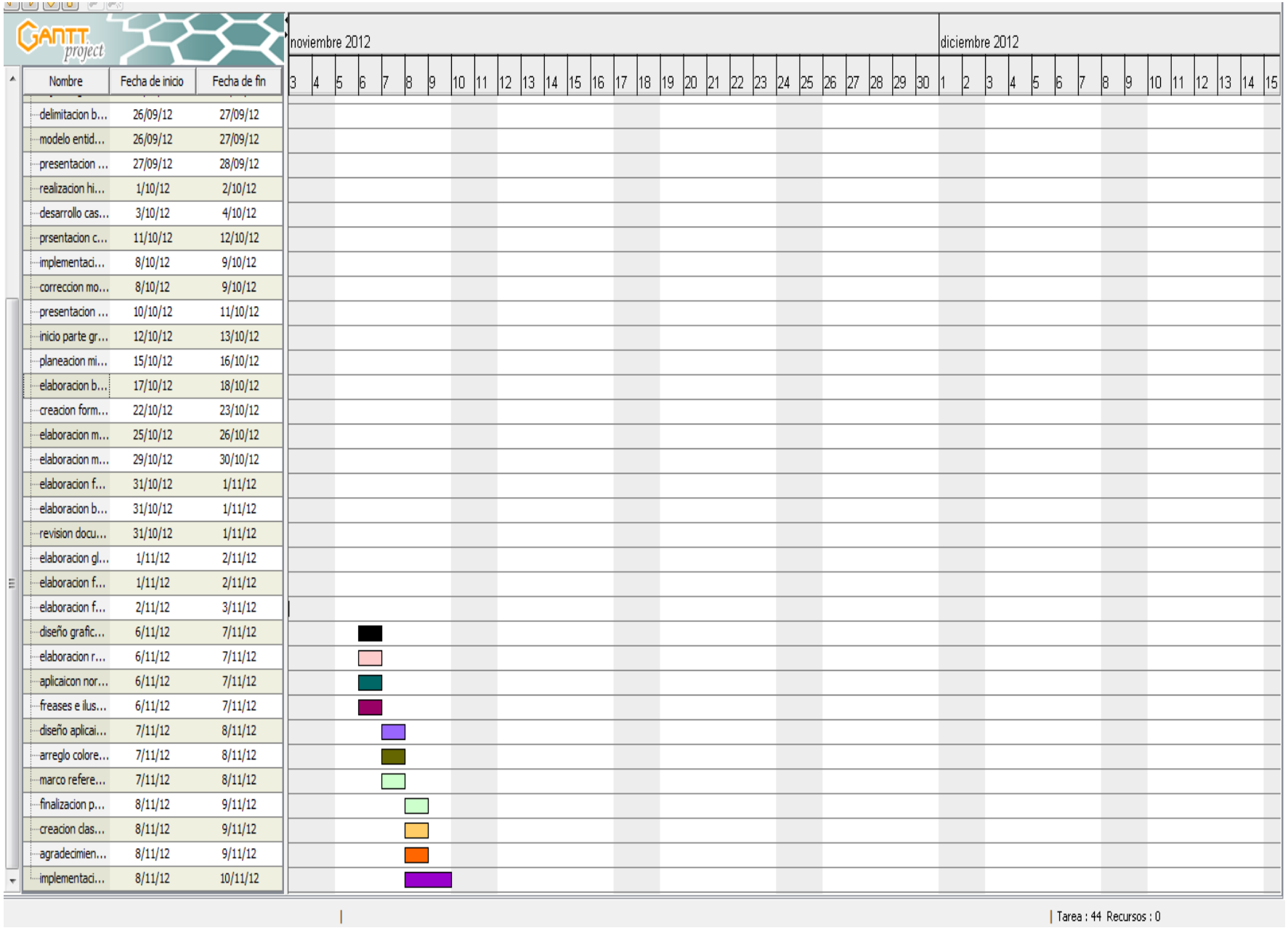
- Anónimo, imagen implementada en el software. 2012.
  Disponible en: http://www.google.com.co/imgres?q=manzanas&um=1&hl=es-419&biw=1440&bih=732&tbm=isch&tbnid=eIZj0UPYJoJ_DM:&imgrefurl=http://www.greenandfresh.co/productos/&docid=d7UOG-Nh8lJpnM&imgurl=http://www.greenandfresh.co/wp-content/uploads/2010/08/banner-manzanas.jpg&w=1020&h=400&ei=M3aZULT8I4Lm8gTO2YGwBQ&zoom=1&iact=hc&vpx=187&vpy=141&dur=760&hovh=140&hovw=359&tx=207&ty=87&sig=101625273194659731805&page=3&tbnh=127&tbnw=292&start=59&ndsp=33&ved=1t:429,r:39,s:20,i:309

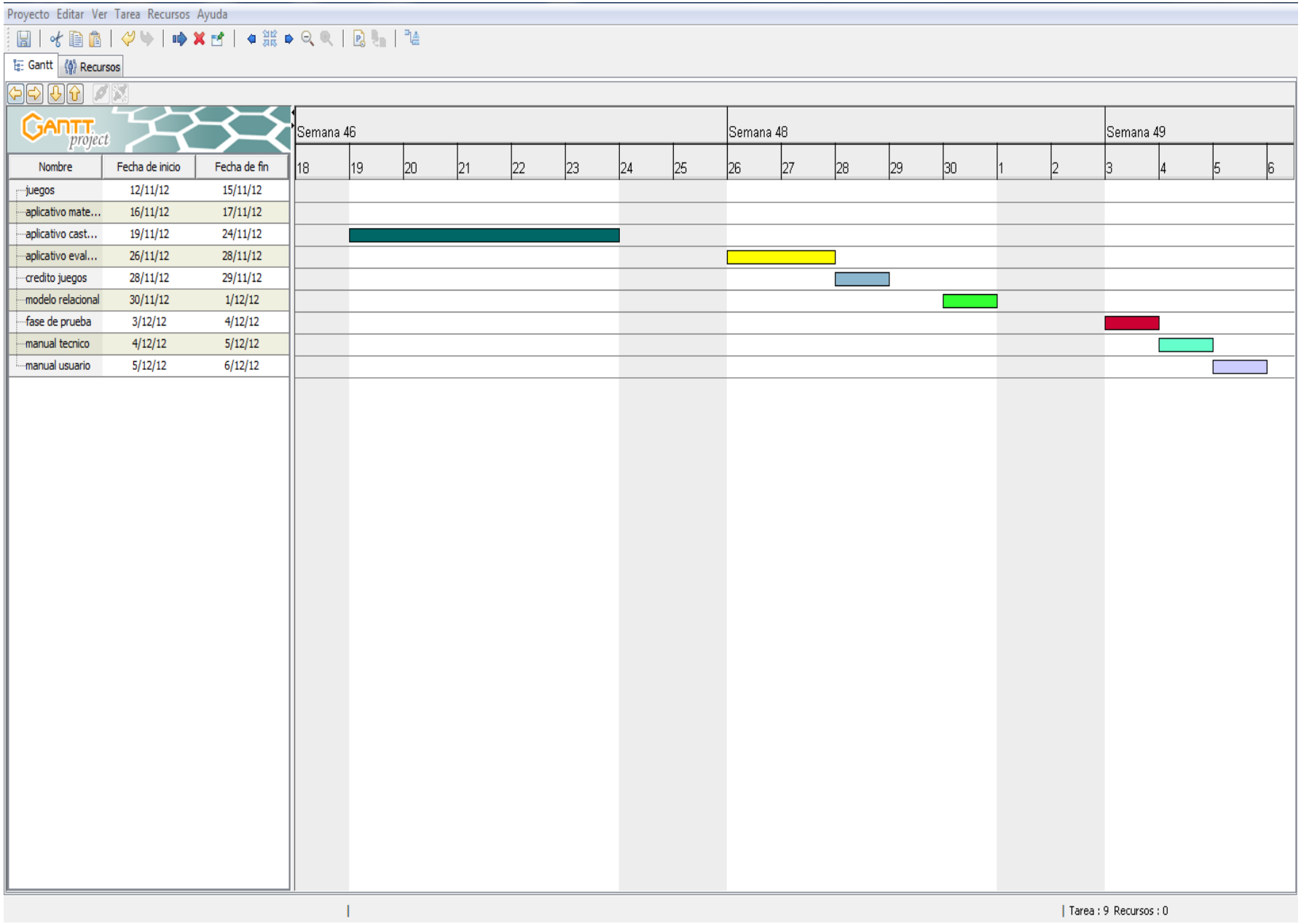
11 ANEXOS

ANEXO I. CRONOGRAMA DE ACTIVIDADES.

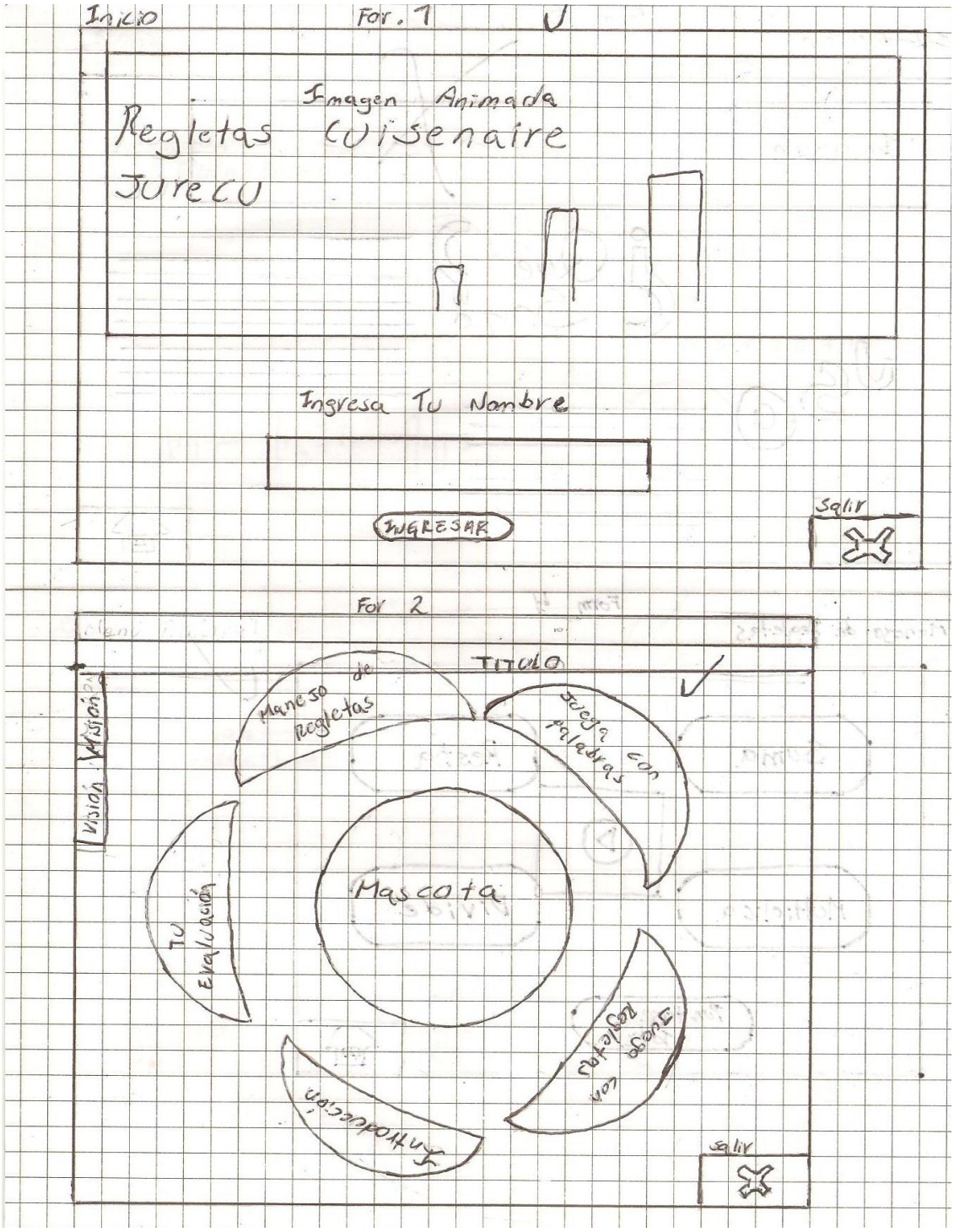

GANTT project
septiembre 2012
octubre 2012
noviembre 2
22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3
Nombre | Fecha de inicio | Fecha de fin
planteamient... | 17/09/12 | 18/09/12
correcion obj... | 17/09/12 | 20/09/12
presentacion ... | 20/09/12 | 21/09/12
diseño parte ... | 24/09/12 | 25/09/12
portada y co... | 24/09/12 | 25/09/12
objetivo gene... | 24/09/12 | 25/09/12
delimitacion b... | 26/09/12 | 27/09/12
modelo entid... | 26/09/12 | 27/09/12
presentacion ... | 27/09/12 | 28/09/12
realizacion hi... | 1/10/12 | 2/10/12
desarrollo cas... | 3/10/12 | 4/10/12
prsentacion c... | 11/10/12 | 12/10/12
implementaci... | 8/10/12 | 9/10/12
correccion mo... | 8/10/12 | 9/10/12
presentacion ... | 10/10/12 | 11/10/12
inicio parte gr... | 12/10/12 | 13/10/12
planeacion mi... | 15/10/12 | 16/10/12
elaboracion b... | 17/10/12 | 18/10/12
creacion form... | 22/10/12 | 23/10/12
elaboracion m... | 25/10/12 | 26/10/12
elaboracion m... | 29/10/12 | 30/10/12
elaboracion f... | 31/10/12 | 1/11/12
elaboracion b... | 31/10/12 | 1/11/12
revision docu... | 31/10/12 | 1/11/12
elaboracion gl... | 1/11/12 | 2/11/12
elaboracion f... | 1/11/12 | 2/11/12
elaboracion f... | 2/11/12 | 3/11/12
diseño grafic... | 6/11/12 | 7/11/12
elaboracion r... | 6/11/12 | 7/11/12
aplicaicon nor... | 6/11/12 | 7/11/12
freases e ilus... | 6/11/12 | 7/11/12
diseño aplicai... | 7/11/12 | 8/11/12
Tarea : 44 Recursos : 0

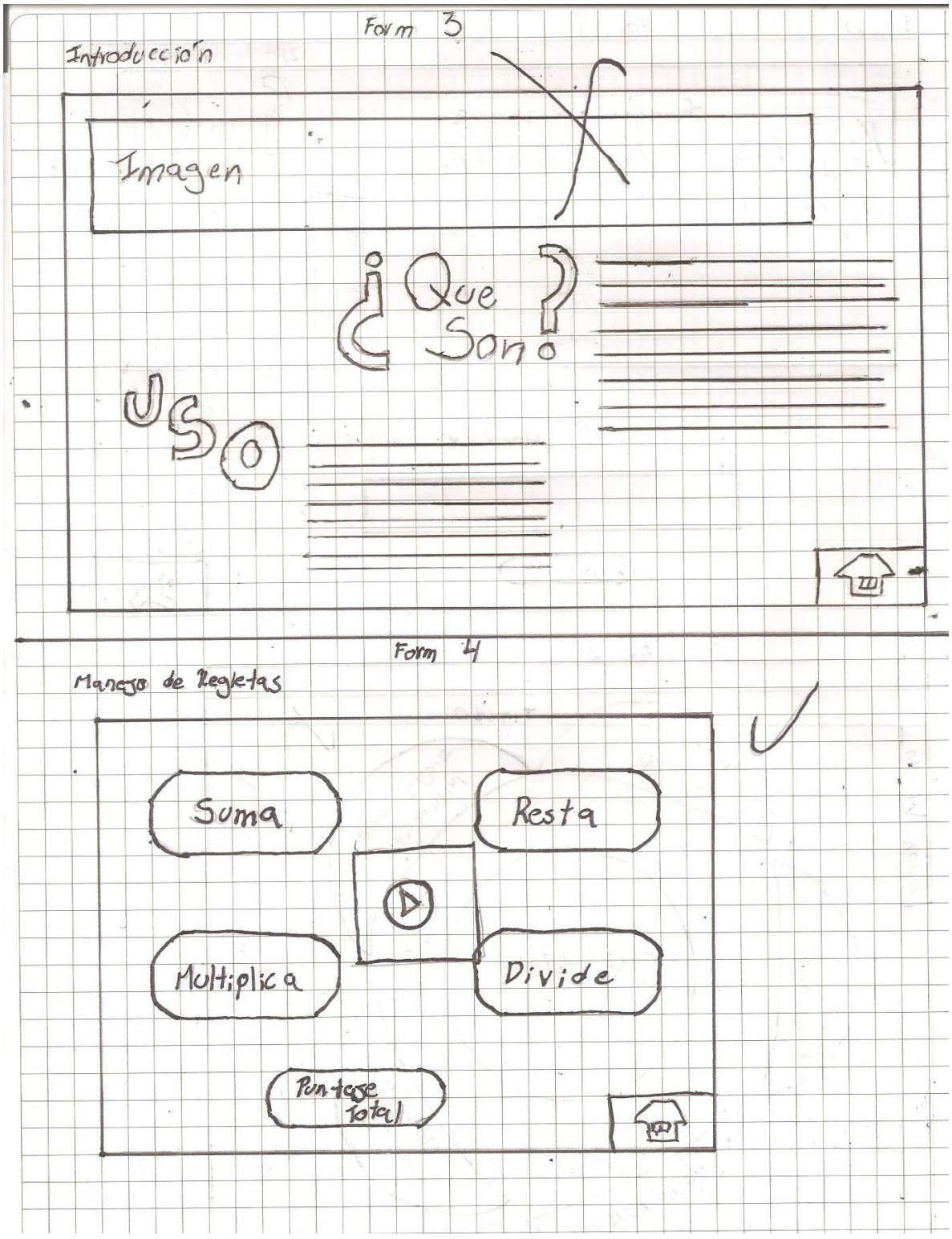

GANTT project
agosto 2012
septiembre 2012
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Nombre | Fecha de inicio | Fecha de fin
reunion direct... | 14/08/12 | 15/08/12
planeacion pr... | 22/08/12 | 23/08/12
cambio titulo ... | 23/08/12 | 25/08/12
tutoria aseso... | 23/08/12 | 24/08/12
planeacion se... | 12/09/12 | 13/09/12
reunion doce... | 13/09/12 | 14/09/12
planteamient... | 17/09/12 | 18/09/12
correcion obj... | 17/09/12 | 20/09/12
presentacion ... | 20/09/12 | 21/09/12
diseño parte ... | 24/09/12 | 25/09/12
portada y co... | 24/09/12 | 25/09/12
objetivo gene... | 24/09/12 | 25/09/12
delimitacion b... | 26/09/12 | 27/09/12
modelo entid... | 26/09/12 | 27/09/12
presentacion ... | 27/09/12 | 28/09/12
realizacion hi... | 1/10/12 | 2/10/12
desarrollo cas... | 3/10/12 | 4/10/12
prsentacion c... | 11/10/12 | 12/10/12
implementaci... | 8/10/12 | 9/10/12
correccion mo... | 8/10/12 | 9/10/12
presentacion ... | 10/10/12 | 11/10/12
inicio parte gr... | 12/10/12 | 13/10/12
planeacion mi... | 15/10/12 | 16/10/12
elaboracion b... | 17/10/12 | 18/10/12
creacion form... | 22/10/12 | 23/10/12
elaboracion m... | 25/10/12 | 26/10/12
elaboracion m... | 29/10/12 | 30/10/12
elaboracion f... | 31/10/12 | 1/11/12
elaboracion b... | 31/10/12 | 1/11/12
revision docu... | 31/10/12 | 1/11/12
elaboracion gl... | 1/11/12 | 2/11/12
elaboracion f... | 1/11/12 | 2/11/12
Tarea : 44 Recursos : 0

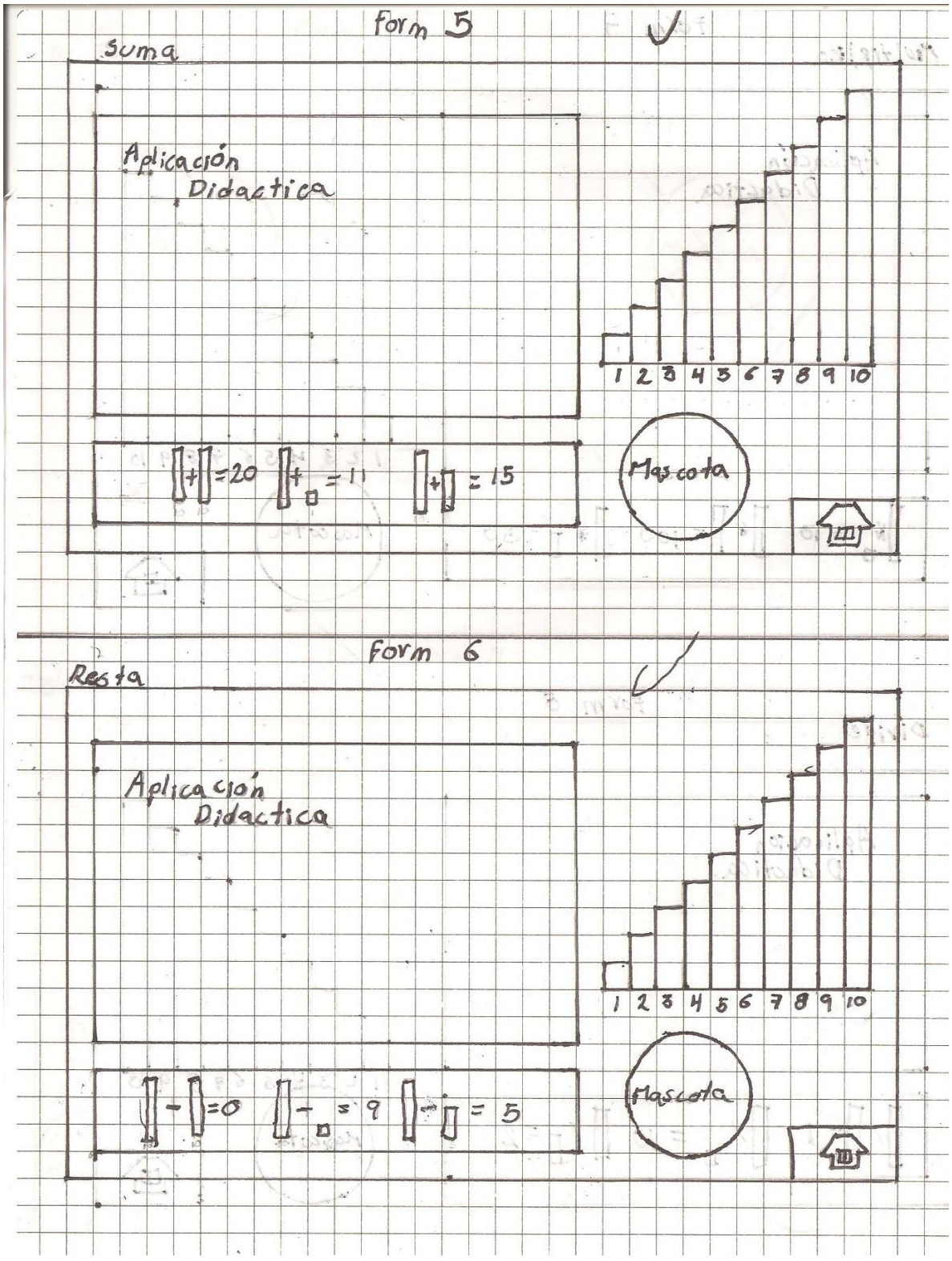

GANTT project
noviembre 2012
diciembre 2012
Nombre
Fecha de inicio
Fecha de fin
delimitacion b... 26/09/12 27/09/12
modelo entid... 26/09/12 27/09/12
presentacion ... 27/09/12 28/09/12
realizacion hi... 1/10/12 2/10/12
desarrollo cas... 3/10/12 4/10/12
prsentacion c... 11/10/12 12/10/12
implementaci... 8/10/12 9/10/12
correccion mo... 8/10/12 9/10/12
presentacion ... 10/10/12 11/10/12
inicio parte gr... 12/10/12 13/10/12
planeacion mi... 15/10/12 16/10/12
elaboracion b... 17/10/12 18/10/12
creacion form... 22/10/12 23/10/12
elaboracion m... 25/10/12 26/10/12
elaboracion m... 29/10/12 30/10/12
elaboracion f... 31/10/12 1/11/12
elaboracion b... 31/10/12 1/11/12
revision docu... 31/10/12 1/11/12
elaboracion gl... 1/11/12 2/11/12
elaboracion f... 1/11/12 2/11/12
elaboracion f... 2/11/12 3/11/12
diseño grafic... 6/11/12 7/11/12
elaboracion r... 6/11/12 7/11/12
aplicaicon nor... 6/11/12 7/11/12
freases e ilus... 6/11/12 7/11/12
diseño aplicai... 7/11/12 8/11/12
arreglo colore... 7/11/12 8/11/12
marco refere... 7/11/12 8/11/12
finalizacion p... 8/11/12 9/11/12
creacion clas... 8/11/12 9/11/12
agradecimien... 8/11/12 9/11/12
implementaci... 8/11/12 10/11/12
Tarea : 44 Recursos : 0


Proyecto Editar Ver Tarea Recursos Ayuda
Gantt
Recursos
GANTT project
Nombre
Fecha de inicio
Fecha de fin
juegos 12/11/12 15/11/12
aplicativo mate... 16/11/12 17/11/12
aplicativo cast... 19/11/12 24/11/12
aplicativo eval... 26/11/12 28/11/12
credito juegos 28/11/12 29/11/12
modelo relacional 30/11/12 1/12/12
fase de prueba 3/12/12 4/12/12
manual tecnico 4/12/12 5/12/12
manual usuario 5/12/12 6/12/12
Semana 46
Semana 48
Semana 49
18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6
Tarea : 9 Recursos : 0

ANEXO II. MODELO NO FUNCIONAL

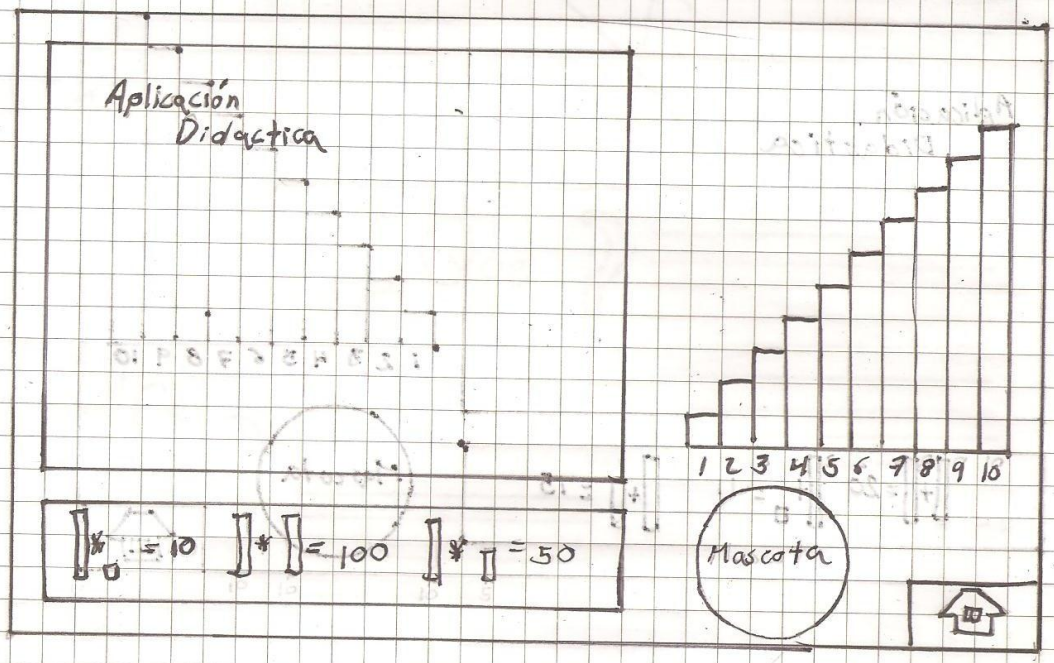



Form 3
Introducción
Imagen
¿Que Son?
Uso
Form 4
Manejo de Regletas
Suma
Resta
Multiplica
Divide
Puntaje Total


Form 5
Suma
Aplicación Didactica
1 2 3 4 5 6 7 8 9 10
Mascota
Form 6
Resta
Aplicación Didactica
1 2 3 4 5 6 7 8 9 10
Mascota

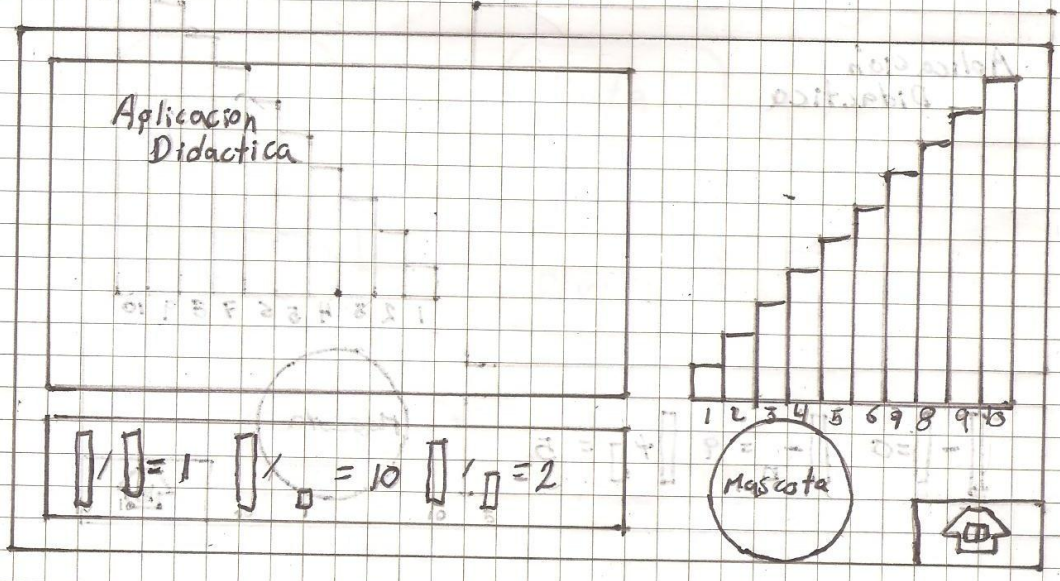
Form 7

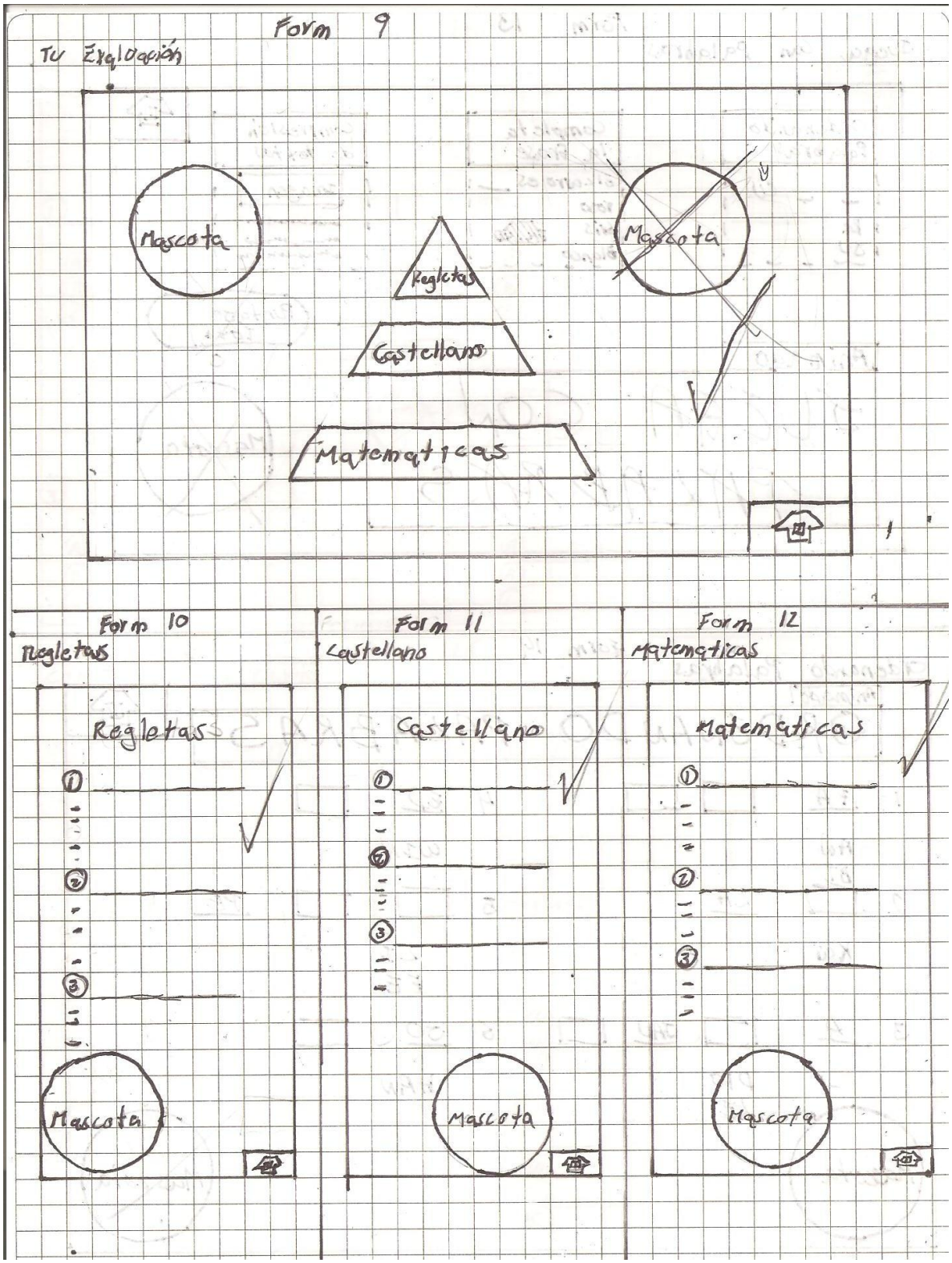
Multiplica



Form 8

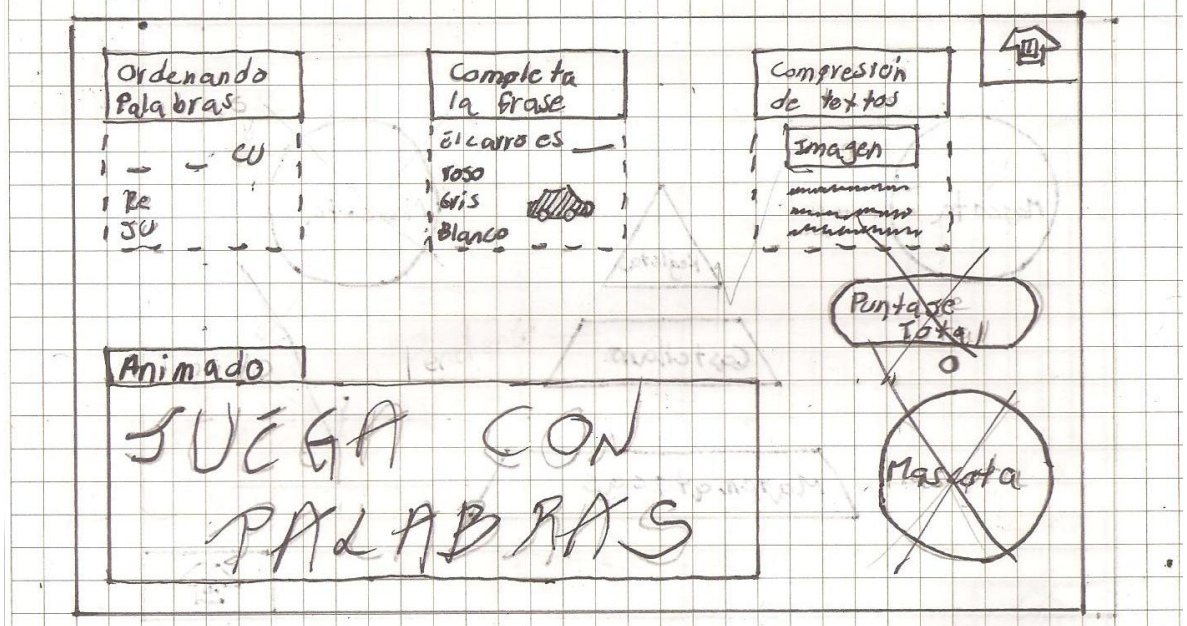
Divide




Form 9
Tu Exploración
Mascota
Regletas
Castellano
Matemáticas
Mascota
Form 10
Regletas
Regletas
Mascota
Form 11
Castellano
Castellano
Mascota
Form 12
Matemáticas
Matemáticas
Mascota

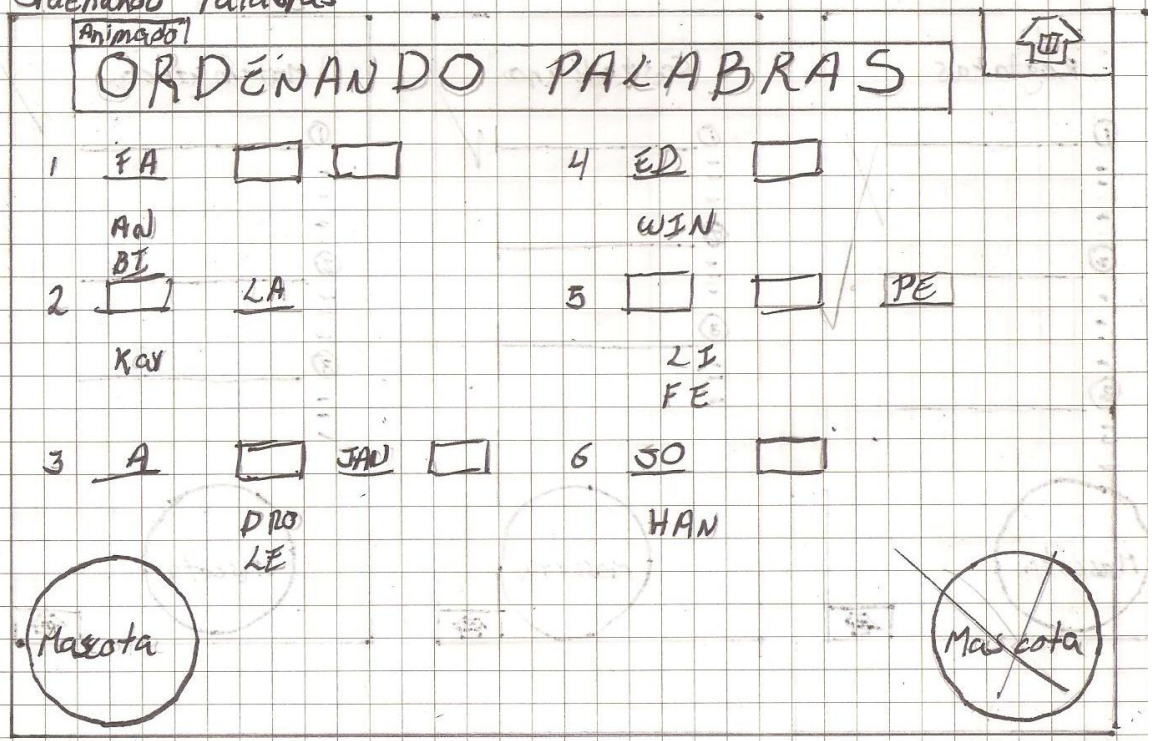
Form 13

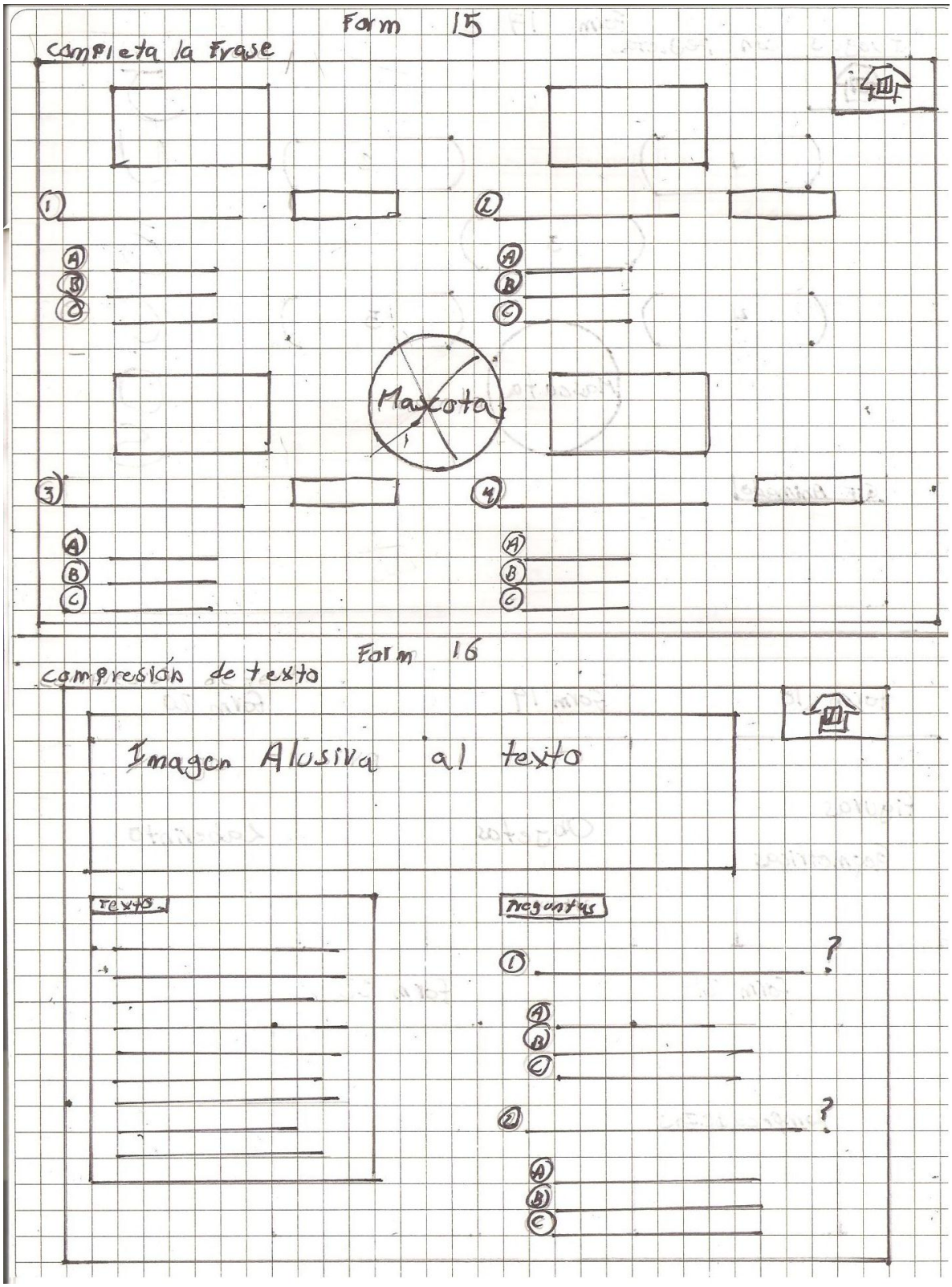
Juega con Palabras

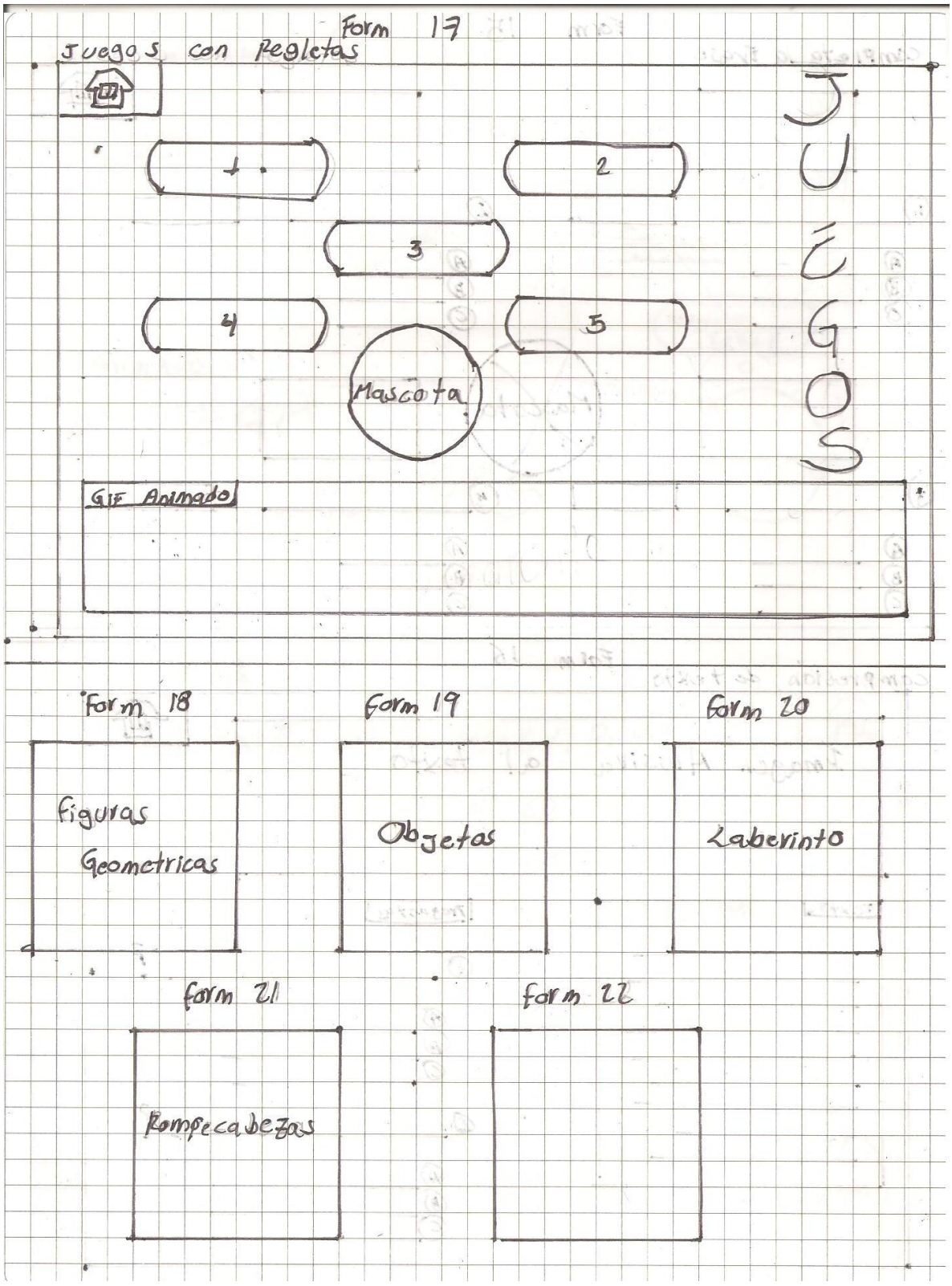


Form 14

Ordenando Palabras




Form 15
completa la frase
1
2
3
4
A
B
C
Mascota
Form 16
comprensión de texto
Imagen Alusiva al texto
texto
preguntas
1
2
?
?


Form 17
Juegos con Regletas
1
2
3
4
5
Mascota
JUEGOS
GIF Animado
Form 18
Figuras Geometricas
Form 19
Objetos
Form 20
Laberinto
form 21
Rompecabezas
form 22

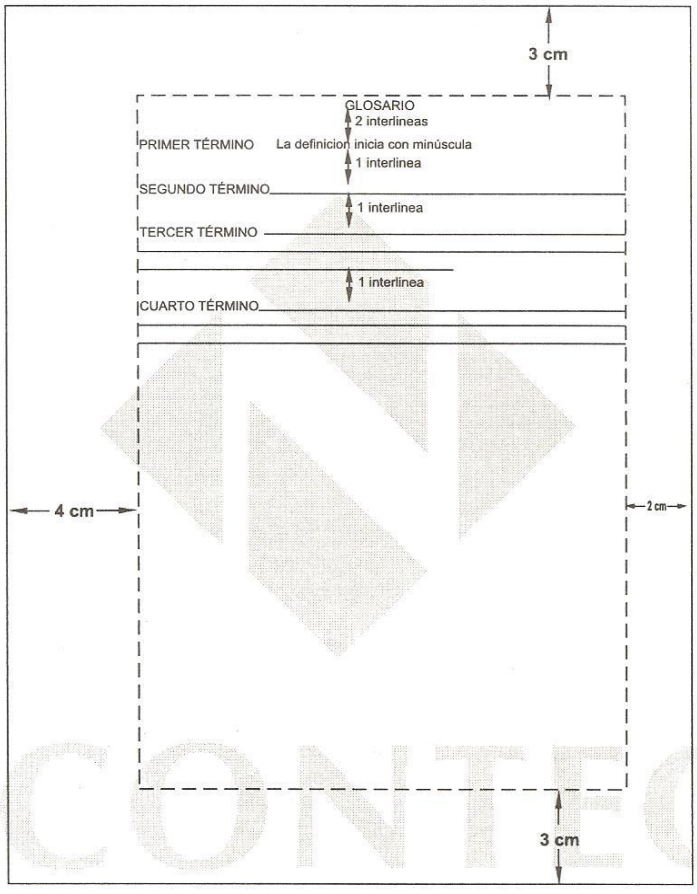
ANEXO IIII. PAGINAS UTILIZADAS ICONTEC

NORMA TÉCNICA COLOMBIANA NTC 1486 (Sexta actualización)

5.2.1.10 Glosario. Lista alfabética de términos y sus definiciones o explicaciones necesarios para la comprensión del documento. La existencia de un glosario no justifica la omisión de una explicación en el texto la primera vez que aparece un término. El título glosario se escribe en mayúscula sostenida, centrado, a 3 cm del borde superior de la hoja.

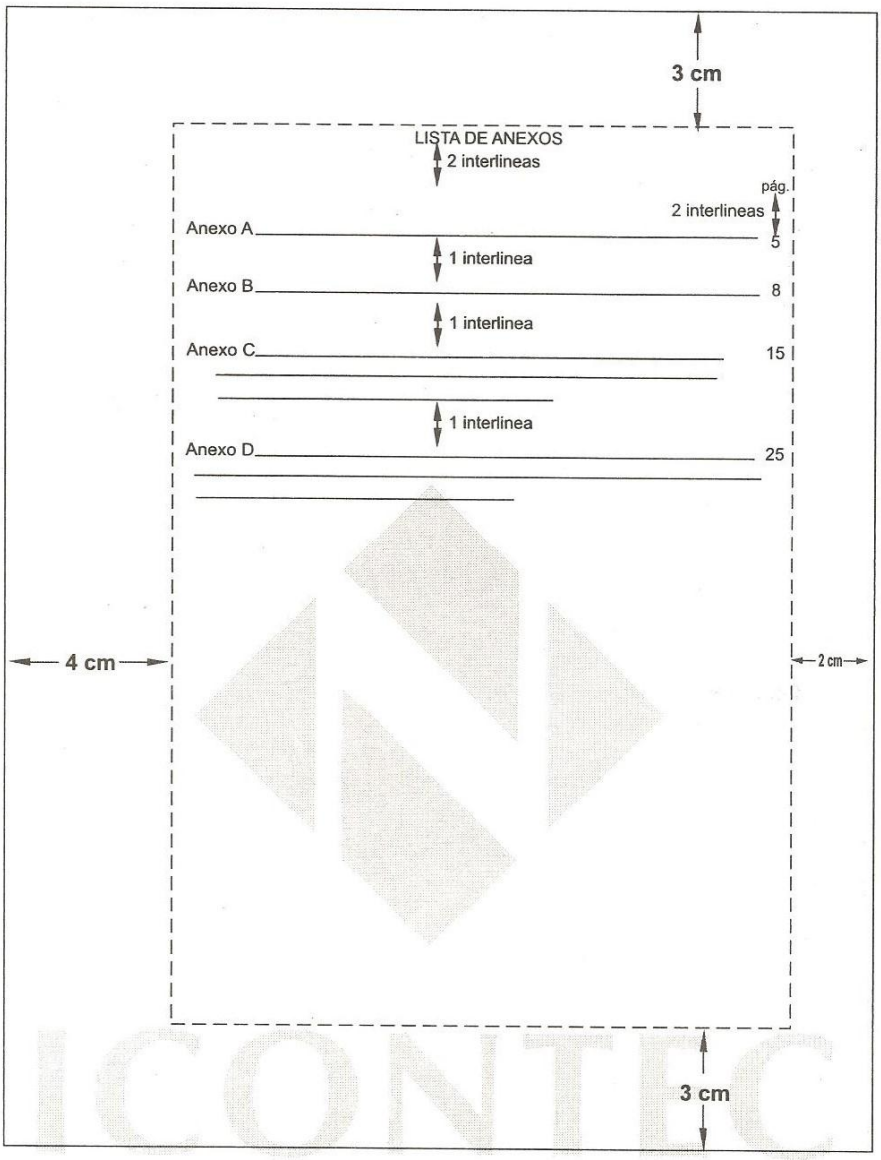
El primer término aparece a dos interlíneas del título glosario, contra el margen izquierdo. Los términos se escriben con mayúscula sostenida seguidos de dos puntos y en orden alfabético. La definición correspondiente se coloca después de los dos puntos, se deja un espacio y se inicia con minúscula. Si ocupa más de un renglón, el segundo y los subsiguientes comienzan contra el margen izquierdo. Entre término y término se deja una interlínea. Su uso es opcional.

EJEMPLO ESQUEMA DE LA PÁGINA DEL GLOSARIO

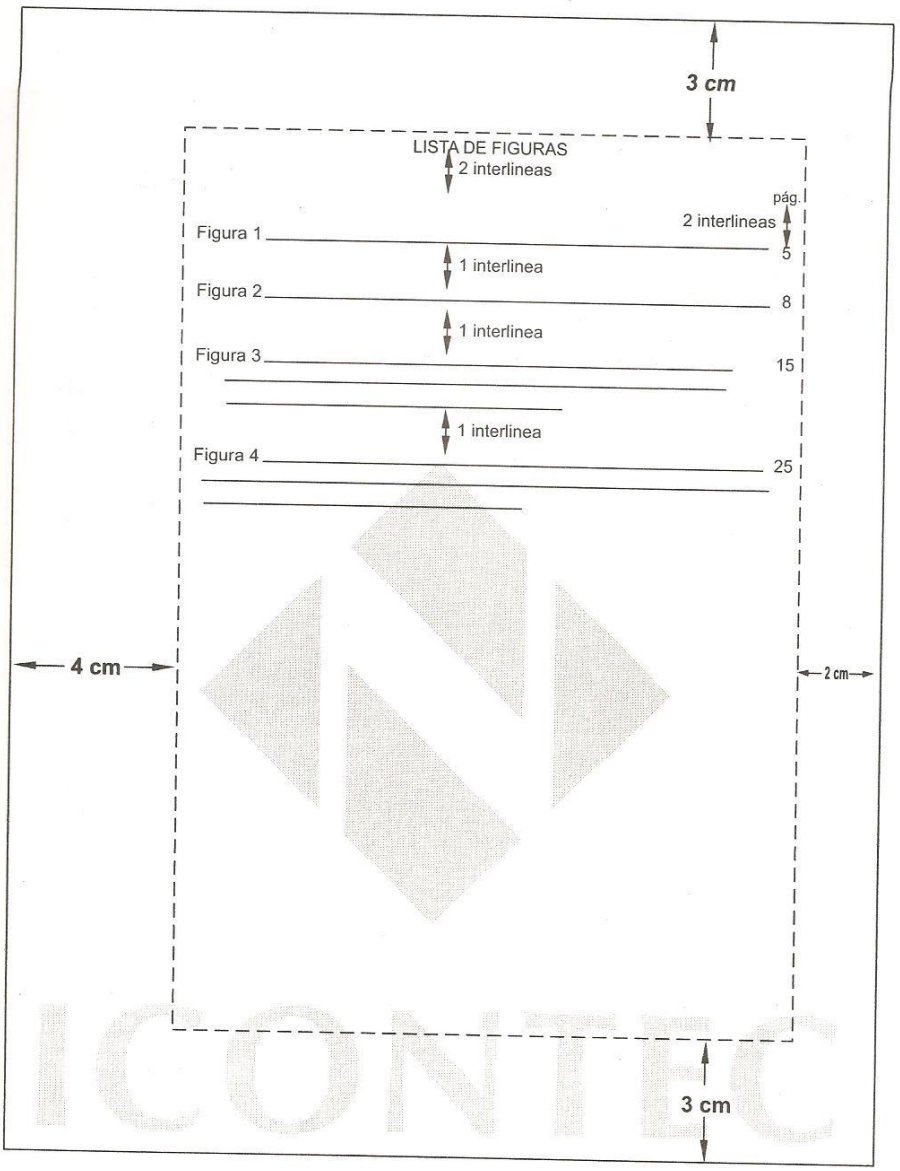


21

ESQUEMA DE LA PÁGINA DE LISTAS ESPECIALES (ANEXOS)



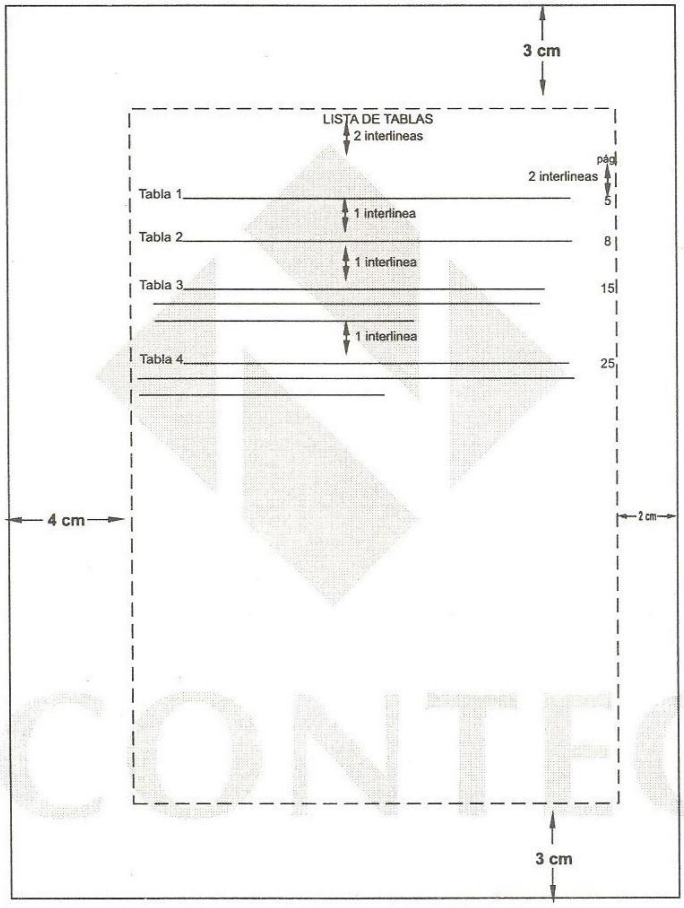
ESQUEMA DE LA PÁGINA DE LISTAS ESPECIALES (FIGURAS)



5.2.1.9. **Lisias especiales.** En las listas especiales se relacionan los títulos de las ilustraciones, tales CDJK) tat:Es, ruadros, símbolos, signos, abreviaturas, anexos y otros elementos similares que forman pañe det trabajo. El título de la lista especial se escribe centrado, en mayúscula sostenida, a 3 cm del toda de la hoja.

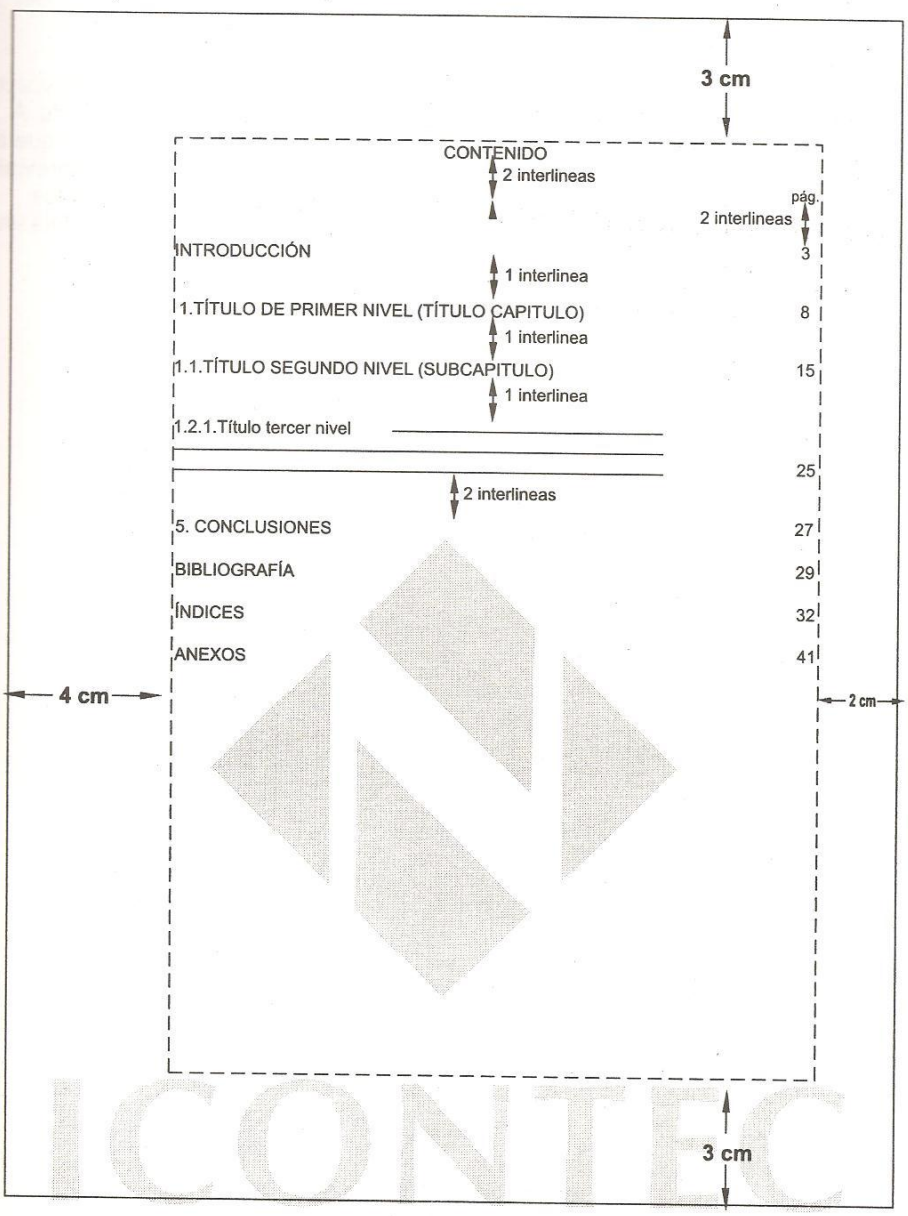
Las palabras tabla, figura, anexo y las abreviaturas, entre otras, se escriben con mayúscula inicial, seguidas del número correspondiente, o letra en los anexos, seguido de punto. A continuación, se escribe el título con mayúscula inicial, y el número de la página en que está ubicado se coloca en una columna hacia el margen derecho, encabezada con la abreviatura pág.. Si el título de la tabla, figura, u otros, ocupa más de un renglón, el segundo y los subsiguientes se escriben contra el margen izquierdo. Entre renglón y renglón se deja una interlínea.

EJEMPLOS ESQUEMA DE LISTAS ESPECIALES (TABLAS)



NORMA TÉCNICA COLOMBIANA NTC 1486 (Sexta actualización)

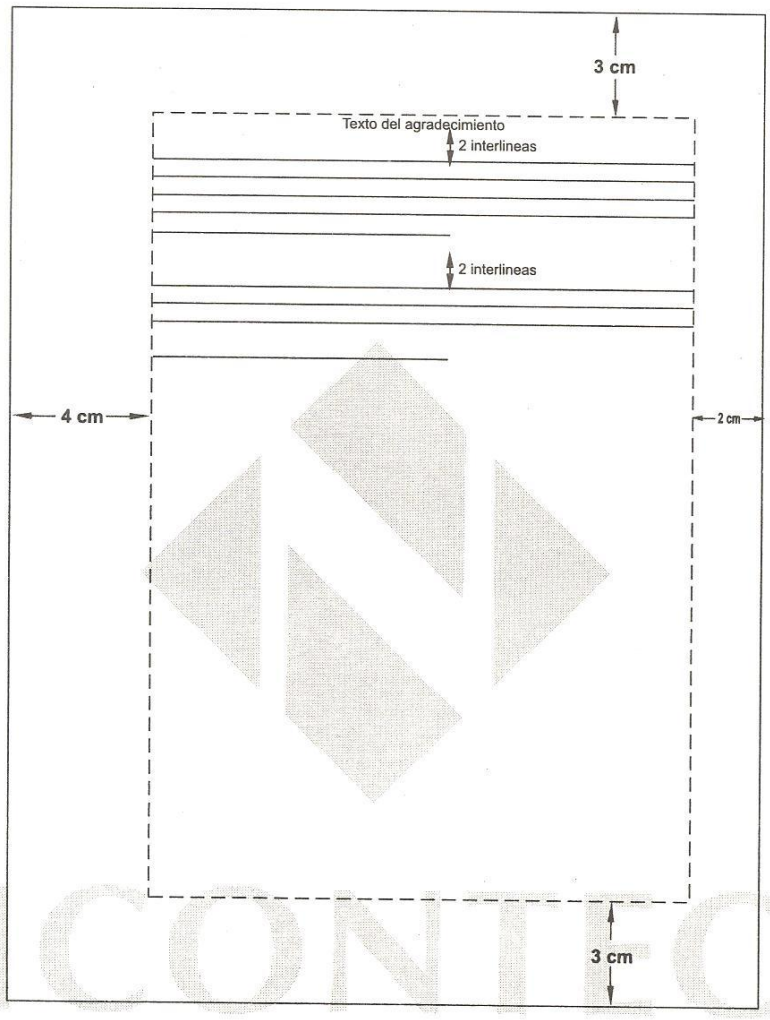
EJEMPLO ESQUEMA DE LA PÁGINA DE CONTENIDO



17

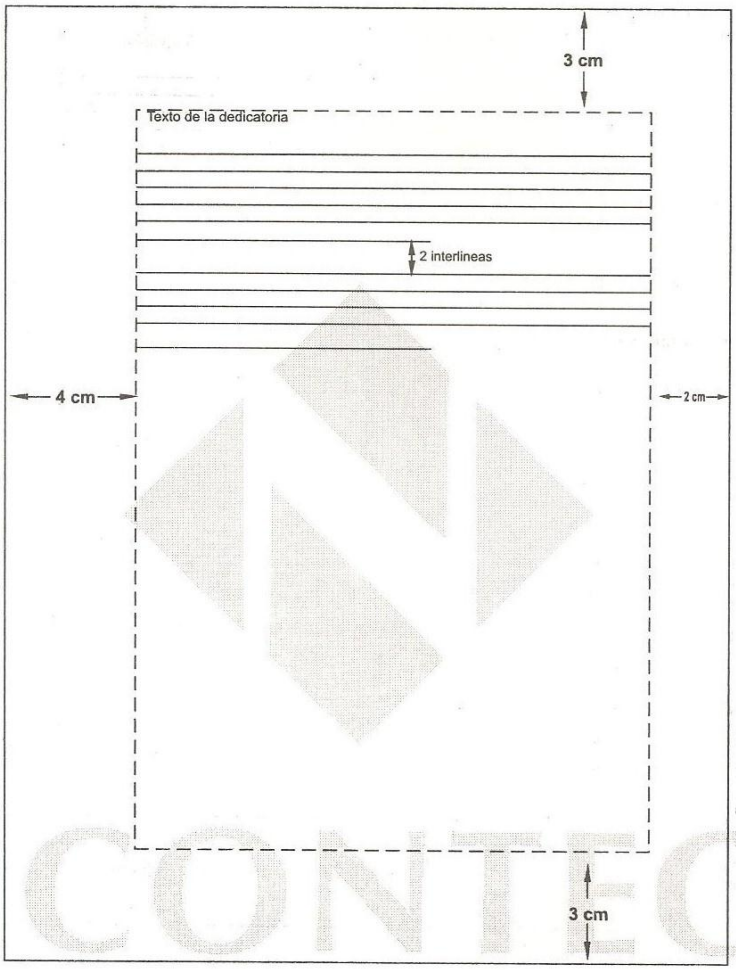
5.2.1.7 Página de agradecimientos. En ella el (los) autor (es) expresa (n) el reconocimiento hacia las personas y entidades que asesoraron técnicamente, suministraron datos, financiaron total o parcialmente la investigación o contribuyeron significativamente al desarrollo del tema. Es opcional y debe contener, además de la nota correspondiente, los nombres de las personas con sus respectivos cargos y los nombres completos de las instituciones y su aporte al trabajo.

EJEMPLO ESQUEMA DE LA PÁGINA DE AGRADECIMIENTOS



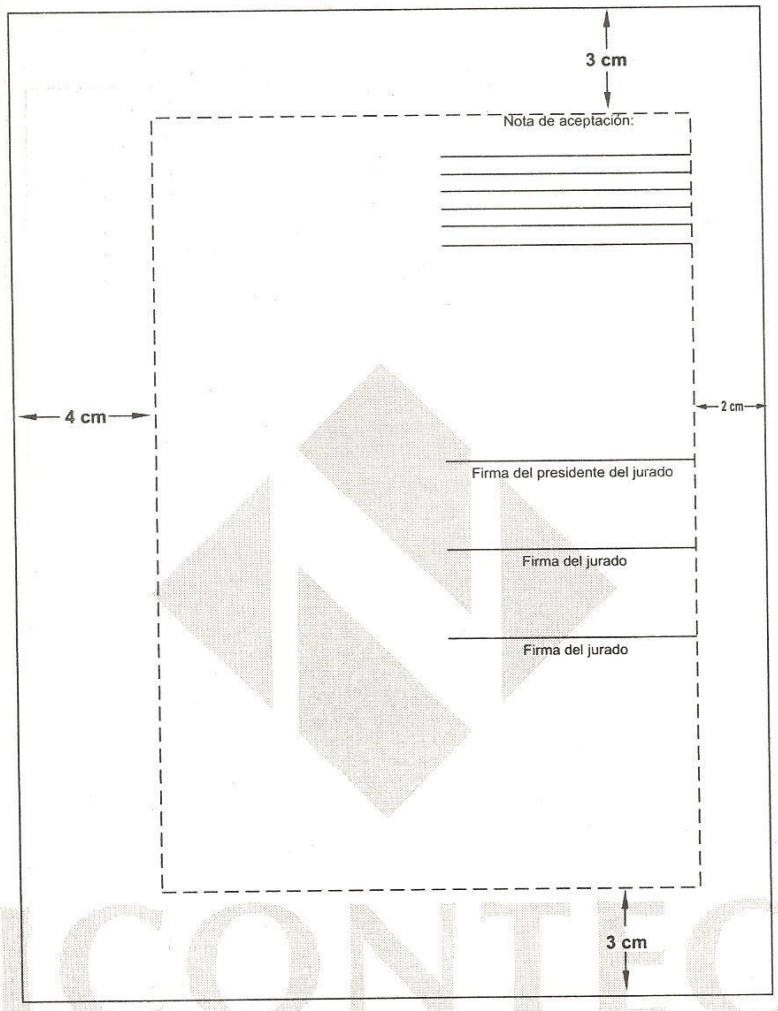
5.2.1.6 Página de dedicatoria. Nota mediante la cual el autor ofrece su trabajo, en forma especial, a personas o entidades. Su presentación es opcional y debe conservar los márgenes.

EJEMPLO ESQUEMA DE PÁGINA DE DEDICATORIA



5.2.1.5 Página de aceptación. Contiene las firmas del presidente o director y de los jurados que participan en la revisión, sustentación y aprobación del trabajo. Adicionalmente, incluye la ciudad y la fecha de entrega (día, mes, año), conservando los márgenes establecidos.

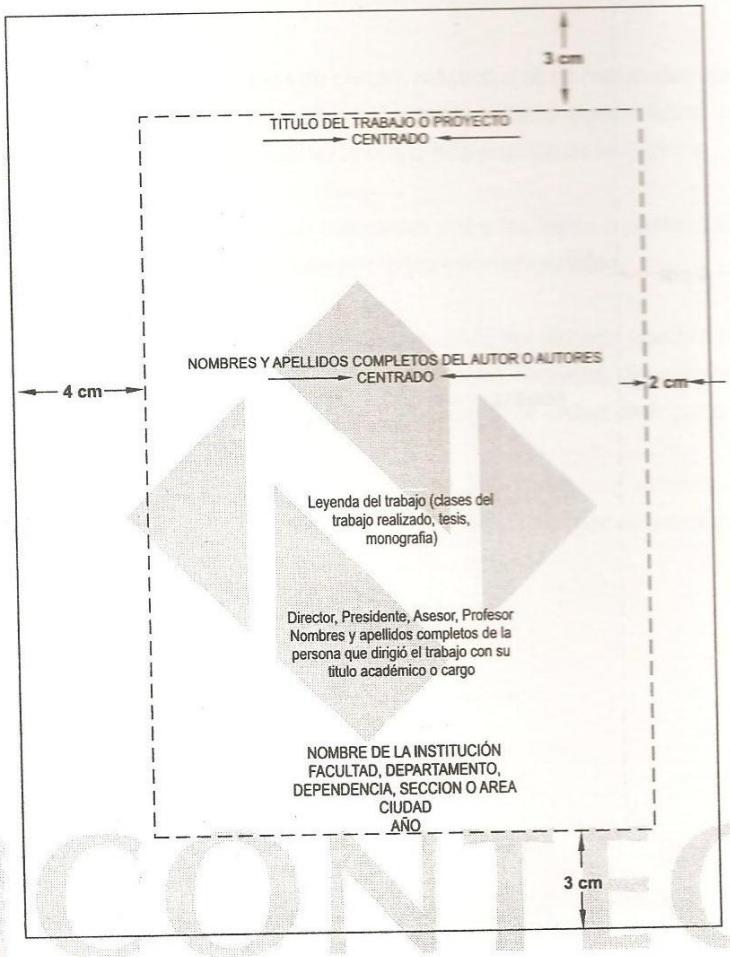
EJEMPLO ESQUEMA DE LA PÁGINA DE ACEPTACIÓN



5.2.1.4 Portada. Página informativa del documento que, además de los elementos de la cubierta; incluye la clase de trabajo realizado (tesis, monografía, trabajo, informe u otro) y el nombre con el título académico o cargo de quien lo dirigió, precedido de la palabra escrita con mayúscula inicial: director, presidente, asesor o profesor, según el caso.

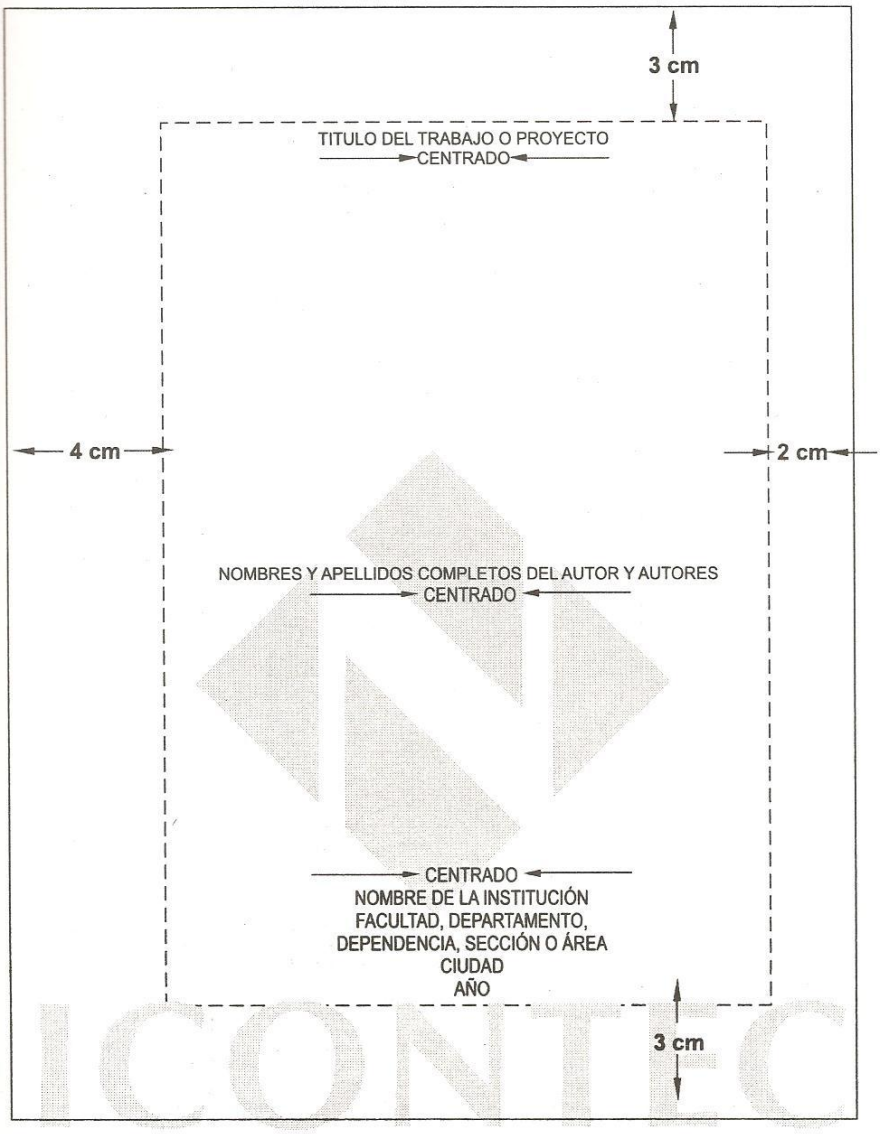
Estos dos datos se ubican equidistantes del autor y la institución, escritos en bloque.

EJEMPLO ESQUEMA DE LA PORTADA



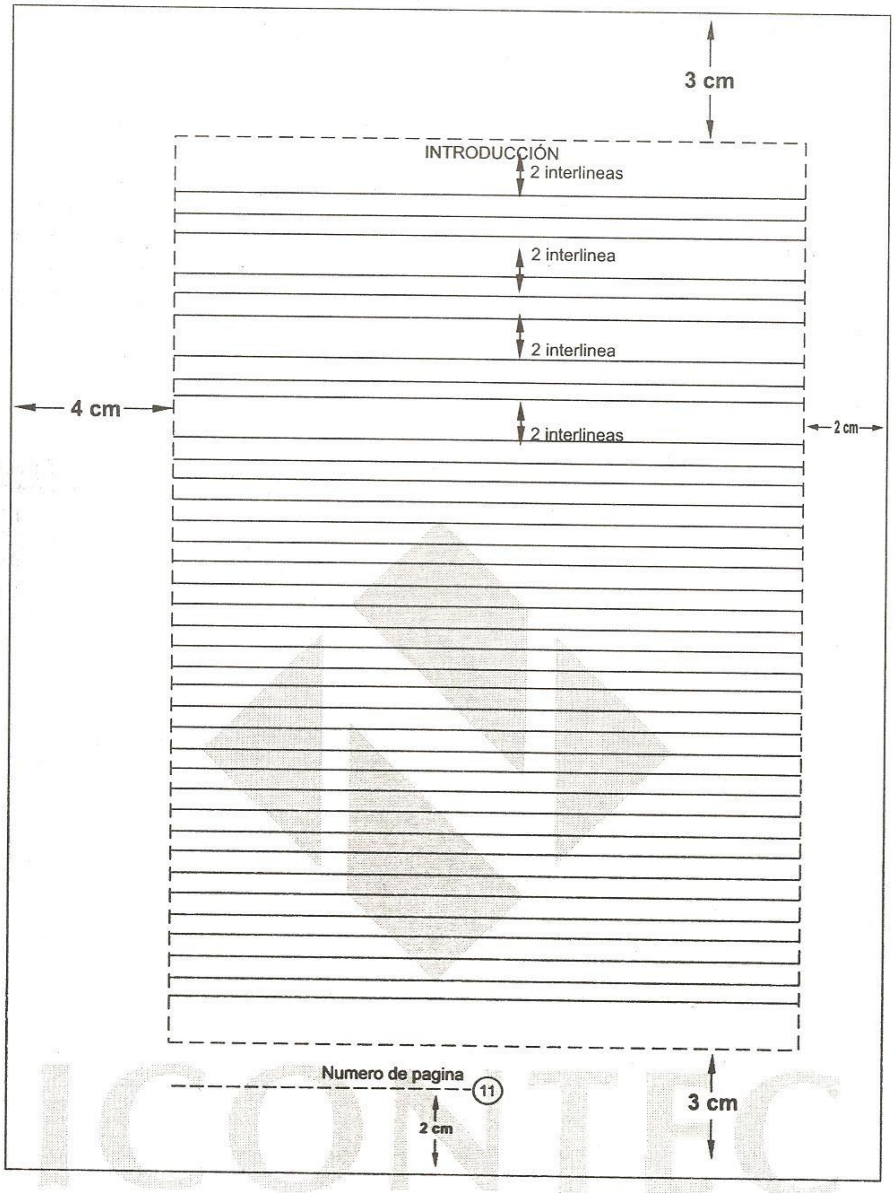
NORMA TÉCNICA COLOMBIANA NTC 1486 (Sexta actualización)

EJEMPLO ESQUEMA DE LA CUBIERTA



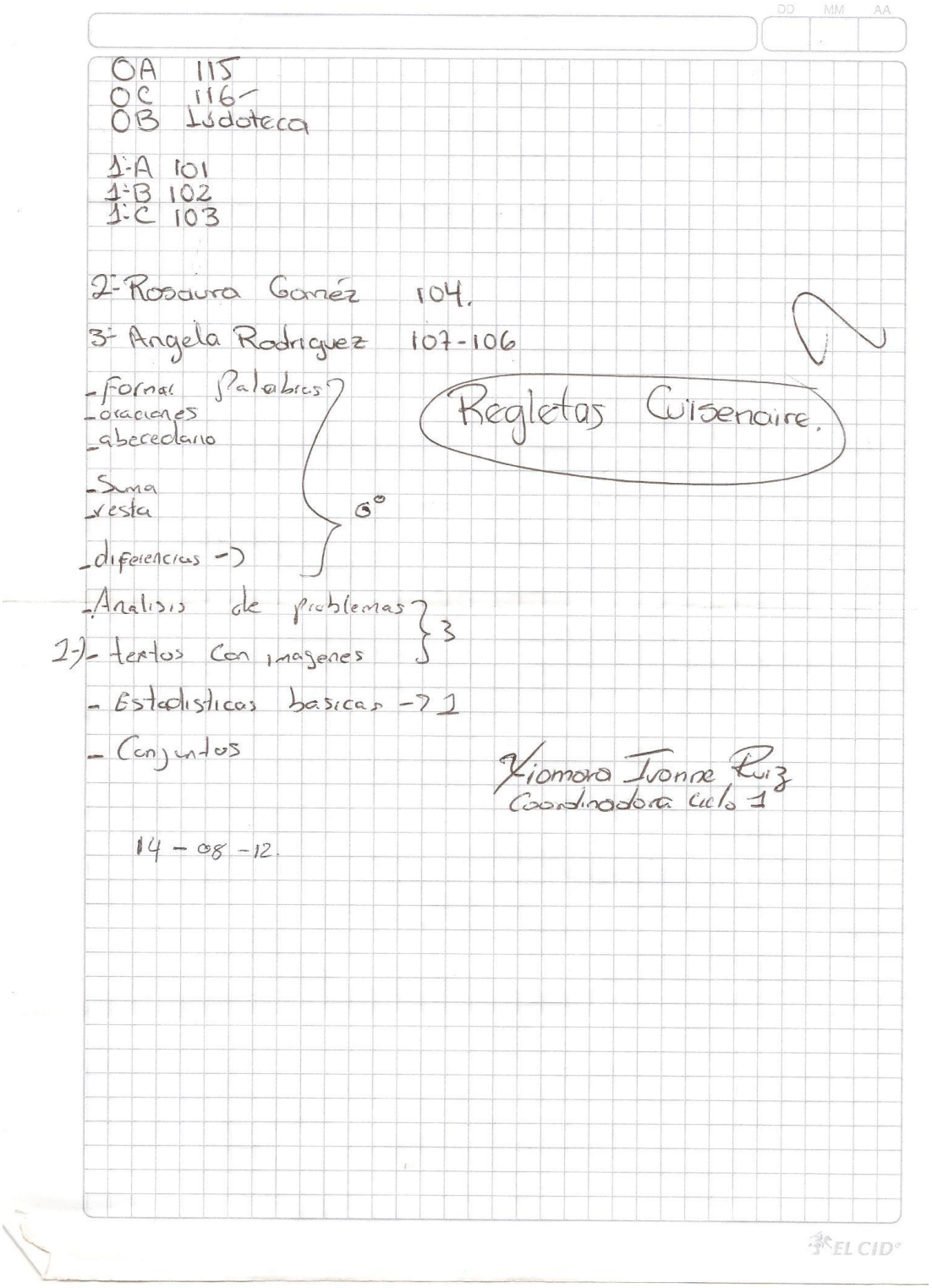
NORMA TÉCNICA COLOMBIANA NTC 1486 (Sexta actualización)

EJEMPLO ESQUEMA DE LA PÁGINA DE LA INTRODUCCIÓN



23

ANEXO IV OTROS



OA 115
OC 116
OB Ludoteca

1-A 101
1-B 102
1-C 103

2- Rosaura Gamez 104.

3- Angela Rodriguez 107-106

- formar Palabras
- oraciones
- abecedario

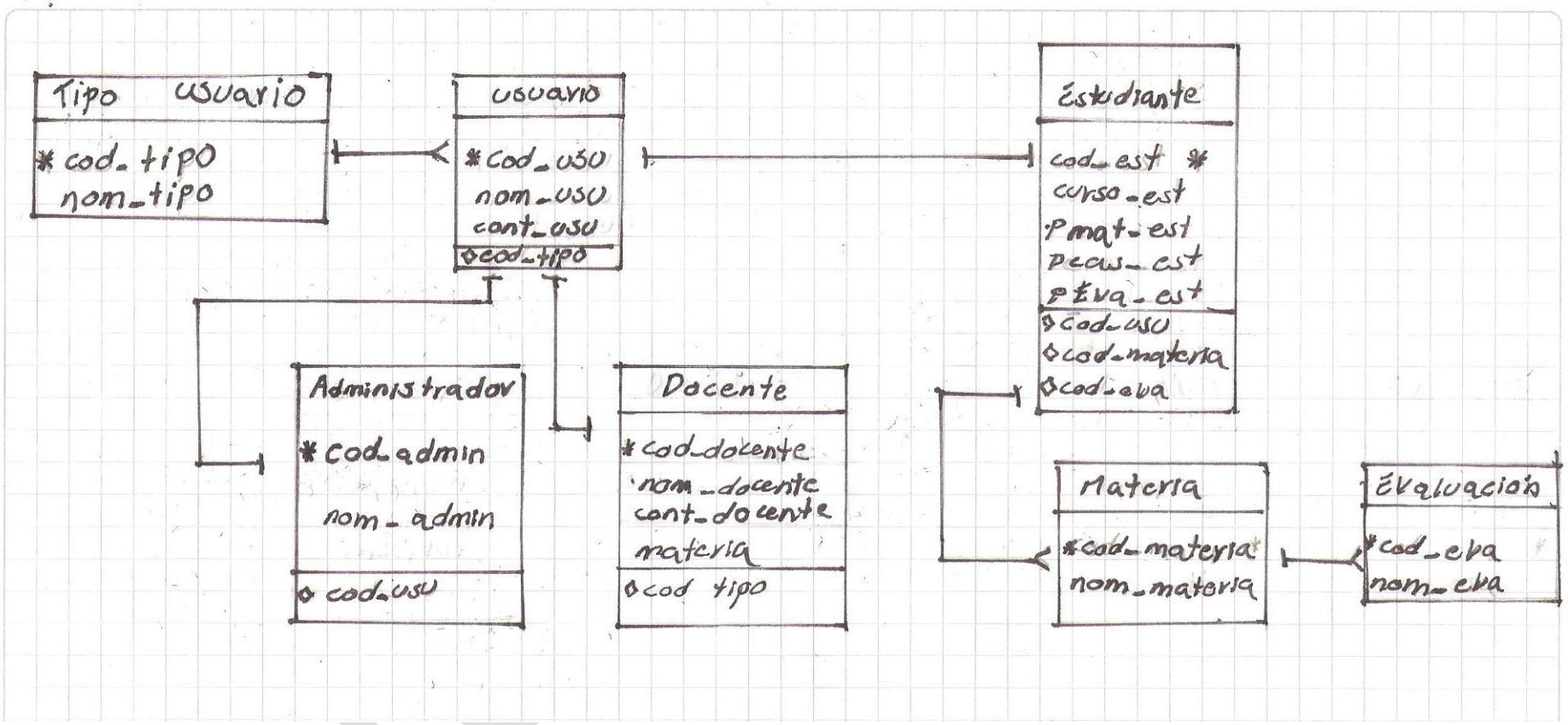
- Suma
- resta
- diferencias -) 6°

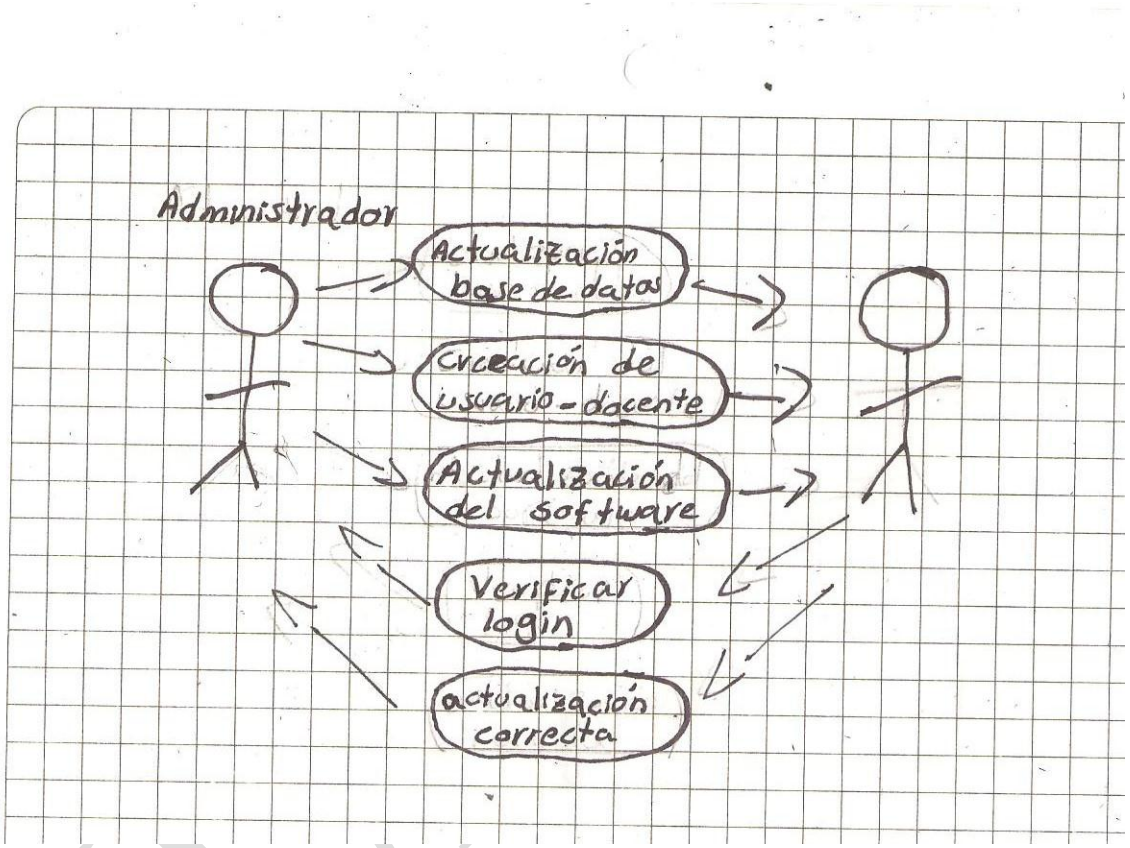
- Analisis de problemas
1)- textos con imagenes } 3
- Estadisticas basicas -> 1
- Conjuntos

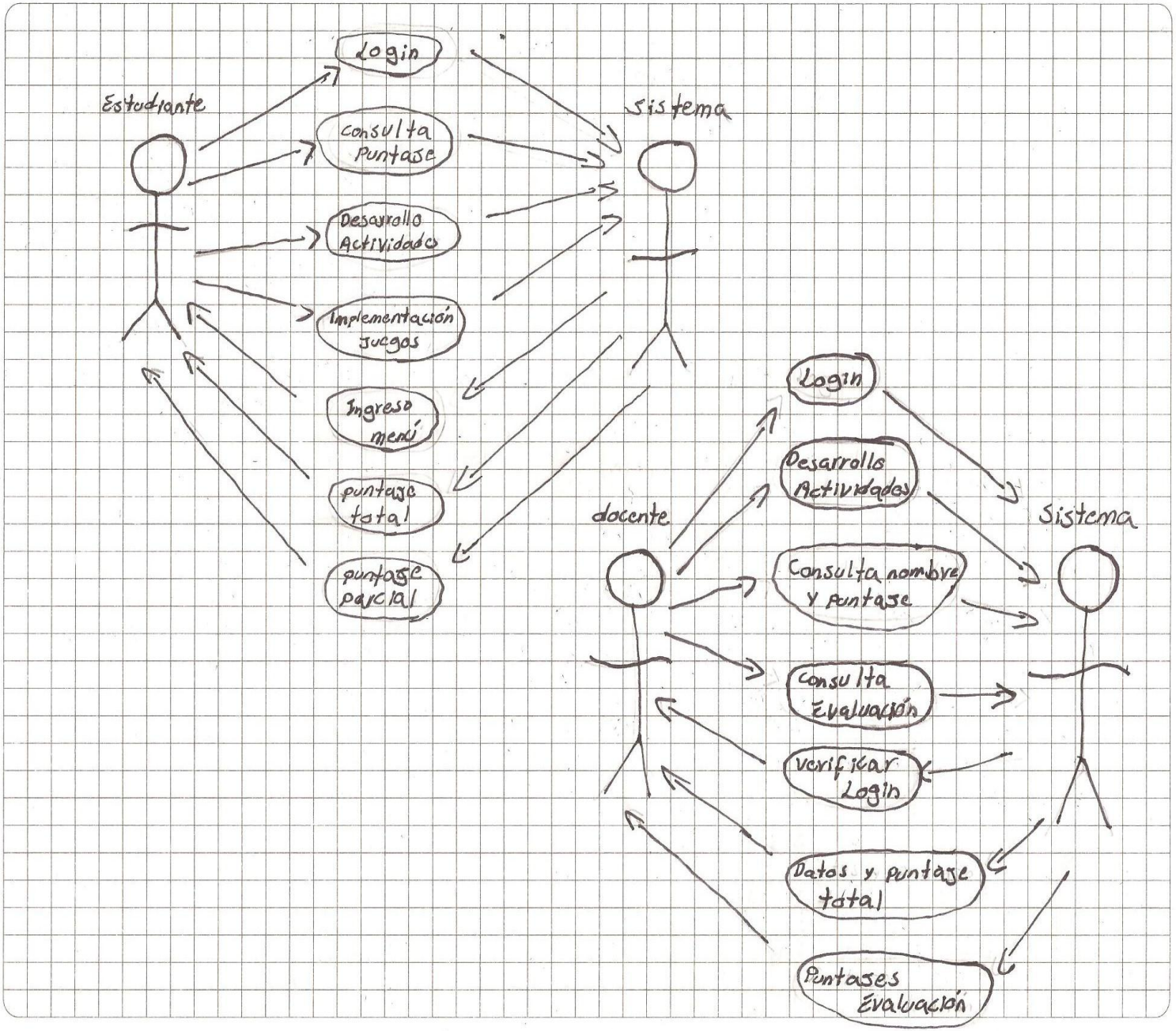
Regletas Cuisenaire.

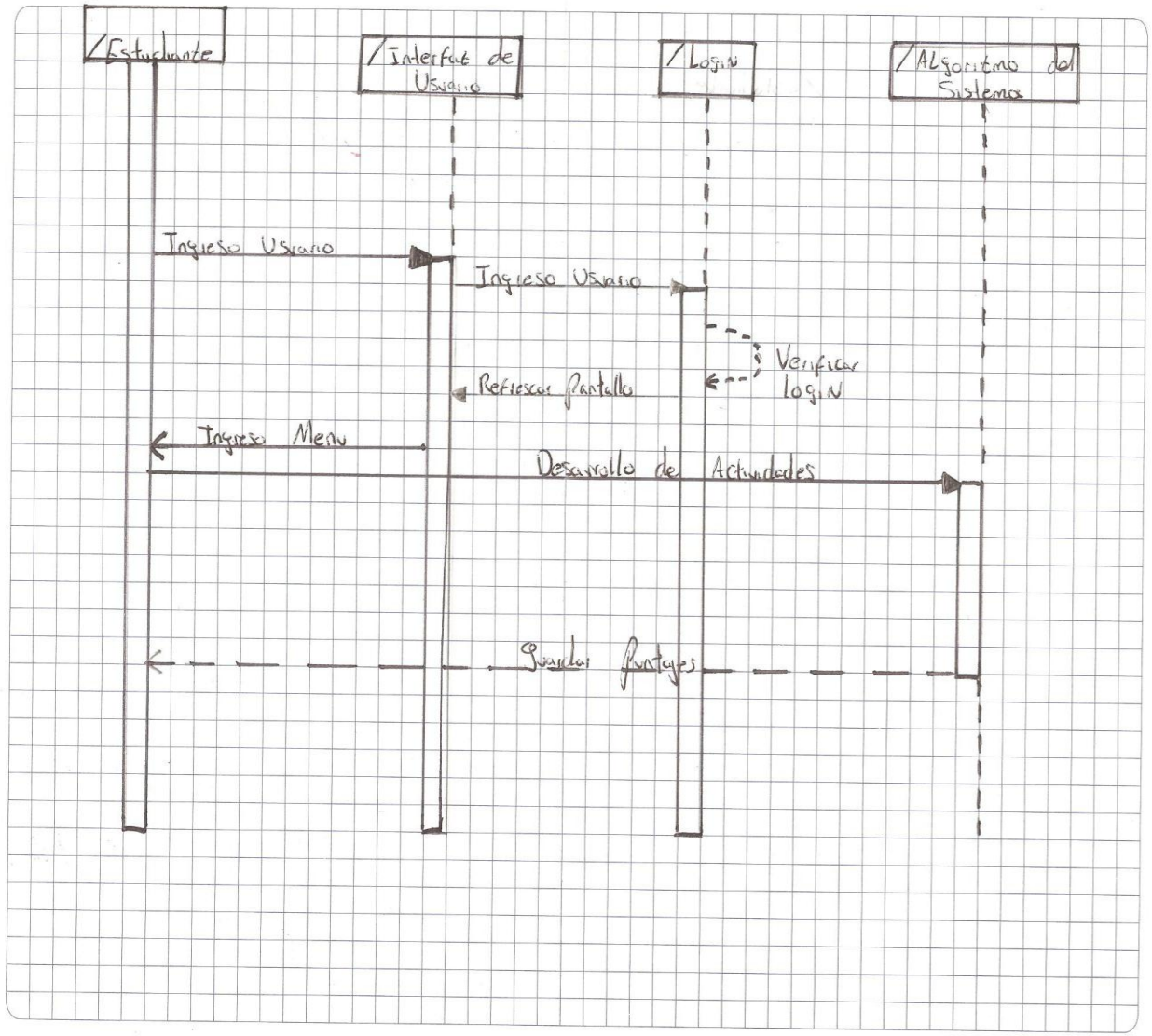
Yiomara Ivonne Ruiz
Coordinadora ciclo 1

14 - 08 - 12.


Tipo usuario
*cod_tipo
nom_tipo
usuario
*cod_usu
nom_usu
cont_usu
◊cod_tipo
Estudiante
cod_est *
curso_est
Pmat_est
Pcas_est
Peva_est
◊cod_usu
◊cod_materia
◊cod_eva
Administrador
*cod_admin
nom_admin
◊cod_usu
Docente
*cod_docente
nom_docente
cont_docente
materia
◊cod tipo
Materia
*cod_materia
nom_materia
Evaluación
*cod_eva
nom_eva


Administrador
Actualización base de datos
Creación de usuario - docente
Actualización del software
Verificar login
actualización correcta


Estudiante
Login
consulta puntaje
Desarrollo Actividades
Implementación juegos
Ingreso menú
puntaje total
puntaje parcial
Sistema
docente
Login
Desarrollo Actividades
Consulta nombre y puntaje
consulta Evaluación
verificar Login
Datos y puntaje total
Puntajes Evaluación
Sistema


Estudiante
Interfaz de Usuario
Login
Algoritmo del Sistema
Ingreso Usuario
Ingreso Usuario
Verificar login
Refrescar pantalla
Ingreso Menu
Desarrollo de Actividades
Guardar Puntajes