



Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ingeniería de Ciencias y Sistemas

**DISEÑO DE UN SISTEMA DE SIMULACIÓN DE EJERCICIOS DE  
TÁCTICAS MILITARES Y ESTRATÉGICAS PARA EL COMANDO  
SUPERIOR DE EDUCACIÓN DEL EJÉRCITO**

**René Gustavo Catalán Gudiel**

**Feliciano Gerardo Charchalac Ochoa**

Asesorado por el Ing. Armín Mazariegos Rabanales

Guatemala, noviembre de 2009

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERIA

**DISEÑO DE UN SISTEMA DE SIMULACIÓN DE EJERCICIOS DE TÁCTICAS  
MILITARES Y ESTRATEGIAS PARA EL COMANDO SUPERIOR DE  
EDUCACIÓN DEL EJÉRCITO.**

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA  
FACULTAD DE INGENIERÍA  
POR

**RENÉ GUSTAVO CATALAN GUDIEL  
FELICIANO GERARDO CHARCHALAC OCHOA**

ASESORADO POR EL ING. JORGE ARMÍN MAZARIEGOS RABANALES

AL CONFERÍRSELES EL TÍTULO DE

**INGENIERO EN CIENCIAS Y SISTEMAS**

GUATEMALA, NOVIEMBRE DE 2009

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA  
FACULTAD DE INGENIERÍA



**NÓMINA DE JUNTA DIRECTIVA**

DECANO	Ing. Murphy Olympto Paiz Recinos
VOCAL I	Inga. Glenda Patricia García Soria
VOCAL II	Inga. Alba Maritza Guerrero e López
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. José Milton De León Bran
VOCAL V	Br. Isaac Sultán Mejía
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

**TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO**

DECANO	Ing. Murphy Olympto Paiz Recinos
EXAMINADOR	Ing. Marlon Antonio Pérez Turk
EXAMINADORA	Inga. Floriza Felipa Ávila Pesquera de Medinilla
EXAMINADORA	Inga. Sonia Yolanda Castañeda Ramírez
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

## **HONORABLE TRIBUNAL EXAMINADOR**

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presentamos a su consideración nuestro trabajo de graduación titulado:

### **DISEÑO DE UN SISTEMA DE SIMULACIÓN DE EJERCICIOS DE TÁCTICAS MILITARES Y ESTRATÉGICAS PARA EL COMANDO SUPERIOR DE EDUCACIÓN DEL EJÉRCITO,**

tema que nos fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, en septiembre de 2008.

Rene Gustavo Catalán Gudiel

Feliciano Gerardo Charchalac Ochoa

Guatemala, 20 de julio de 2009

Ingeniera  
Norma Ileana Sarmiento Zeceña  
Directora Unidad EPS Facultad de Ingenieria  
Universidad de San Carlos de Guatemala  
Presente

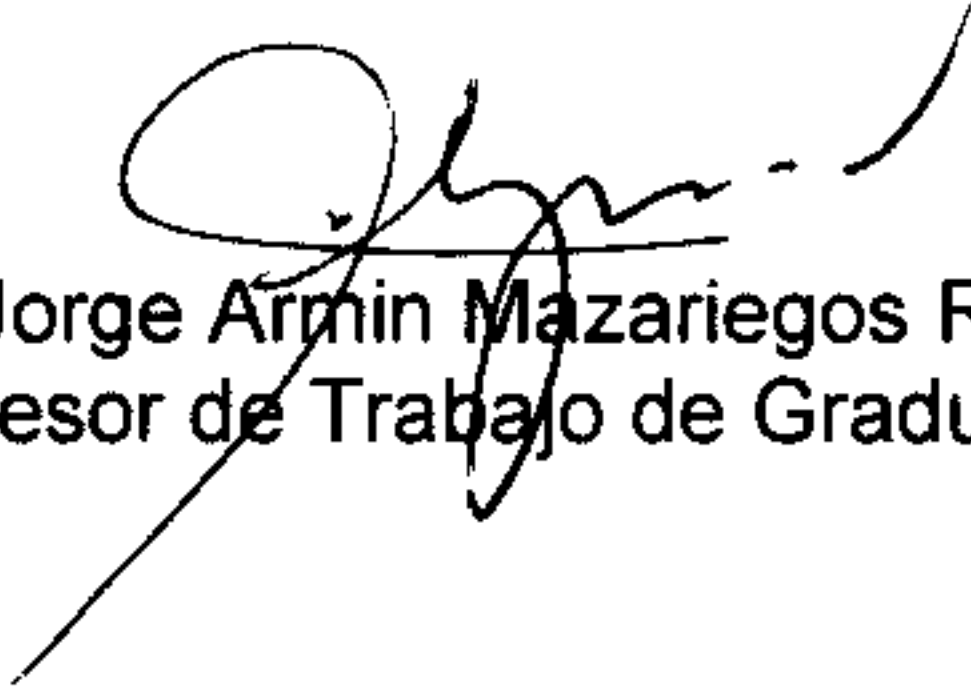
Señora Ingeniera:

Sirva la presente para informarle que asesore al estudiante universitario de la Carrera de Ingenieria en Ciencias y Sistemas, **FELICIANO GERARDO CHARCHALAC OCHOA**, en su Trabajo de Graduacion titulado: **DISEÑO DE UN SISTEMA DE SIMULACION DE EJERCICIOS DE TACTICAS MILITARES Y ESTRATEGICAS PARA EL COMANDO SUPERIOR DE EDUCACION DEL EJERCITO.**

Habiendo cumplido con los objetivos y requisitos de ley apruebo su contenido en mi calidad de Asesor, solicitandole darle el tramite respectivo.

Sin otro particular me suscribo de Usted,

Deferentemente,



Ing. Jorge Armin Mazariegos Rabanales  
Asesor de Trabajo de Graduacion

Guatemala, 20 de julio de 2009

Ingeniera  
Norma Ileana Sarmiento Zeceña  
Directora Unidad EPS Facultad de Ingenieria  
Universidad de San Carlos de Guatemala  
Presente

Señora Ingeniera:

Sirva la presente para informarle que asesore al estudiante universitario de la Carrera de Ingenieria en Ciencias y Sistemas, **RENE GUSTAVO CATALAN GUDIEL**, en su Trabajo de Graduacion titulado: **DISEÑO DE UN SISTEMA DE SIMULACION DE EJERCICIOS DE TACTICAS MILITARES Y ESTRATEGICAS PARA EL COMANDO SUPERIOR DE EDUCACION DEL EJERCITO.**

Habiendo cumplido con los objetivos y requisitos de ley apruebo su contenido en mi calidad de Asesor, solicitandole darle el tramite respectivo.

Sin otro particular me suscribo de Usted,

Deferentemente,

  
Ing. Jorge Armin Mazariegos Rabanales  
Asesor de Trabajo de Graduacion



**UNIDAD DE E.P.S.**

Guatemala, 05 de octubre de 2009.  
Ref.EPS.DOC.1409.10.09.

Inga. Norma Ileana Sarmiento Zeceña de Serrano  
Directora Unidad de EPS  
Facultad de Ingeniería  
Presente

Estimada Ingeniera Sarmiento Zeceña.


Por este medio atentamente le informo que como Supervisora de la Práctica del Ejercicio Profesional Supervisado, (E.P.S) de los estudiantes universitarios de la Carrera de Ingeniería en Ciencias y Sistemas, **René Gustavo Catalán Gudiel** Carné No. **200011639** y **Feliciano Gerardo Charchalac Ochoa** Carné No. **200011364** procedí a revisar el informe final, cuyo título es **“DISEÑO DE UN SISTEMA DE SIMULACIÓN DE EJERCICIOS DE TÁCTICAS MILITARES Y ESTRATÉGICAS PARA EL COMANDO SUPERIOR DE EDUCACIÓN DEL EJÉRCITO”**.

En tal virtud, **LO DOY POR APROBADO**, solicitándole darle el trámite respectivo.

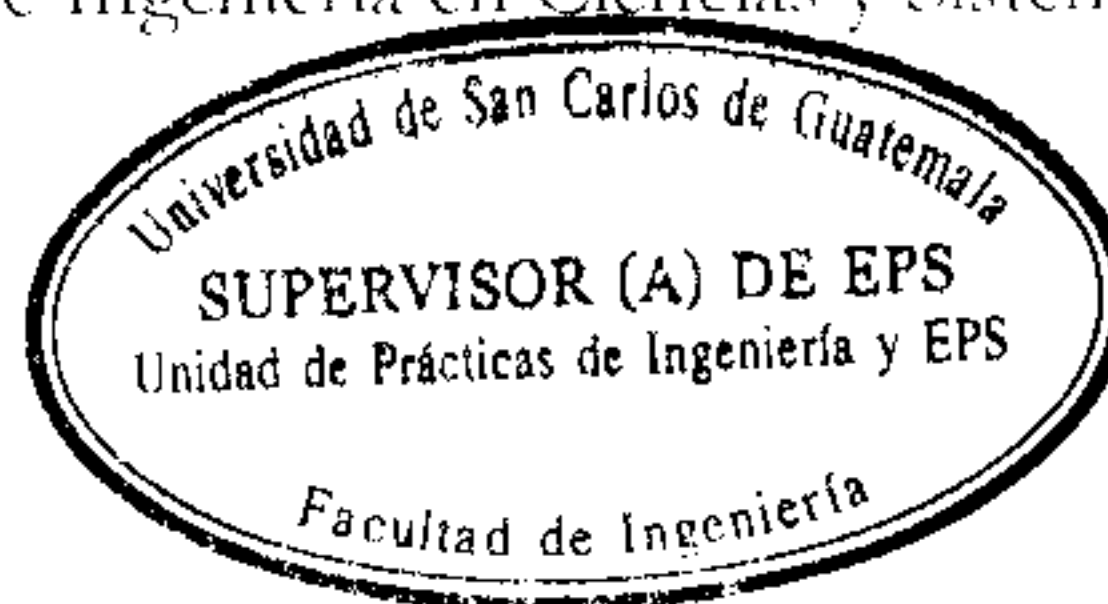
Sin otro particular, me es grato suscribirme.

Atentamente,

*“Id y Enseñad a Todos”*

  
Inga. Floriza Felipa Avila Pesquera de Medinilla  
Supervisora de EPS  
Área de Ingeniería en Ciencias y Sistemas

FFAPdM/RA





**UNIDAD DE E.P.S.**

Guatemala, 05 de octubre de 2009.  
Ref.EPS.D.645.10.09.

Ing. Marlon Antonio Pérez Turck  
Director Escuela de Ingeniería Ciencias y Sistemas  
Facultad de Ingeniería  
Presente

Estimado Ingeniero Perez Turck.

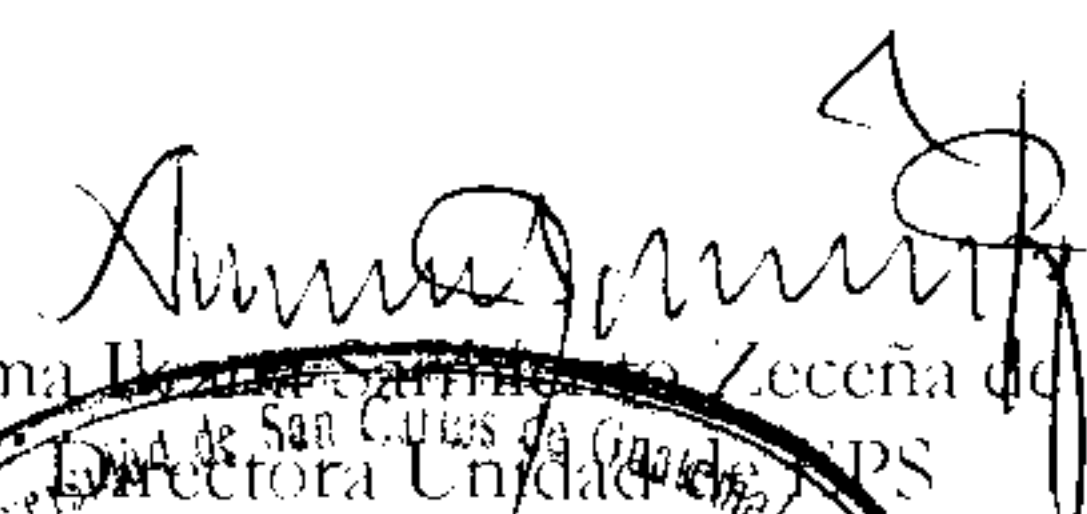
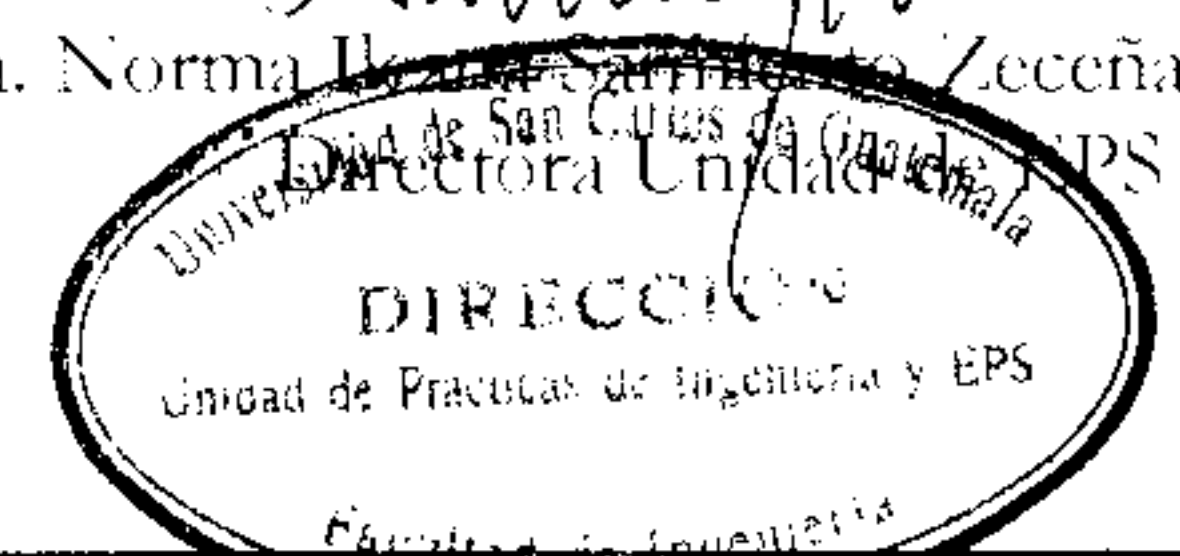
Por este medio atentamente le envío el informe final correspondiente a la práctica del Ejercicio Profesional Supervisado, (E.P.S) titulado **“DISEÑO DE UN SISTEMA DE SIMULACIÓN DE EJERCICIOS DE TÁCTICAS MILITARES Y ESTRATÉGICAS PARA EL COMANDO SUPERIOR DE EDUCACIÓN DEL EJÉRCITO”**, que fue desarrollado por los estudiantes universitarios **René Gustavo Catalán Gudiel** Carné No. **200011639** y **Feliciano Gerardo Charchalac Ochoa** Carné No. **200011364**, quienes fueron debidamente asesorados por el Ing. Jorge Armin Mazariegos y supervisados por la Inga. Floriza Felipa Ávila Pesquera de Medinilla

Por lo que habiendo cumplido con los objetivos y requisitos de ley del referido trabajo y existiendo la aprobación del mismo por parte del Asesor y de la Supervisora de EPS, en mi calidad de Directora apruebo su contenido solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,

*“Id y Enseñad a Todas”*

  
Inga. Norma Ileana Sambrano Zeceña de Serrano  
Directora Unidad de Prácticas de Ingeniería y EPS  


NISZ/ra





Universidad San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 30 de Septiembre de 2009


Ingeniero  
**Marlon Antonio Pérez Turk**  
Director de la Escuela de Ingeniería  
En Ciencias y Sistemas


Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de los estudiantes **RENE GUSTAVO CATALAN GUDIEL Y FELICIANO GERARDO CHARCHALAC OCHOA**, titulado: **"DISEÑO DE UN SISTEMA DE SIMULACION DE EJERCICIOS DE TACTICAS MILITARES Y ESTRATEGICAS PARA EL COMANDO SUPERIOR DE EDUCACION DEL EJERCITO"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

  
**Ing. Carlos Alfredo Azurdia**  
Coordinador de Privados  
y Revisión de Trabajos de Graduación



E  
S  
C  
U  
E  
L  
A  
  
D  
E  
  
C  
I  
E  
N  
C  
I  
A  
S  
  
Y  
  
S  
I  
S  
T  
E  
M  
A  
S

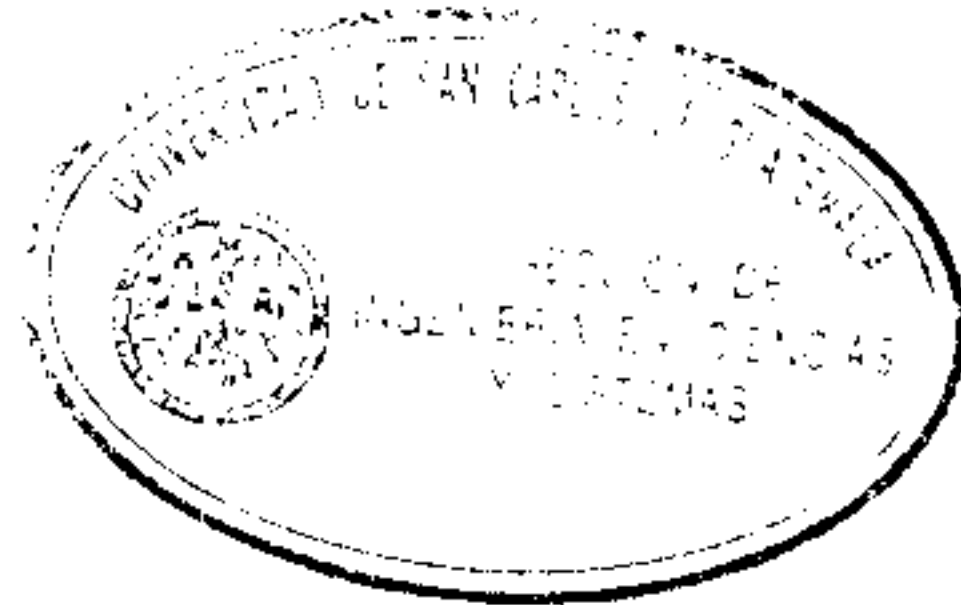
UNIVERSIDAD DE SAN CARLOS  
DE GUATEMALA



FACULTAD DE INGENIERIA  
ESCUELA DE CIENCIAS Y SISTEMAS  
TEL: 24767644

*El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado **"DISEÑO DE UN SISTEMA DE SIMULACIÓN DE EJERCICIOS DE TÁCTICAS MILITARES Y ESTRATÉGICAS PARA EL COMANDO SUPERIOR DE EDUCACIÓN DEL EJÉRCITO"**, presentado por el estudiante René Gustavo Catalán Gudiel y Feliciano Gerardo Charchalac Ochoa, aprueba el presente trabajo y solicita la autorización del mismo.*

**"ID Y ENSEÑAD A TODOS"**



  
Ing. Marlon Antonio Pérez Turk  
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 25 de noviembre 2009



El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **DISEÑO DE UN SISTEMA DE SIMULACIÓN DE EJERCICIOS DE TÁCTICAS MILITARES Y ESTRATÉGICAS PARA EL COMANDO SUPERIOR DE EDUCACIÓN DEL EJÉRCITO**, presentado por los estudiantes universitarios **René Gustavo Catalán Gudiel y Feliciano Gerardo Charchalac Ochoa**, autoriza la impresión del mismo.

IMPRÍMASE.

A handwritten signature in black ink, enclosed within a hand-drawn oval shape.

Ing. Murphy Olimpo Paiz Recinos  
DECANO

Guatemala, noviembre de 2009



/cc  
c.c. archivo.



## **AGRADECIMIENTOS A:**

Dios	Por guiar mis pasos y darme fuerza para superar cada obstáculo que se presenta en el transcurso de mi vida.
Universidad de San Carlos de Guatemala	Por acogerme en su prestigiosa casa de estudios y permitir desarrollarme profesionalmente.
Mis padres	María Leonor Gudiel y José Gonzalo Catalán, por haberme inculcado el deseo de superación personal y profesional con su buen ejemplo.
Mis hermanos	Sandra, Juan y Gaby, por brindarme su apoyo incondicional, por estar siempre presentes en los momentos más importantes de mi vida y porque sin ellos esta meta no hubiese sido una realidad.
Mis sobrinos	Julio, Andrea y Juan Manuel, por ser una razón más para luchar y brindarles un buen ejemplo.
Mis padrinos	Rudy Fuentes y Regina de Fuentes, por darme su amor y apoyo absoluto en cada una de las etapas de mi vida, por creer en mí y motivarme a seguir adelante.
Mi novia	Dra. Carmen Hernández, por sus palabras de aliento, por alegrar mi vida y por darme la oportunidad de conocer la magia de su amor.
Mis compañeros	Por ser mis amigos leales al brindarme su ayuda cuando más la necesité y acompañarme en los buenos y duros momentos a lo largo de mi carrera.

**René Gustavo Catalán Gudiel**

## **DEDICATORIA**

Acto que dedico a mi madre por ser la principal merecedora de tan esperada meta y porque sin duda alguna éste es el resultado del esfuerzo, lucha y sacrificio de toda su vida. ¡Felicidades mamá!

**René Gustavo Catalán Gudiel**

## **AGRADECIMIENTOS A:**

Dios, por haberme dado la sabiduría y entendimiento necesarios a lo largo de estos años de estudio.

Mis padres, Sebastián Francisco Charchalac Santay y Lilian de Carmen Ochoa Cifuentes, por ser mis modelos de vida y ejemplos a seguir.

Mis hermanos, Catalina, Sebastián, Lilian Raquel y Lucrecia, por su apoyo, cariño y las palabras de aliento que siempre me han dado.

La Universidad de San Carlos de Guatemala, y su Facultad de Ingeniería, instituciones que me formaron profesionalmente.

Doña Nuria de Chávez, por brindarme un segundo hogar durante mis años de estudio.

Mis amigos y compañeros de estudio, por compartir los años de Universidad y por la ayuda desinteresada que me han brindado en todo momento, en especial a René Catalán, Evelyn Arita, Ronald Villacinda, Gladys Siliézar y David Argueta.

**Feliciano Gerardo Charchalac Ochoa**

## **DEDICATORIA**

Acto que dedico a mis padres: Sebastián Francisco Charchalac Santay y Lilian del Carmen Ochoa Cifuentes, por su apoyo incondicional en todo momento y porque gracias a su esfuerzo y dedicación logro culminar mis estudios con éxito.

**Feliciano Gerardo Charchalac Ochoa**

## ÍNDICE GENERAL

<b>ÍNDICE DE ILUSTRACIONES</b>	VI
<b>GLOSARIO</b>	IX
<b>RESUMEN</b>	XI
<b>OBJETIVOS</b>	XIII
<b>INTRODUCCIÓN</b>	XV
<b>1. UN POCO DE UML</b>	1
<b>1.1 Diagrama del sistema</b>	3
1.1.1 Diagramas de estructura	4
1.1.1.1 Diagrama de clases	4
1.1.1.2 Diagrama de componentes	5
1.1.1.3 Diagrama de objetos	7
1.1.1.4 Diagrama de despliegue	7
1.1.1.5 Diagrama de paquetes	8
1.1.2 Diagrama de comportamiento	10
1.1.2.1 Diagrama de casos de uso	10
1.1.2.2 Diagrama de estado	13
1.1.2.3 Diagrama de actividades	14
1.1.2.4 Diagrama de secuencia	16
1.1.2.5 Diagrama de colaboración	17
<b>1.2 Acerca del ciclo de vida del <i>software</i></b>	18
<b>1.3 Investigación preliminar</b>	20
1.3.1 Factibilidad técnica	21
1.3.2 Factibilidad económica	22
1.3.3 Factibilidad operacional	22
1.3.4 Factibilidad legal	23
<b>1.4 Análisis</b>	24
<b>1.5 Diseño</b>	25



1.6	<b>Desarrollo del <i>software</i></b>	26
1.7	<b>Pruebas</b>	26
1.8	<b>Implementación y evaluación</b>	27
2.	<b>DISEÑO DE <i>GAMEPLAY</i></b>	29
2.1	<b>Juegos de simulación</b>	29
2.2	<b>Mecánica general de los juegos de simulación</b>	30
2.2.1	Estructura	30
2.3	<b><i>Gameplay</i></b>	32
2.3.1	Controles	32
2.3.2	Objetivo de juego	32
2.4	<b>Ambiente del juego</b>	33
2.4.1	El mapa	33
2.4.2	Unidades	33
2.4.3	Unidades de infantería	34
2.4.4	Unidades de tanques	34
2.4.5	Unidades de artillería	34
2.4.6	Otras unidades	34
2.4.7	Identificación de las unidades	35
2.5	<b>Simbología</b>	35
2.5.1	Símbolos que representan las agrupaciones	35
2.5.2	Símbolos que representan equipamiento	36
2.5.2.1	Color de las unidades	38
2.5.2.2	Unidades al inicio del juego	38
2.5.2.3	Orden de operaciones	38
2.6	<b>Escenarios</b>	38
3	<b>ESPECIFICACIONES SUPLEMENTARIAS</b>	41
3.1	<b>Propósitos</b>	41
3.2	<b>Alcances</b>	41

<b>3.3</b>	<b>Definiciones y abreviaciones</b>	41
<b>3.4</b>	<b>Seguridad</b>	42
3.4.1	<i>Firewalls de Hardware</i>	43
3.4.2	<i>Firewalls de Software</i>	43
<b>3.5</b>	<b>Usabilidad</b>	43
3.5.1	Facilidad de uso	43
3.5.2	Desarrollo de tareas	44
3.5.3	Ayuda para el usuario	44
<b>3.6</b>	<b>Confiabilidad</b>	44
3.6.1	Disponibilidad	44
3.6.2	Disponibilidad de información	45
3.6.2.1	Base de datos en paralelo	45
3.6.2.2	Recuperación de desastres	45
3.6.3	Precisión	45
3.6.4	Requerimiento de confiabilidad	45
<b>3.7</b>	<b>Desempeño</b>	46
3.7.1	Velocidad	46
3.7.2	Capacidad	46
3.7.3	Tiempo de respuesta	47
3.7.4	Tiempo medio de recuperación	47
<b>3.8</b>	<b>Soportabilidad</b>	47
3.8.1	Adaptabilidad	47
3.8.2	Compatibilidad	48
3.8.3	Extensibilidad	48
3.8.3.1	<i>Software</i>	48
3.8.3.2	<i>Hardware</i>	48
4.8.4	Mantenibilidad	48
4.8.5	Configurabilidad	49

<b>3.9</b>	<b>Restricciones de diseño</b>	49
<b>4.</b>	<b>ARQUITECTURA DEL SISTEMA</b>	51
<b>4.1</b>	<b>Propósito</b>	51
<b>4.2</b>	<b>Ámbito</b>	51
<b>4.3</b>	<b>Definiciones, acrónimos y abreviaciones</b>	51
<b>4.4</b>	<b>Referencias</b>	52
<b>4.5</b>	<b>Vista general</b>	52
<b>4.6</b>	<b>Representación de la arquitectura</b>	52
<b>4.7</b>	<b>Metas y límites de la arquitectura</b>	52
<b>4.8</b>	<b>Vista del caso de uso</b>	53
4.8.1	Actores involucrados	55
4.8.1.1	Actividad del actor jugador	55
4.8.1.2	Actividad del actor evaluador	58
4.8.1.3	Actividad del actor administrador	59
<b>4.9</b>	<b>Caso de uso</b>	60
4.9.1	Login	60
4.9.2	Jugar	60
4.9.3	Crear juego	60
4.9.4	Asignar roles	60
4.9.5	Revisar estadísticas	61
4.9.6	Crear escenario	61
4.9.7	Crear, modificar y eliminar usuario	61
<b>4.10</b>	<b>Vista lógica</b>	61
4.10.1	Clases del paquete de vista	62
4.10.1.1	Juego	62
4.10.1.2	Login	63
4.10.1.3	Ingreso de datos	63
4.10.1.4	Crear escenario	63

4.10.1.5	Historial	63
4.10.1.6	Asignar roles	63
4.10.2	Clases del paquete vista	64
4.10.2.1	Unidad	65
4.10.2.2	Característica unidad	66
4.10.2.3	Diagnóstico unidad	66
4.10.2.4	Evaluación acción	66
4.10.3	Paquete de datos	66
4.10.3.1	Modelo entidad-relación	68
<b>4.11</b>	<b>Vista de procesos</b>	69
4.11.1	Proceso de ingreso al sistema y autenticación	69
4.11.2	Proceso de crear escenario	70
4.11.3	Proceso de ingreso a un juego	72
4.11.4	Proceso de un juego	73
<b>4.12</b>	<b>Vista despliegue</b>	75
4.12.1	Ambiente de <i>hardware</i>	76
<b>4.13</b>	<b>VISTA de implementación</b>	77
4.13.1	Vista general	77
4.13.2	Manejo de interfaz	78
<b>CONCLUSIONES</b>		79
<b>RECOMENDACIONES</b>		81
<b>BIBLIOGRAFÍA</b>		83

## ÍNDICE DE ILUSTRACIONES

### FIGURAS

1.	Jerarquía de diagramas UML	4
2	Estructura de diagrama de clase sencillo	5
3	Tipos de componentes que pueden aparecer en la construcción de un <i>software</i>	6
4	Diagrama de objetos sencillos	7
5	Diagrama de despliegue	8
6	Diagrama de paquetes	9
7	Diagrama de caso de uso	11
8	Notación de una comunicación directa	12
9	Caso de uso de origen que utiliza la funcionalidad del caso de uso destino	12
10	Caso de uso con una relación de extensión de funcionalidad o comportamiento	13
11	Diagrama de estado	14
12	Diagrama de actividades	15
13	Diagrama de secuencia de mensajes entre un usuario y una aplicación <i>web</i>	16
14	Diagrama de colaboración	17
15	Desarrollo del <i>software</i>	19
16	Vista de casos de usos	54
17	Actores involucrados en el sistema	55
18	Actividades del jugador	57
19	Actividades del evaluador	58
20	Actividades del administrador	59
21	Clases de paquete de vista	62

22	Clases paquete de control	64
23	Diagrama de clases	65
24	Clases del paquete datos	67
25	Diagrama de base de datos	68
26	Proceso de ingreso y autenticación del sistema	69
27	Proceso de crear escenario	70
28	Definición de escenario	71
29	Selección de áreas a trabajar	72
30	Proceso de ingreso a un juego	73
31	Proceso de juego	74
32	Vista de despliegue	75
33	Vista de implementación	77

## **TABLAS**

I	Símbolos que representan las agrupaciones	36
II	Símbolos que representan equipamiento	37



## GLOSARIO

- FAC:** (Frequently Asked Questions) Preguntas más frecuentes, se refiere a aquella lista de preguntas y respuestas que pueden llegar a surgir en un determinado momento dentro de un contexto y para algún tema en particular.
- HTTP:** (Hypertext Transfer Protocol) El protocolo de transferencia de hipertexto, es un sistema de comunicación que permite al usuario visualizar las páginas *web* desde cualquier navegador.
- MOF:** (Meta Object Facility), Metamodelo Facilidad meta objeto, es un lenguaje para definir lenguajes de modelado. Permite a los usuarios definir totalmente nuevos lenguajes a partir de metamodelos. [5]
- OMG:** (Object Management Group) Grupo de gestión de objetos, es una organización que emite recomendaciones para la elaboración de productos *software* para computación de objetos distribuidos. [6]
- PIM:** (Platform Independent Model) Modelo independiente de plataforma, es un modelo de un sistema de *software* o sistema de empresa, que es independiente de la plataforma tecnológica utilizada para su ejecución [7]



**PSM:** (Platform Specific Model) Modelo de plataforma específico, es un modelo de un *software* o sistema de empresa que está vinculada a una plataforma tecnológica (por ejemplo, un lenguaje de programación, sistema operativo o base de datos). [8]

**UML:** (Unified Model Language) Lenguaje modificado de modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. [7]

**DAPLAME:** (Manual de datos de planificación militar escolar) Documento que contiene los datos de uso corriente y necesario sobre las características del material que el Comando Superior de Educación utiliza en sus cursos de Comando y Estado Mayor. [11]

## RESUMEN

El Comando Superior de Educación del Ejército de Guatemala, como parte de la formación académica a oficiales, se encarga de capacitarlos en el proceso inmediato y oportuno de la toma de decisiones, las cuales se basan en el manejo adecuado de un gran volumen de información que les permita obtener la iniciativa y optimizar el empleo de los medios disponibles para la conducción de las operaciones militares.

La capacitación a los oficiales está dividida en dos fases: la primera de ellas es la parte teórica doctrinaria; y la segunda la parte práctica, la cual se ejecuta de manera manual sobre un mapa y es evaluada por oficiales instructores.

En virtud de lo anterior, se diseñó un sistema que ayudará a automatizar la parte práctica del entrenamiento, el cual aprovechará el equipo informático con que cuenta el Ejército y las capacidades de este equipo para simular de una forma realista el desarrollo de un juego de guerra, con base en las instrucciones que se ingresen. Debido a la magnitud del proyecto, en el presente documento se desarrolla únicamente el diseño del sistema y la implementación parcial de la base de datos.

El presente Sistema de Simulación de Estrategias Militares, mediante trabajos aplicados, permitirá a Comandantes, integrantes del Estado Mayor, Comandantes Subalternos o Unidades Menores, tener la oportunidad de practicar y adquirir experiencia en acciones de guerra, es decir, podrá catalogarse como trabajo aplicado, sin tropas.

Así mismo, el sistema se diseñó utilizando UML y un documento de especificación de requerimientos para asegurar que la funcionalidad del mismo se cumpla. Además, se incluyen bocetos que detallan la forma en que se presentarán las pantallas. Igualmente, el documento de diseño se elaboró de una forma fácil de entender el programa de trabajo para la fase de implementación, además, en el diseño se detallan las herramientas que se utilizarán.

## OBJETIVOS

### General:

- Desarrollar un sistema que permita al Ejército de Guatemala, realizar entrenamiento en tácticas militares, que sean adecuadas a los diversos escenarios y en situaciones que se ajusten lo más posible a una realidad bélica o desastres naturales.

### Específicos:

1. Utilizar un lenguaje de modelado como lo es UML para diseñar la herramienta de una manera fácil de entender para los futuros desarrolladores.
2. Identificar claramente los procesos involucrados en el Sistema de Simulación de Estrategias Militares.
3. Reducir el tiempo de evaluación de la parte práctica de los cursos de Comando impartidos por el COSEDE.
4. Diseñar un Sistema de Simulación de Estrategias Militares en donde se pueda establecer una situación tal, que mediante la estructuración de un escenario real o ficticio, se observe el comportamiento y los resultados de la aplicación de decisiones.
5. Diseñar un sistema de simulación de estrategias militares en donde los contrincantes utilicen sus mejores estrategias y así puedan cumplir las misiones que les sean dadas en la orden de operación al inicio de cada escenario.



## INTRODUCCIÓN

Un juego de simulación de estrategias militares, es aquel que mediante la estructuración de escenarios reales o ficticios permite a los usuarios se ajusten lo más posible a una realidad bélica, y con ello puedan poner en práctica todos aquellos conocimientos de estrategias que le sean posibles.

En el presente documento se diseñó un Sistema de Simulación de Estrategias Militares, el cual se llevó a cabo en El Comando Superior de Educación del Ejército de Guatemala, el mismo cubre aquellos requisitos no funcionales del sistema, tales como la confiabilidad, usabilidad, eficiencia, rendimiento y su completa descripción.

Para la elaboración del presente documento fue necesario realizar entrevistas a los oficiales encargados de impartir el curso de Comando y Estado Mayor para conocer el procedimiento actual. Además, se llevaron a cabo varias reuniones de seguimiento para mostrar resultados parciales y para informar acerca de los avances del proyecto.

Así mismo, el documento fue estructurado en cuatro capítulos, los cuales se describen a continuación:

El capítulo uno está constituido por conceptos teóricos relacionados con la construcción de un sistema de *software*, utilizando un lenguaje UML. En el capítulo dos, se describe la mecánica general del juego, los escenarios a utilizar, los objetivos del mismo, el ambiente y la simbología a utilizar en el juego de Simulación de Estrategias Militares.

En el capítulo tres, se describen todos aquellos requerimientos que no son parte funcional del sistema pero que son de gran importancia para el desarrollo de éste. En el capítulo cuatro se desarrolla toda la arquitectura del sistema propuesto en el cual se utiliza una serie de diagramas que permiten visualizar los diferentes aspectos del sistema. Al final del presente informe se presentan las conclusiones y recomendaciones, así mismo, se incluye la bibliografía consultada durante el mismo.

## 1. UN POCO DE UML

Modelar, o llevar a cabo un modelado, es la parte fundamental en la construcción de un sistema de *software* o cualquier sistema en general, debido a que, al realizar un modelo se toma la esencia misma de cualquier sistema como lo son: funcionalidad, características, comportamiento interno y la forma de interacción con los factores externos al sistema mismo, ya sea de forma directa o indirecta. Cuando se piensa trasladar un sistema real, intuitivo y con una lógica funcional definida, hacia una aplicación que produzca los mismos o mejores resultados que el sistema actual, se emprende el reto de asimilar completamente la lógica del sistema, para comprenderlo de una forma adecuada para plasmarlo claramente en un lenguaje entendible, que represente todo el sistema real y la forma en que se “mapeará” a una aplicación de *software*.

Cuando se modela un sistema de *software*, el ciclo de vida del mismo se encuentra concretamente en la fase de diseño, esto quiere decir, que se debe realizar antes de la codificación o del traslado a un lenguaje específico de programación, independientemente de la metodología de desarrollo de *software* que se utilice. Si no está claramente definido en su totalidad cada uno de los procesos y características (requerimientos) del sistema o aplicación de *software* a construir, se tendrá como resultado una fase de desarrollo y/o codificación exhausta, no optimizada, poco escalable y renuente al cambio.

Siempre es necesario tener un nivel de calidad superior de resultados en la fase de análisis del sistema, para lograr un diseño más eficiente y apegado a la realidad del sistema, debido a que un documento de especificación de requerimientos del sistema mediocre y poco coincidente con la realidad, traerá en consecuencia el diseño de un sistema de *software* no funcional por lo tanto no es aplicable para las reglas del negocio y necesidades del problema original.



En la fase de diseño en el ciclo de vida del *software*, se debe contar con un nivel de descripción gráfico capaz de mostrar tanto la estructura del sistema, como el comportamiento e interacción del mismo. Actualmente existen varios lenguajes de modelado, sin embargo, la OMG (Object Management Group) ha desarrollado y estandarizado un lenguaje de modelado muy completo, el UML. Con distintos niveles de abstracción, el UML es capaz de describir claramente los dos ejes principales de un desarrollo de *software* dirigido por modelos (MDD), los cuales son: la funcionalidad del sistema e implementación de dicha funcionalidad a tecnologías específicas. Evidentemente el UML es uno de los más completos que existen actualmente, aunque cabe mencionar también que la OMG desarrolló otro lenguaje distinto llamado MOF (Meta Object Facility), que se utiliza para formalizar la definición de los otros lenguajes de modelado.

Utilizando el UML en el proceso de diseño del sistema de *software*, es posible representar de forma gráfica la independencia entre las distintas partes del sistema real y al mismo tiempo también se refleja claramente las condiciones estáticas y dinámicas del sistema, su estructura o arquitectura e interacción respectivamente.

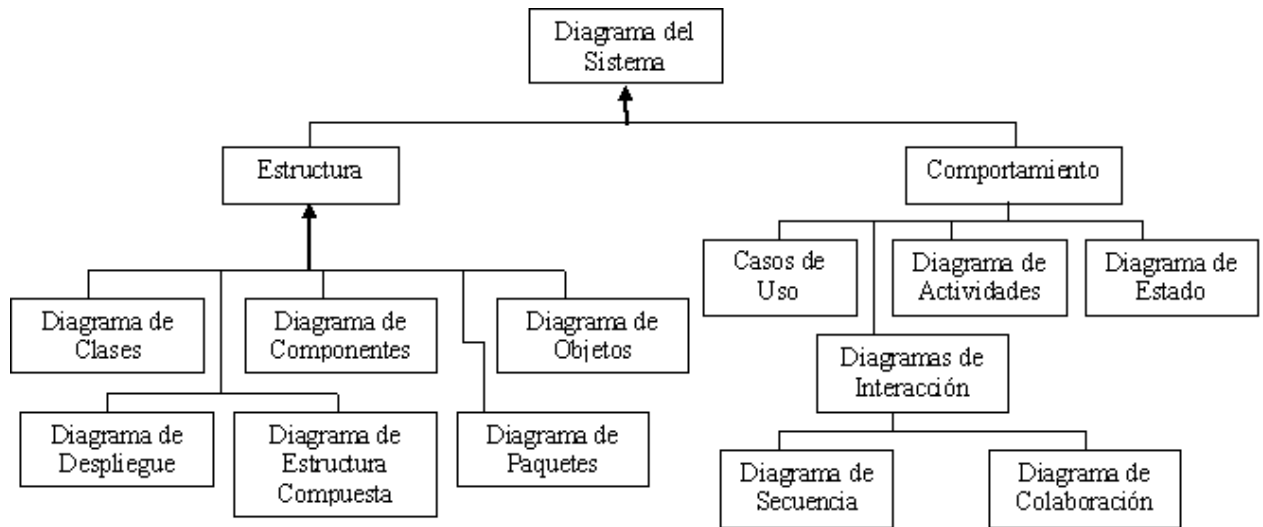
UML, es el lenguaje unificado de modelado que permite modelar un sistema de *software* desde su concepción, hasta su aplicación desde distintos puntos de vista. Su nivel de abstracción hace énfasis en los dos tipos de modelos con los que se define la construcción de una aplicación de *software*, los PIM's y los PSM's. Un PIM, representa la funcionalidad en esencia del sistema real, es decir todo aquello que sea el propósito o razón de ser del sistema, independientemente de la tecnología que se utilice para llevar a cabo dicha funcionalidad. Un PSM, representa todo aquello que sea parte del sistema en términos de implementación, en una tecnología específica, con la cual se construirá la funcionalidad del sistema.

Tomando en cuenta que UML hace clara distinción entre modelos de la lógica del negocio o funcionalidad del proceso original y la implementación tecnológica en la que se construirá la aplicación de *software*, se cuenta de una serie de diagramas completos con nomenclatura clara para la elaboración de los modelos.

### **1.1. Diagramas del sistema**

Los diagramas se encuentran clasificados en: de estructura y de comportamiento. Los de comportamiento describen todos los requerimientos funcionales, la forma en la cual interactúan, se comportan y se comunican los elementos del sistema. Los diagramas de estructura, definen la arquitectura física y conceptual de cómo se construirá el sistema de *software*, identificando claramente la tecnología en la que será implementado.

**Figura 1. Jerarquía de diagramas UML**



El número y tipo de diagramas incluidos en el diseño de una aplicación de *software*, dependerán del esquema de arquitectura que se haya elegido para presentar los modelos.

### **1.1.1. Diagramas de estructura**

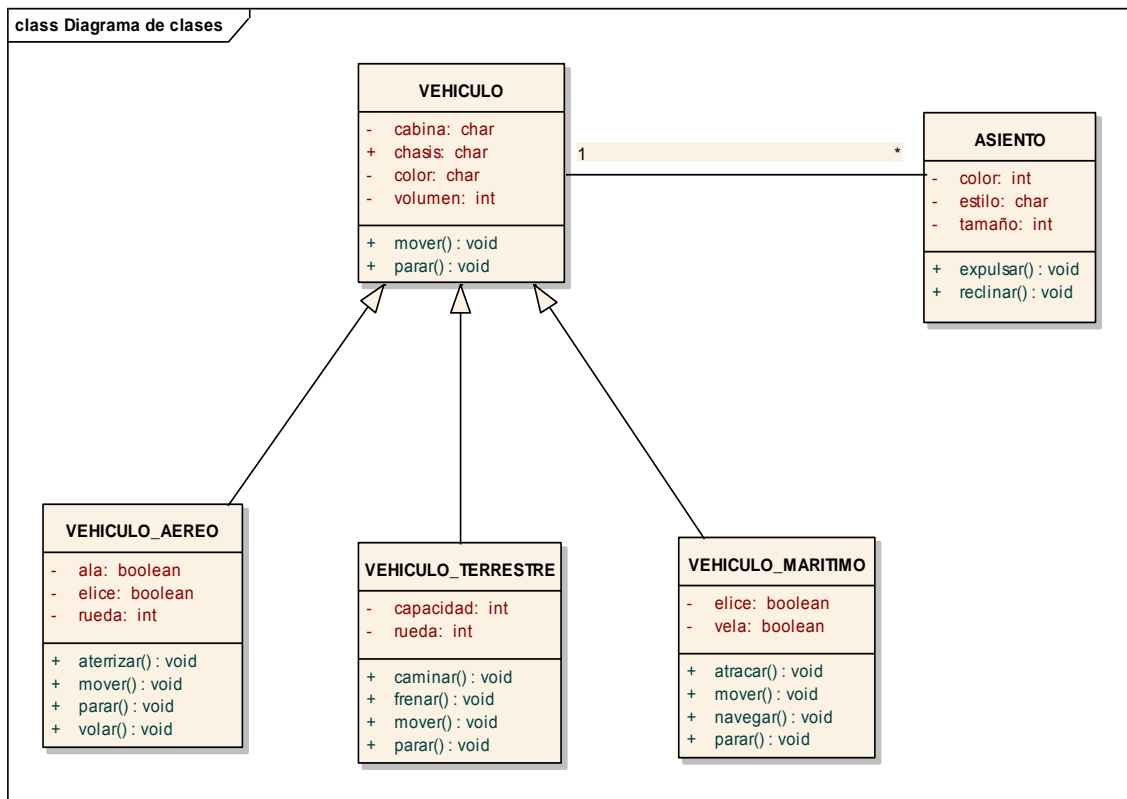
Los diagramas de estructura representan modelos de la arquitectura física y conceptual del sistema, es decir, indica los elementos que deben estar presentes en el sistema modelado.

#### **1.1.1.1. Diagrama de clases**

El diagrama de clases es el núcleo conceptual del sistema modelado. Su concepción se diseña en base al paradigma orientado a objetos. En éste diagrama se reflejan todas las clases, características y acciones que contiene el sistema, así como, la interacción entre dichas clases a lo largo de la ejecución

del mismo. Para conocer la sintaxis completa de un diagrama de clases, se hace referencia al sitio correspondiente a UML publicado por la OMG citado en la bibliografía del presente documento.

**Figura 2. Estructura de diagrama de clase sencillo**

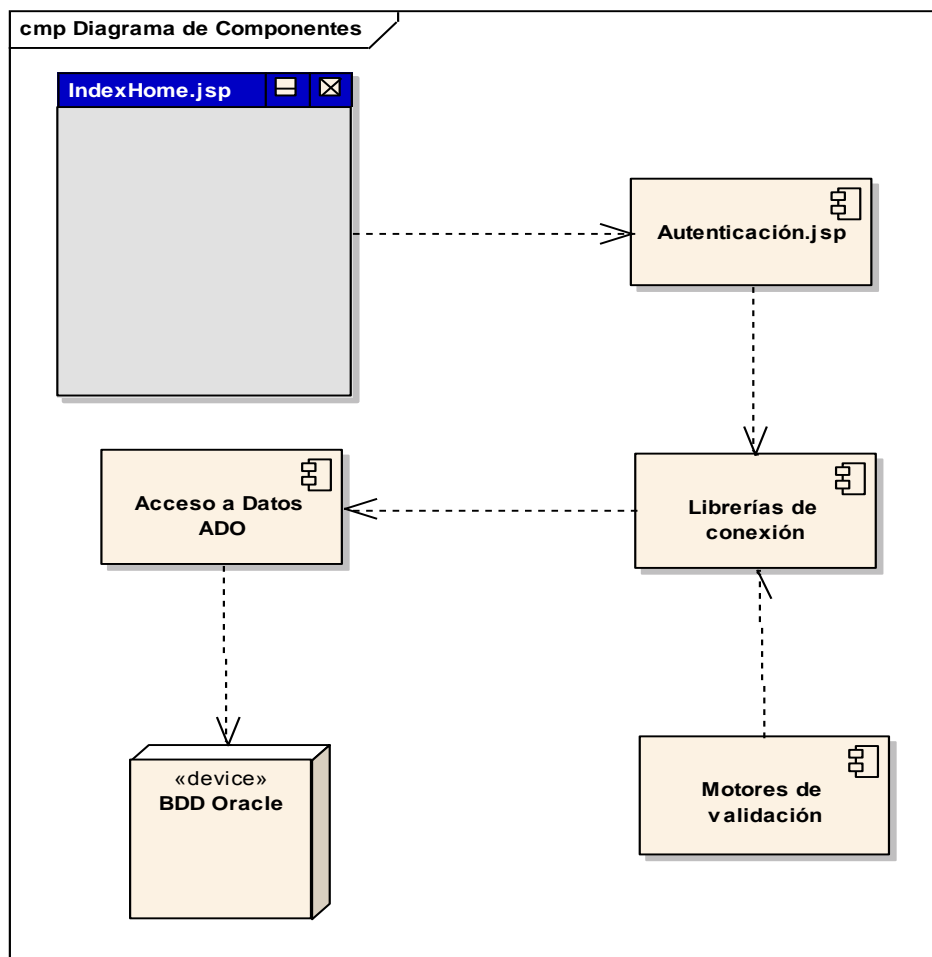


### 1.1.1.2. Diagrama de componentes

En el diagrama de componentes se presentan todos los elementos de *software* físicos que intervienen en el proceso de desarrollo de la aplicación. Dichos componentes pueden ser archivos binarios, librerías y el mismo código fuente, etc. También se definen las relaciones de dependencias entre dichos componentes lo cual indica cuales de esos componentes hacen uso de servicios o funciones que proveen otros componentes. En la figura 3 se muestra un

ejemplo de algunos tipos de componentes que pueden aparecer en la construcción de una aplicación de *software*, obviamente cada diagrama varía dependiendo de qué tecnologías se elijan.

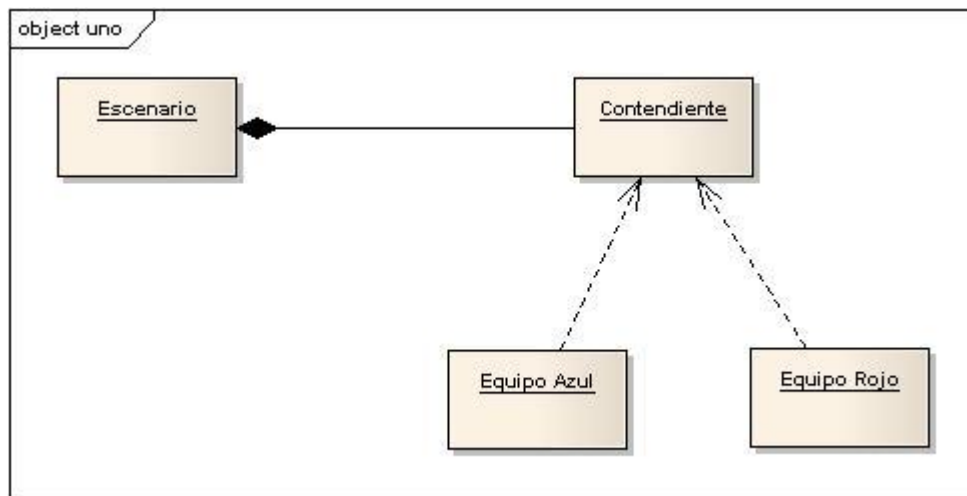
**Figura 3. Tipos de componentes que pueden aparecer en la construcción de un *software***



### 1.1.1.3. Diagrama de objetos

A diferencia del diagrama de clases que muestra la abstracción de las clases de una parte del dominio de la aplicación, el diagrama de objetos muestra las clases en una situación concreta y específica del dominio. El diagrama de objetos permite identificar claramente las instancias de las clases del sistema en algún tiempo determinado. La figura 4 es un ejemplo de un diagrama de objetos sencillos, la notación es casi la misma que el diagrama de clases, con la diferencia que no se especifican los atributos y acciones.

**Figura 4. Diagrama de objetos sencillos**

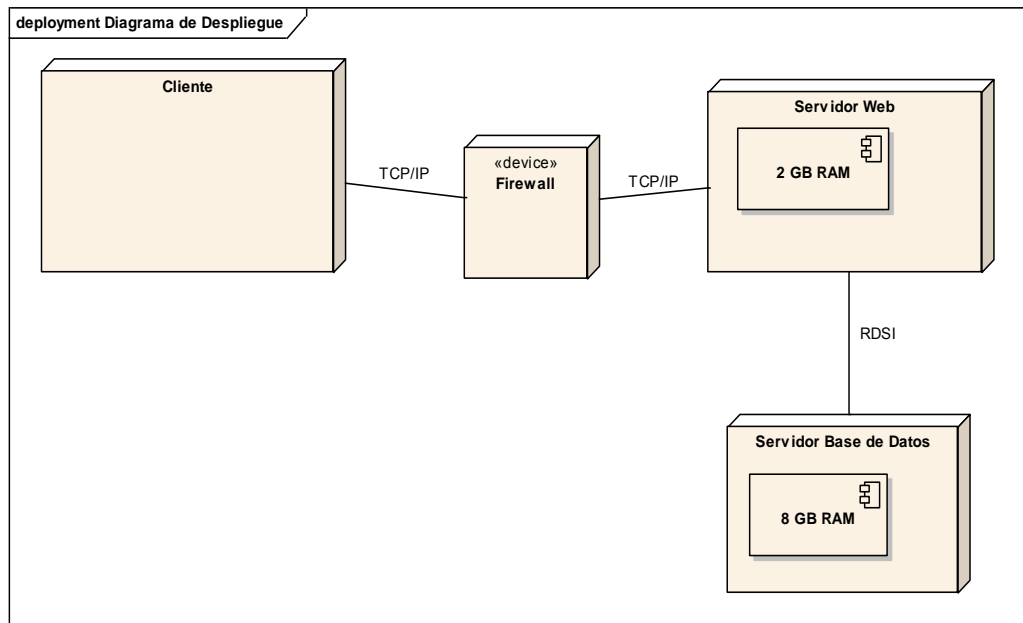


### 1.1.1.4. Diagrama de despliegue

En el diagrama de despliegue se representan los diferentes nodos físicos necesarios para la implementación del sistema, así mismo, se muestra la distribución de todos los componentes físicos descritos en el diagrama de componentes, entre todos los nodos. También se definen los tipos y números

de conexiones entre los nodos para saber cómo interactuarán entre ellos. Para poder identificar que piezas de equipo son necesarias para la aplicación, se utilizan los diferentes estereotipos de equipo, tales como: dispositivo, memoria y procesador. La figura 5 muestra el tipo de comunicación entre cada uno de los componentes del diagrama de despliegue para una arquitectura física de una Aplicación Web.

**Figura 5. Diagrama de despliegue**

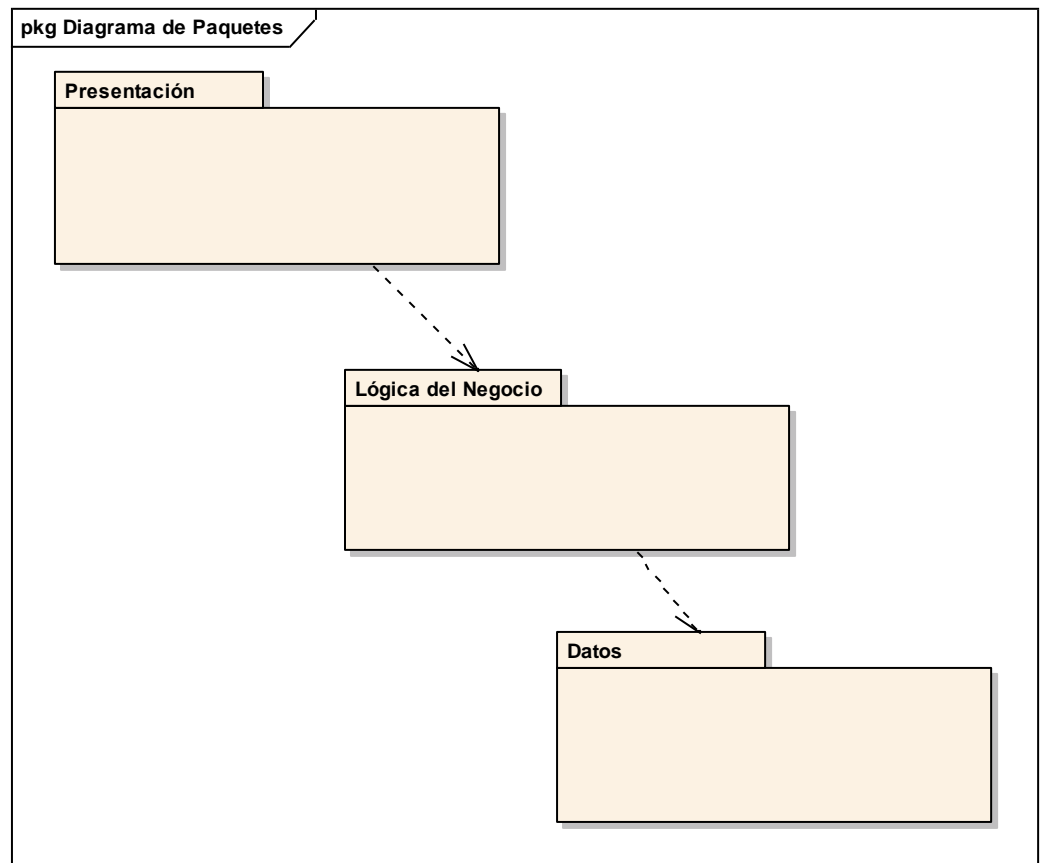


#### 1.1.1.5. Diagrama de paquetes

Los diagramas de paquetes se utilizan para describir la organización lógica del sistema. Se agrupan los distintos elementos de todo el sistema en paquetes, los cuales tienen bien definido la coherencia y la posición que ocupan dentro de la lógica del sistema. Las clases del sistema deben pertenecer a solo un paquete, mientras que cada paquete puede incluir varias de las clases del

sistema. El diagrama de paquetes provee una abstracción y vista general de cómo se compone la construcción del sistema, y el propósito en común que tienen las clases y elementos pertenecientes a un paquete específico.

**Figura 6. Diagrama de paquetes**





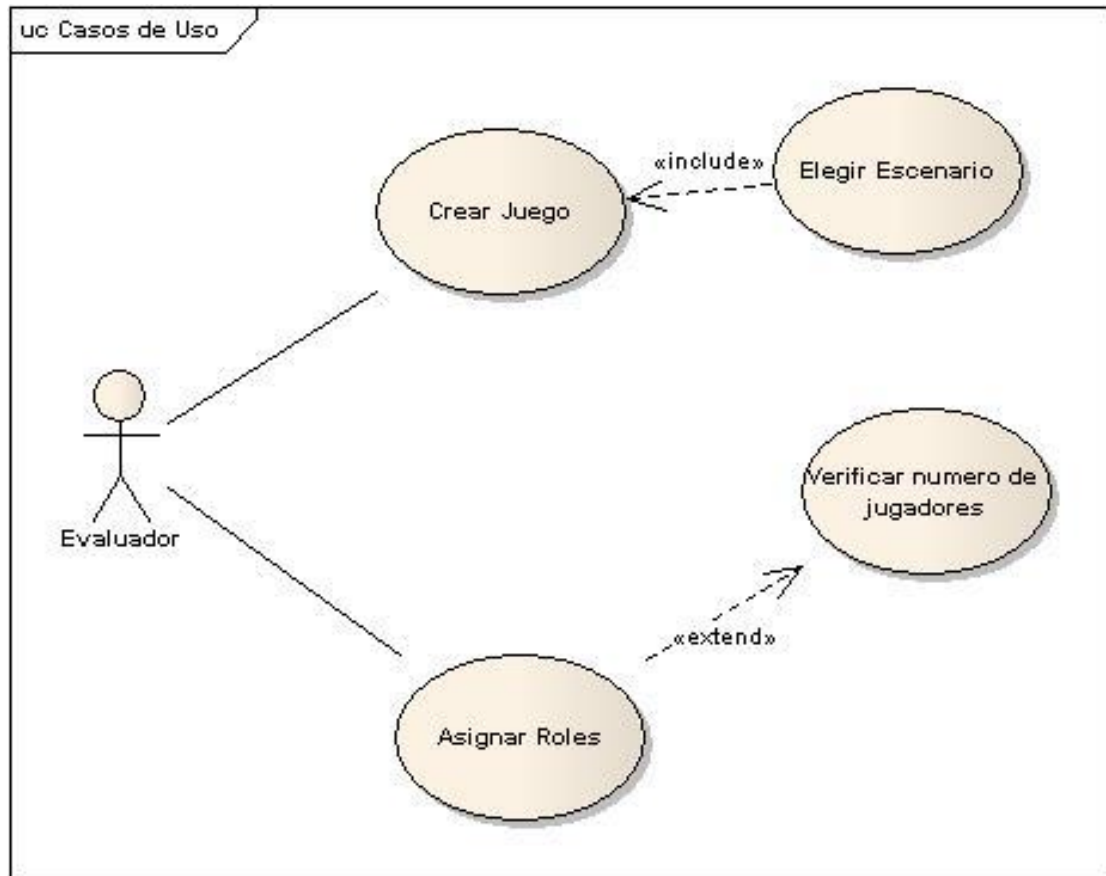
## **1.1.2. Diagramas de comportamiento**

### **1.1.2.1. Diagrama de casos de uso**

Los diagramas de casos de uso son la abstracción de toda la lógica del sistema real a modelar. Con los casos de uso se logra identificar claramente los diferentes escenarios en que se puede encontrar el sistema, además de observar los actores que intervienen e interactúan dentro de todo el proceso de ejecución.

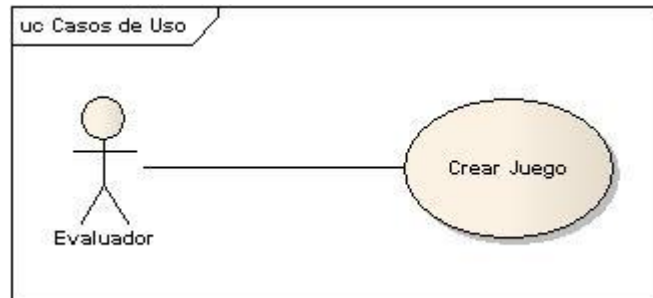
El modelado de casos de uso, es la columna vertebral de todos los modelos, ya que en estos diagramas se expresa la funcionalidad del sistema, independientemente de cuál sea la tecnología que se elija para fabricar el sistema de *software*, es más, sin un diagrama de casos de uso, es muy probable que se escojan tecnologías poco apropiadas para la construcción de la aplicación, las cuales deben coincidir con los propósitos y objetivos del sistema real. En la figura 7 se muestra un diagrama de caso de uso sencillo que identifica tareas como las que un evaluador realiza en el sistema descrito en este documento.

Figura 7. Diagrama de caso de uso



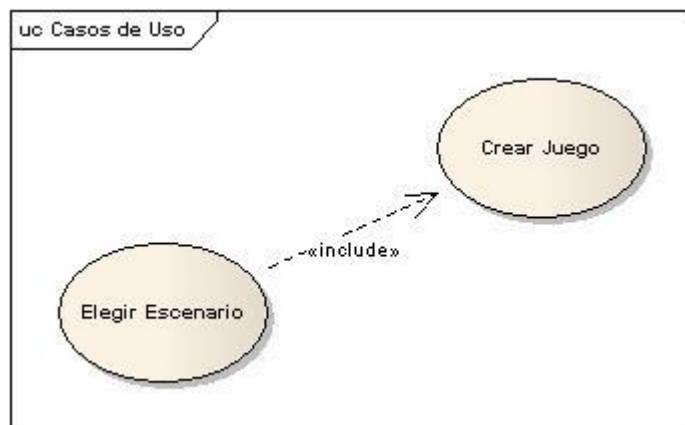
Cuando se describe una línea continua entre un actor y un caso de uso significa que el actor hace uso o interactúa de forma directa con el caso de uso enlazado, es decir, hay comunicación entre el actor y el caso de uso. La figura 8 muestra la notación de una comunicación directa.

**Figura 8. Notación de una comunicación directa**



Cuando se encuentra una línea discontinua enlazando dos casos de uso y dicha línea contenga la leyenda <<include>>, se refiere a que el caso de uso origen utiliza o toma en cuenta la funcionalidad del caso de uso destino [1] para el resultado de sus acciones, así, el caso de uso origen incluye al caso de uso destino para completar sus funciones. .

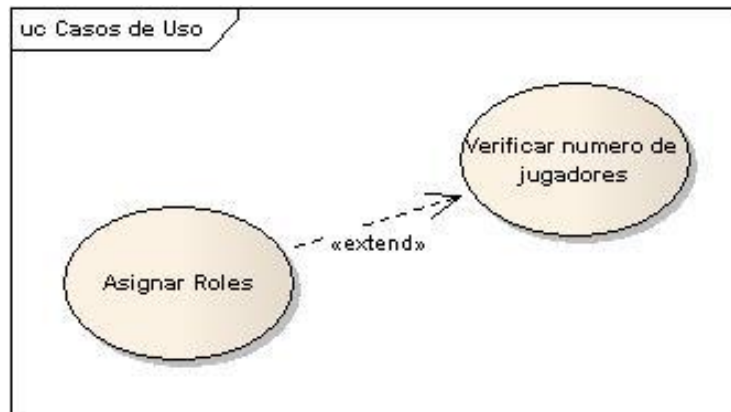
**Figura 9. Caso de uso de origen que utiliza la funcionalidad del caso de uso destino**



De igual forma cuando aparece una línea discontinua con la leyenda <<extend>>, se refiere a que el caso de uso origen extiende el comportamiento

del caso de uso destino [1], es decir, que el caso de uso destino aporta su comportamiento como una porción del caso de uso origen.

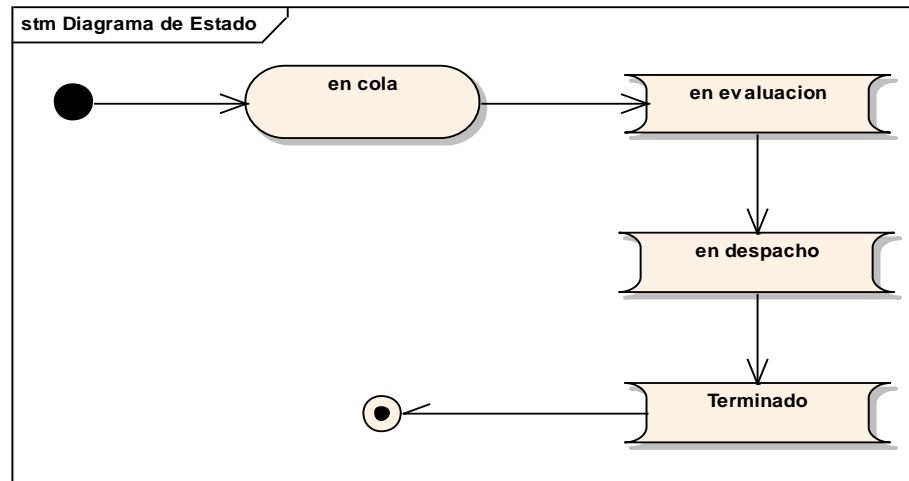
**Figura 10. Caso de uso con una relación de extensión de funcionalidad o comportamiento**



#### 1.1.2.2. Diagrama de estado

El diagrama de estado muestra las distintas facetas o estados por las que pasan los objetos sobresalientes e importantes del sistema. A través de las transiciones, un objeto cambia su comportamiento durante el tiempo de ejecución. Así, un estado se define por los valores de las características propias del objeto en algún momento en particular del tiempo. En la figura 11 se presenta un ejemplo de diagrama de estado para un objeto llamado “formulario”, el cual parte de un estado inicial (al momento de instanciarse) y cambia de estado a través de transiciones marcadas como eventos representadas por flechas de flujo. Cada estado diferente se representa por un rectángulo con esquinas redondeadas, todo esto conlleva al estado final en que el objeto “muere”, aunque en realidad el objeto no se destruye, sino que regresa al nivel de instancia superior del objeto que lo invocó.

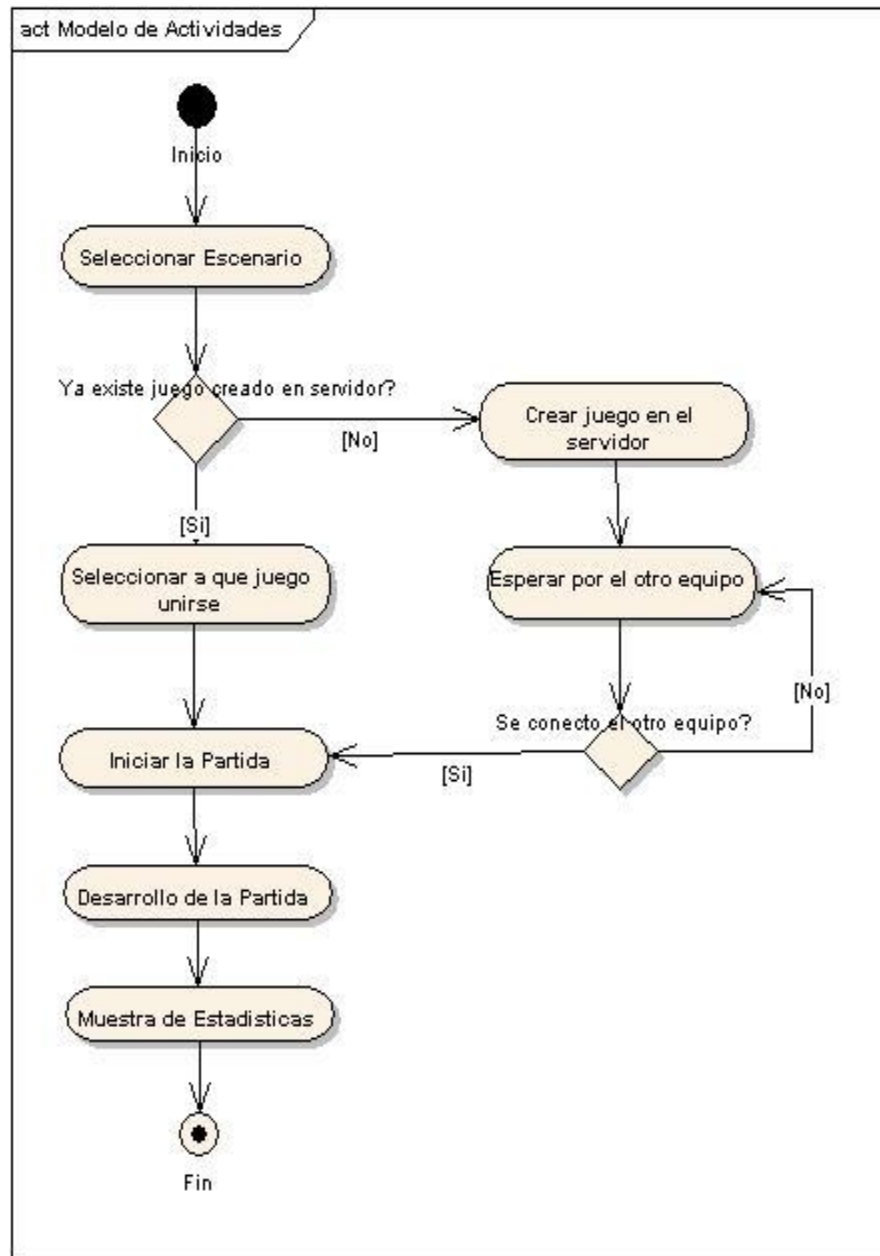
**Figura 11. Diagrama de estado**



### 1.1.2.3. Diagrama de actividades

Sirve de apoyo a los diagramas de caso de uso, ya que brinda un detalle las actividades necesarias de realizar en un caso de uso. Las actividades y los caminos que puede tomar el flujo de información, representan en sí la explicación de cómo se comporta el problema real inicial. A diferencia de los estados del diagrama de estados, las actividades no esperan eventos para ejecutarse, sino más bien, cuando termina la actividad previa se ejecuta la siguiente, hasta que se llega al final del procedimiento. Las actividades se representan con rectángulos con esquinas redondeadas, iguales a los utilizados en el diagrama de estados, de la misma forma se utilizan flechas con líneas continuas para describir las transiciones de una actividad a otra, teniendo en cuenta que si aparece un rombo indica la presencia de una decisión y en consecuencia el flujo de la información se decide continuar por cualquiera de los dos diferentes caminos planteados en la condición.

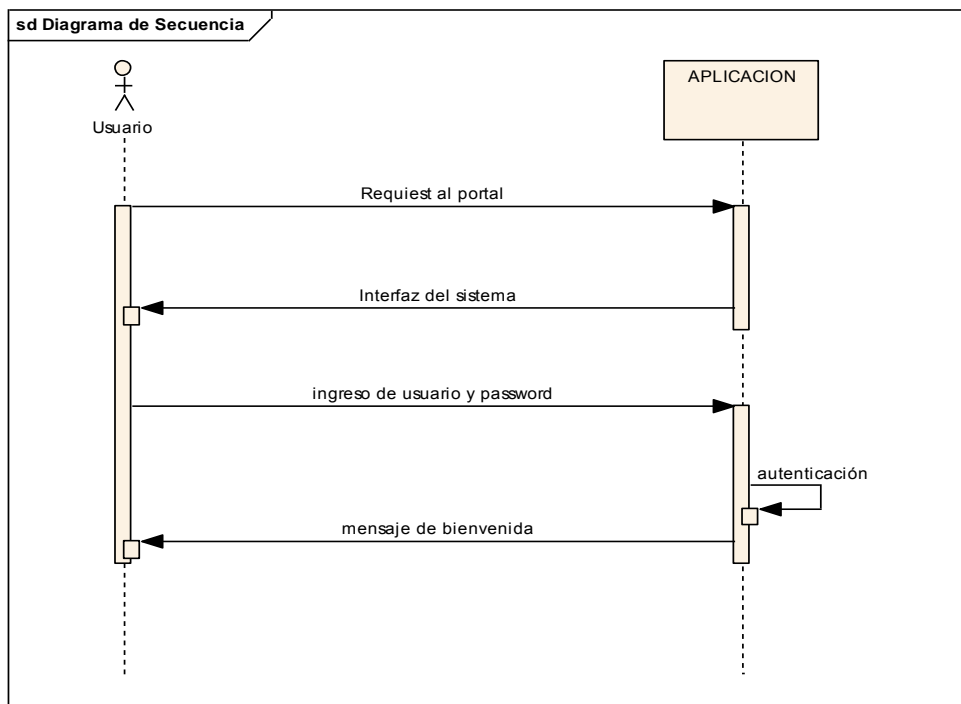
Figura 12. Diagrama de actividades



#### 1.1.2.4. Diagrama de secuencia

Así como el diagrama de casos de uso representa los requisitos funcionales del sistema y muestra la comunicación que existe entre los actores y las partes funcionales del sistema, el diagrama de secuencia muestra el orden del envío de mensajes entre los distintos objetos del sistema, es decir que, indica “¿cómo?” se están mandando los objetos en el caso de uso correspondiente. La notación es muy explícita, los objetos se identifican con rectángulos y una línea de vida vertical que indica la permanencia en el tiempo del objeto creado hasta que finalice su función y se “destruya”. Los mensajes entre los objetos se denotan con flechas rellenas e indicando el texto del contenido del mensaje.

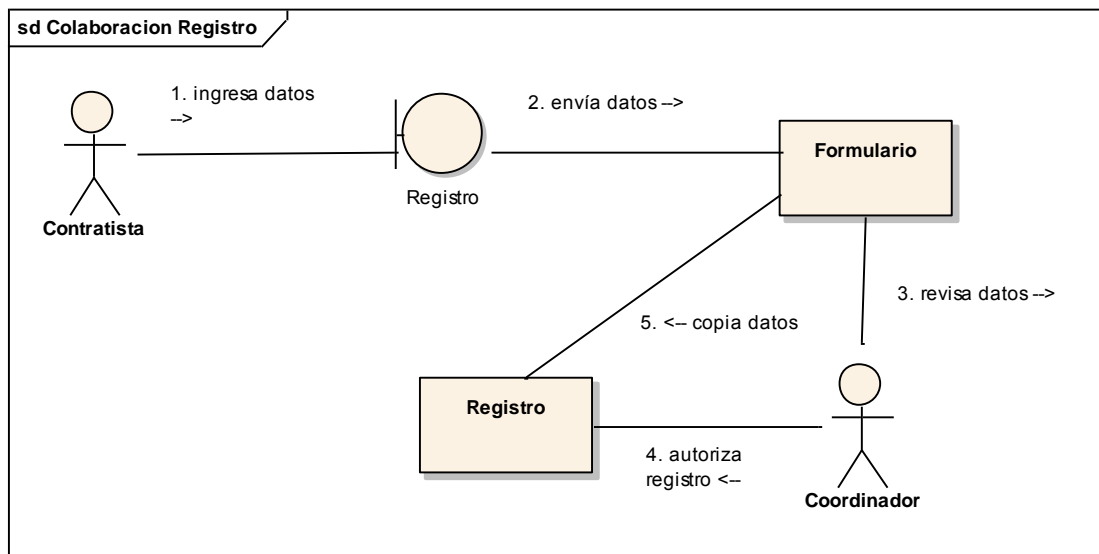
**Figura 13. Diagrama de secuencia de mensajes entre un usuario y una aplicación web**



### 1.1.2.5. Diagrama de colaboración

El diagrama de colaboración es muy importante en el diseño de arquitectura de un sistema de *software*, ya que muestra la interacción entre los objetos durante la ejecución de la aplicación. La interacción entre los objetos puede darse de dos formas diferentes: estática y dinámica. La parte estática está definida por los enlaces entre los objetos, es decir, el modo de conexión o canal de comunicación entre ellos. La parte dinámica está definida por los mensajes que se envían a través de dichos enlaces ya formados previamente entre los objetos. Dichos mensajes pueden transmitirse de varias formas: múltiple, única o bien condicionada.

Figura 14. Diagrama de colaboración





## 1.2. Acerca del ciclo de vida del *software*

Todo proyecto de ingeniería tiene fines ligados a la obtención de un producto, proceso o servicio de *software* que satisfaga las necesidades de usuarios finales a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, el cual es necesario para continuar hacia el producto final y facilitar la gestión del proyecto de *software*. Sin embargo, la forma de agrupar las actividades, los objetivos de cada fase, los tipos de productos intermedios que se generan, etc. pueden ser muy diferentes dependiendo del tipo de producto o proceso a generar y de las tecnologías empleadas.

La división de los proyectos en fases sucesivas es un primer paso para la reducción de su complejidad, debido a que se escoge las partes más sencillas de manera que sus relaciones entre sí sean lo más simples posibles. Al conjunto de fases en las que se divide un proyecto de ingeniería, se le conoce como “El Ciclo de Vida” [1].

La definición de un ciclo de vida, facilita el control sobre los tiempos en que es necesario aplicar recursos al proyecto, tales como: personal, equipos, suministros, etc. Al, al tener una separación en fases del ciclo de vida en puntos clave, se garantiza en gran medida un nivel superior de control de calidad del proyecto.

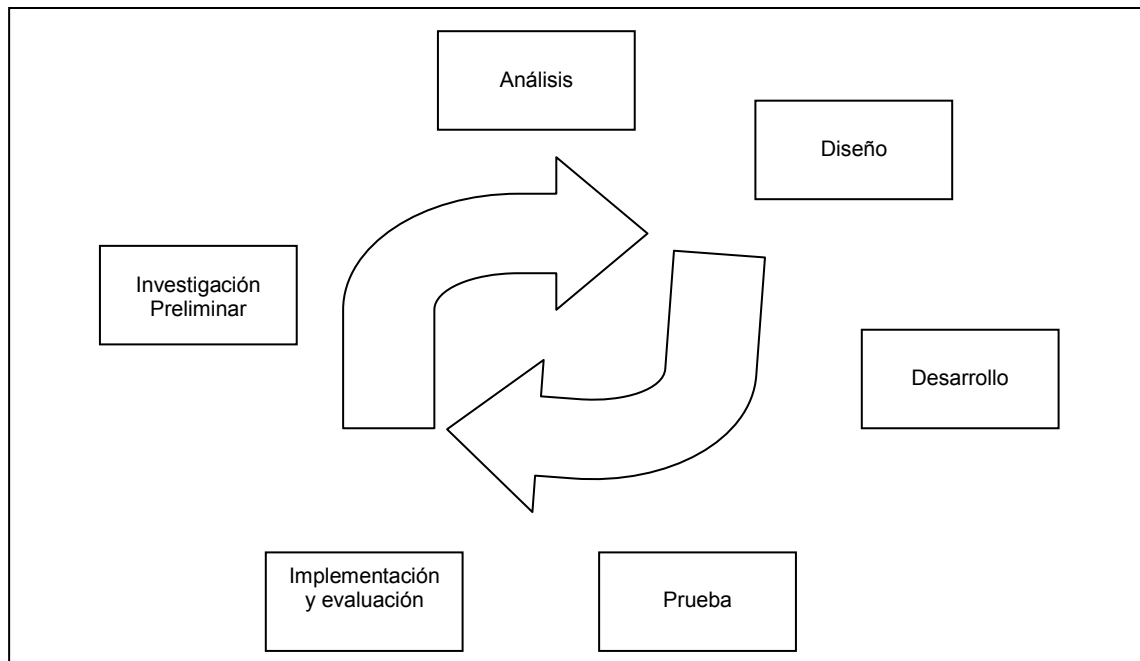
En la medida que se emplea el ciclo de vida del *software* para la creación de sistemas de aplicación, se adquiere experiencia útil para proyectos de *software* futuros, que conllevarán a resultados eficientes que responden a las necesidades del sistema inicial.

En la mayoría de los procedimientos empresariales las actividades están relacionadas entre sí, a tal grado que son inseparables, y se complica en gran medida la distinción de la secuencia de las mismas. Así mismo, las partes que componen el proyecto pueden encontrarse al mismo tiempo en distintas fases de desarrollo, algunos componentes en la fase de análisis mientras que otros en etapas avanzadas de diseño.

El ciclo de vida del desarrollo de sistemas de *software* consta de las siguientes actividades:

- Investigación preliminar
- Análisis
- Diseño del sistema
- Desarrollo del *software*
- Prueba de los sistemas
- Implantación y evaluación

**Figura 15. Desarrollo del *software***



### **1.3. Investigación preliminar**

La necesidad para recibir ayuda de un sistema de información puede originarse por diversas razones, es decir que el proceso se inicia siempre con la petición de una persona (administrador, empleado o especialista en sistemas, etc.).

Cuando la solicitud es realizada comienza la primera etapa del ciclo de vida, la cual es la investigación preliminar [2]. Se puede decir que esta etapa está dividida en varias partes, las cuales son: la identificación del problema, un estudio de factibilidad y por último se puede mencionar la aprobación de la solicitud [2].

Es en ésta etapa, en la que se realiza una lista de posibles soluciones que son conocidas como sistemas candidatos [4], y de estas se elige una, la cual se procederá a darle seguimiento durante las restantes fases del ciclo de vida, y a esta solución se le conoce como sistema propuesto [4].

Uno de los primeros pasos de esta fase del ciclo de vida es la aclaración de la solicitud, ya que muchas solicitudes que provienen de empleados y usuarios no están realizadas de forma clara, y antes de considerar cualquier investigación del sistema, la solicitud debe de examinarse para determinar con mayor precisión lo que el usuario desea.

Muchos casos, llevan a que los usuarios que realizan la solicitud del sistema, tienen una buena idea de lo que necesitan pero no están seguros de cómo explicarlo o transmitirlo para que los que recopilan los requerimientos tengan una idea clara de lo que el usuario solicita.

El diseño del sistema, se vuelve muy difícil cuando los encargados que realizan las solicitudes no están completamente claros de lo que desean, o bien no saben que es lo que está mal o en donde se encuentra el problema, entonces, antes de continuar con cualquier fase del ciclo de vida, la solicitud del proyecto debe de estar claramente planteada, al igual que las dos partes que están involucradas deben estar conformes, es decir, que los usuarios del sistema que solicita el *software* estén de acuerdo con lo obtenido por la persona o personas encargadas de recolectar los requerimientos

Luego de tener clara la solicitud del sistema, se procede a realizar un estudio de factibilidad, el cual se utiliza para determinar que el sistema solicitado sea factible de realizar, es decir, que sea apropiado o no implementarlo tomando en cuenta una serie de aspectos como lo son: técnicos, económicos, legales y operacionales [2].

### **1.3.1. Factibilidad técnica**

El análisis de factibilidad técnica evalúa si el equipo y *software* están disponibles. Sucede lo mismo en el caso del *software* si es posible desarrollarse y si tienen las capacidades técnicas requeridas por cada alternativa del diseño que se esté considerando. Los estudios de factibilidad técnica también consideran las interfaces entre los sistemas actuales y el sistema nuevo, ya que los componentes que tienen diferentes especificaciones no pueden interconectarse, y los programas de *software* no pueden pasar datos a otros programas si tienen diferentes formatos en los datos o sistemas de codificación, porque tales componentes y programas no son compatibles técnicamente. Además, los estudios de factibilidad técnica también consideran si la organización tiene el personal que posee la experiencia técnica requerida para diseñar, implementar, operar y mantener el sistema propuesto. Si el personal no

tiene dicha experiencia, puede instruírsele, o pueden emplearse a nuevos consultores que la tengan. Sin embargo, una falta de experiencia técnica dentro de la organización puede llevar al rechazo de una alternativa particular.

### **1.3.2. Factibilidad económica**

Los estudios de factibilidad económica incluyen análisis de costos y beneficios asociados con cada fase o punto del proyecto. En un análisis de costo/beneficio, todos los costos y beneficios de adquirir y operar cada sistema alternativo se identifican y se hace una comparación de ellos.

Los costos de implementación incluyen comúnmente el costo remanente de la investigación de sistemas, de *hardware* y *software*, los de operación del sistema para su vida útil esperada y los costos de mano de obra, material, energía, reparaciones y mantenimiento. A través del análisis de costo/beneficio, la organización debe utilizar los conceptos tradicionales de análisis financiero y herramientas como: análisis de costos diferenciales y análisis de flujos descontados [3].

### **1.3.3. Factibilidad operacional**

Este estudio comprende una determinación de la probabilidad de que un nuevo sistema se use como en realidad debe usarse. Se deben considerar cuatro aspectos de la factibilidad: Primero, ¿Un nuevo sistema puede ser demasiado complejo para los usuarios de la organización o los operadores del sistema?, Si lo fuese, los usuarios pueden ignorar el sistema o bien usarlo en tal forma que cause errores o fallas en el sistema, lo cual no representa ninguna ayuda al desarrollo de las actividades del proceso actual. Segundo, un sistema puede hacer que los usuarios se resistan a él, como consecuencia de una

técnica de trabajo, miedo a ser desplazados, intereses en el sistema antiguo u otras razones. Para cada alternativa debe explorarse con cuidado la posibilidad de resistirse al cambio al nuevo sistema. Tercero, un nuevo sistema puede introducir cambios demasiado rápido para permitir al personal adaptarse a él y aceptarlo, sin importar qué tan atractivo pueda ser un sistema en su aspecto económico, si la factibilidad operacional indica que es muy probable que los usuarios no acepten el sistema o que el uso del mismo resultará incurrir en muchos errores o en una baja en la motivación personal, el sistema no debe implantarse.

#### **1.3.4. Factibilidad legal**

En este estudio se analizan todos los documentos legales que puedan estar inmersos en la realización del sistema, como documentos legales se puede hacer mención de contratos que se tengan que realizar, licencias que se deban adquirir para un uso legal de algún *software*, o bien porque es probable que se realice el sistema en algún país distinto al país que realiza la solicitud, y esto conlleva a regirse bajo los marcos legales locales.

El estudio de factibilidad legal lo lleva a cabo regularmente un equipo pequeño de personas, en ocasiones una o dos, las cuales están familiarizadas con las distintas facetas legales de los sistemas de información. Este equipo esta formado por personal experto en los procesos de análisis y diseño de sistemas, regularmente analistas capacitados o bien directivos.

Luego que se realiza un estudio de factibilidad del sistema se procede a la Aprobación de la Solicitud. Puesto que no todos los proyectos solicitados son deseable o factibles. Esto hace referencia, a que se ha analizado en esta fase

del ciclo de vida que el proyecto es factible y es posible realizarlo con los puntos que se mencionan en la solicitud.

#### **1.4. Análisis**

Un aspecto fundamental es comprender todas las facetas importantes de la parte de la empresa que se encuentra bajo estudio, es decir comprender el funcionamiento de la empresa que realiza la solicitud del *software*. Los analistas cuando trabajan con los empleados y administradores de las empresas deben de estudiar el funcionamiento de la empresa, esto con el fin de poder responder a las siguientes preguntas [1]:

- ¿Qué es lo que se hace?
- ¿Cómo se hace?
- ¿Con que frecuencia se presenta?
- ¿Qué tan grande es el volumen de transacciones o de decisiones?
- ¿Cuál es el grado de eficiencia con el que se efectúan las tareas?
- ¿Existe algún problema?
- Si existe un problema, ¿Qué tan serio es?
- Si existe un problema, ¿Cuál es la causa?

Para poder contestar estas preguntas, el analista debe de conversar con varias personas para reunir detalles relacionados con los procesos de la empresa, así como sus opiniones, y las soluciones que proponen. Para esto se puede hacer uso de cuestionarios o entrevistas, esto implica hablar con las personas adecuadas, es decir, con las personas que saben exactamente cuales son los requerimientos, o bien el funcionamiento interno de la empresa, esto es para no recolectar malos requerimientos, o bien no comprender realmente la necesidad de la empresa.

## 1.5. Diseño

En esta etapa del ciclo de vida del *software*, se realiza y se estructura todos los componentes que formaran el sistema final, así como la forma en la que el sistema cumplirá con los requerimientos identificados en la fase de análisis.

Un buen punto de partida para esta fase, es identificar los reportes y demás salidas que debe producir el sistema. Una vez que se ha hecho esto, se puede identificar con precisión los datos específicos para cada reporte y salida. En este punto se puede realizar algún bosquejo del formato o de la pantalla que se mostrara.

En esta etapa también se debe de considerar los datos de entrada, los datos que serán calculados así como también los datos que serán almacenados en el sistema. También es en esta fase en donde se selecciona los dispositivos de almacenamiento, tales como discos, o dispositivos de rendimiento como procesadores, memoria, etc., que serán utilizados por el sistema. Así mismo en esta etapa la estructura o diseño del sistema propuesto es finalizada, esto incluye los archivos, bases de Datos, entradas, salidas, así como también las pantallas que formaran el sistema.

Se realiza la arquitectura completa del sistema, la cual se compone de serie de documentos en los cuales se representan las especificaciones en las cuales se basará el sistema para su implementación, se pueden mencionar diagramas, tablas, símbolos, etc. Toda esta documentación se realiza con el fin de proporcionarla al equipo de implementación para que este desarrolle el sistema bajo normas definidas y una lógica meticulosamente diseñada.



## **1.6. Desarrollo del *software***

En esta etapa del ciclo de vida, es en donde se procede a crear todo lo que se ha especificado en la etapa de diseño. También puede llamarse Codificación, puesto que es aquí en donde se realiza toda la traducción a código de un lenguaje específico para crear y desarrollar el sistema propuesto.

En esta fase no solo se realiza la codificación en si del sistema, si no que también es en donde se realiza toda la documentación que el mismo pueda necesitar. Con esto nos referimos a los manuales, ya sea de instalación, de cómo utilizar el sistema, así como de los manuales en los cuales se puede describir como esta realizado el sistema. Toda esta documentación es necesaria para probar el sistema, así como también para llevar a cabo el mantenimiento una vez que el sistema ha sido instalado.

## **1.7. Pruebas**

Durante esta fase de pruebas, el sistema se emplea de manera que se asegure que el *software* no tenga fallas, es decir que funcione de acuerdo con las especificaciones y la forma en que los usuarios esperan que lo haga. Para realizar esto se introducen los datos de prueba al sistema, estos datos no son mas que un conjunto de información ya sea verídica o no, con el fin de comprobar el correcto funcionamiento del mismo. El funcionamiento del sistema se hace, comprobando las salidas que muestra, y comparándolas con las salidas que se espera que el sistema despliegue.

Esta fase es muy importante dentro del ciclo de vida, puesto que es mejor descubrir las fallas antes de que el sistema se implemente, puesto que si son descubiertas con el sistema ya en funcionamiento, esto puede llevar a un gasto

mayor de recursos etc. Estas pruebas pueden ser realizadas ya sea por personas ajenas a los programadores, como por ejemplo, un grupo de usuarios que serán los que manejen el sistema, con el fin de asegurar un mayor funcionamiento del sistema, puesto que los usuarios de prueba son realmente los usuarios finales del mismo.

### **1.8. Implantación y evaluación**

La implantación es el proceso en el cual se hace la verificación e instalación del nuevo equipo, así como también es en donde se capacitan a los usuarios, se instala la aplicación en sí y configurar todos los archivos de datos que sean necesarios para que el sistema funcione.

Durante esta fase, si el sistema es muy grande, es probable que se implante solamente en algún área de la empresa, con el fin de comprobar el funcionamiento del mismo, para luego proceder a instalarlo en todas las aéreas que sean necesarias, a esto se le puede llamar Prueba Piloto [3].

Esta fase no consiste solamente en implantar el sistema en el negocio, si no que también implica toda la asesoría que pueda venir con el paso del tiempo, debido que un sistema posee una vida útil por lo regular grande. En esta asesoría es cuando se realizan los cambios y modificaciones que el sistema necesite, así como para implementar nuevos requerimientos que posea la empresa, es por esto que esta fase es un proceso en constante evolución.

La evaluación de un sistema se lleva a cabo para poder encontrar y definir los puntos fuertes y débiles del sistema. Esta se puede dar de la siguiente forma:

- Operacional: Es cuando se valora como funciona el sistema, incluyendo aquí la facilidad de uso del mismo, el tiempo de respuesta, la confiabilidad global y el nivel de utilización.
- Organizacional: Se identifican los beneficios que pueda aportar a la organización en las distintas aéreas de la empresa.
- Opinión de los organizadores: Se evalúa la actitud de los directivos y de los administradores hacia el sistema, así como de los usuarios finales del mismo.
- Desempeño del desarrollo: Se realiza un análisis del sistema, para comprobar si concuerdan con los estándares establecidos al inicio del ciclo de vida.[3]

En la actualidad la evaluación de los sistemas no siempre recibe la atención que se merece. Sin embargo cuando se realiza adecuadamente, puede aportar mucha información que pueda ayudar a mejorar la efectividad de los sistemas que se realicen subsecuentemente.

## **2. DISEÑO DE GAMEPLAY**

El juego es una simulación en tiempo real de una batalla en un campo, con un grupo de tropas que incluyen artillería, infantería y municiones extra. La batalla se dará entre dos contrincantes humanos que están en distintas computadoras. El objetivo es cumplir con las misiones que se dan en la orden de operaciones al inicio de cada escenario utilizando la mejor estrategia posible.

### **2.1 Juegos de simulación [10]**

Un juego de simulación, permite crear una situación tal, que, mediante la estructuración de un escenario real o ficticio, se observe el comportamiento y los resultados de la aplicación de decisiones. Esto, frente a otros actores que son parte del juego, y que manifiestan sus intereses a través de otras acciones. El escenario está construido sobre un mapa o carta geográfica que representa el teatro de acciones. Estos juegos o ejercicios de simulación militar, tienen como objetivo poner a los oficiales que se están entrenando en situaciones que se ajusten lo más posible a una realidad bélica, añadiendo a ello el empleo de sistemas informáticos que faciliten el entrenamiento.

Los juegos de guerra que se utilizan actualmente en los ejércitos occidentales son copias de los juegos originales que se utilizaban en Europa, los cuales a su vez tienen su origen en el oriente, siendo utilizados por China y ejércitos árabes, los cuales adaptaron el ajedrez a estas técnicas de entrenamiento militar.

Los juegos de simulación, permiten mediante trabajos aplicados, tener la oportunidad de practicar y adquirir experiencia en acciones de guerra a Comandantes, integrantes del Estado Mayor, Comandantes Subalternos o Unidades Menores. Los juegos de simulación se catalogan como trabajos aplicados sin tropas.

Tras la Segunda Guerra Mundial, el uso militar de los juegos de guerra se ha ido extendiendo, sofisticando y tecnificando. El aumento de la capacidad de procesamiento de las computadoras ha traído a su vez un aumento del realismo de las simulaciones que se realizan en las mismas, lo que ha hecho que muchas empresas se dediquen a comercializar juegos que simulan estrategia militar; sin embargo, pocos o ninguno de estos juegos se adaptan a las necesidades que requiere un ejército en sus prácticas, debido sobre todo a características propias de los juegos, que modifican la realidad para dar un mayor nivel de entretenimiento. Esta situación ha motivado que los ejércitos se dediquen a crear sus propios juegos de simulación, en los que las características de las unidades y del terreno sean lo más apegadas a la realidad posible.

## **2.2 Mecánica general de los juegos de simulación [10]**

### **2.2.1 Estructura**

Los juegos de simulación se estructuran en dos fases principales para su desarrollo, en cualquiera de los niveles de la conducción, una primera fase de preparación y posteriormente una fase de ejecución.

Específicamente en el caso de un juego de simulación de manejo de crisis, aspecto puntual que plantea el tema de este trabajo de investigación, los niveles de la conducción en los cuales se encuadra este sistema de entrenamiento son:

- Nivel político
- Nivel político-estratégico

En la fase de preparación, la dirección general del juego, representada por los docentes, planifica y establece el marco teórico de éste, de tal manera

que, los alumnos que constituyen la dirección del juego, tengan el punto de partida para la conformación del legajo del juego. Este legajo en particular, contendrá la documentación pertinente que permita materializar la ficción de éste y los aspectos puntuales para cada uno de los equipos que se conformen.

el marco teórico deberá contener el enunciado del juego de simulación, en un documento que debe sintetizar todos aquellos antecedentes que permitan a los respectivos Directores, orientar el planteamiento y desarrollo del Juego mismo, y es entregado por la dirección general del juego.

Los aspectos a considerar en el enunciado son:

- a) Tema: Se refiere al tipo de Ejercicio sobre el cual se desarrollará el juego.
- b) Fecha: Se refiere a la duración del Juego y a su ubicación en el calendario.
- c) Lugar: Se refiere al espacio físico donde se llevará a cabo el juego.
- d) Nivel: Se refiere al nivel de la conducción, en el cual se enmarca el juego.
- e) Alcance: Se refiere a la acción que tendrá el juego, pudiendo este ser de acción simple o de doble acción.
- f) Elemento de Trabajo: Se refiere al conjunto que operacionalizará el juego como unidad de ejecución.
- g) Participantes: Se refiere al equipo humano que conformaran los tres elementos que dan vida al juego, y que son: La dirección, los equipos y el equipo de apoyo.
- h) Región de trabajo: Se refiere a la zona geográfica, en la cual se desarrollará la acción.
- i) Finalidad: Se refiere al objetivo primario del Juego orientado básicamente al propósito de este y al logro que se desea alcanzar.

## **2.3 Gameplay**

El juego consiste en escenarios que se juegan sobre un mapa. Se deben tener en cuenta 3 áreas:

- Órdenes del usuario: Órdenes de movimiento, fuego, o sobre el comportamiento de las unidades.
- Interacciones con el terreno: En qué áreas del mapa pueden pasar las unidades, que unidades pueden abrirse camino, que pasa cuando una unidad llega a un lugar donde no puede avanzar más.
- Eventos generados por el contrincante: Fuego enemigo, visibilidad de unidades contrincantes.

### **2.3.1 Controles**

El juego será controlado principalmente con el *mouse*, seleccionando las opciones para las unidades en un menú contextual y otras opciones desde un menú principal o bien una barra de opciones.

### **2.3.2 Objetivo del juego**

El objetivo del juego es completar las misiones en un escenario determinado. Estas pueden ser:

- Ocupar un área determinada.
- Eliminar las unidades enemigas en un área determinada.
- Eliminar un cierto porcentaje de las unidades enemigas.
- Defender un área.
- Mantener un porcentaje de las unidades propias.
- Una barrera de tiempo durante la cual debe cumplirse cualquiera de los objetivos.

En un escenario puede haber una combinación de varias misiones para obtener la victoria. Al finalizar un escenario se presentara un resumen de todas las acciones y las posiciones de todas las unidades al final. Se considera guardar un record de cada acción para poder mostrar todos los pasos que se hicieron durante el juego.

Si el objetivo total ya no puede cumplirse debido a que alguna de las misiones falla, se mostrara en pantalla un aviso. También se mostrará un aviso cuando se cumpla con alguna de las misiones. El juego concluirá automáticamente si alguno de los contrincantes cumple con todas sus misiones, si alguno ya no puede realizar ninguna de sus misiones o bien si alguno se rinde desde el menú de opciones.

## **2.4 Ambiente del juego**

### **2.4.1 El mapa**

El mapa de juego es un mapa de elevaciones sobre el que se seleccionaran las áreas en que se va a desarrollar el escenario. Se planea desarrollar una herramienta que permita la creación de escenarios de juego.

En el mapa pueden encontrarse obstáculos naturales que eviten el transitar de las unidades o al menos lo hagan más lento. Puede haber distintos tipos de terreno, por ejemplo bosque, terreno limpio, etc.

### **2.4.2 Unidades**

Se creara una biblioteca de unidades (a la que se pueden agregar nuevas unidades personalizadas), en la que se encontraran separadas según su tipo de unidad. Estos tipos serán: Infantería, tanques, artillería, transporte.



Cada unidad cuenta con armas que le permitan entrar en combate. Cada unidad tiene una velocidad promedio, que puede variar según las condiciones del terreno en que se encuentre. Cada unidad tiene un rango de alcance que puede variar según el tipo de arma, sobre todo para las unidades de artillería.

### **2.4.3 Unidades de infantería**

Las unidades de infantería se encuentran por pelotones, y pueden llevar armas pequeñas y lanzamisiles como máximo. Son muy lentas, pero pueden esconderse en trincheras y ser transportadas por unidades con capacidad de transporte. Tiene un rango corto. Pueden pasar casi por cualquier tipo de terreno.

### **2.4.4 Unidades de tanques**

Los tanques cuentan con armas de alto calibre y pueden llevar armas de un calibre más pequeño para defenderse de infantería a corta distancia. Tienen una buena velocidad pero son muy vulnerables al terreno en que se desplacen. También hay unidades scout que son enviadas a investigar, estas son muy rápidas y llevan poco armamento.

### **2.4.5 Unidades de artillería**

Las unidades de artillería tienen velocidad media, pueden desplazarse por terrenos similares a por los que pueden desplazarse los tanques. Cuentan habitualmente con distintos tipos de municiones, ya sean bombas, minas, humo para cubrir movimientos, misiles guiados por láser o iluminación.

### **2.4.6 Otras unidades**

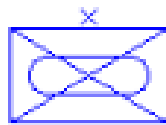
Existen otras unidades, por ejemplo los breaches que pueden abrir paso a

través de obstáculos para que pasen otras unidades. Además, se puede evaluar agregar unidades que están en la versión completa del juego (según las descripciones) como por ejemplo unidades aéreas o marítimas.

#### **2.4.7 Identificación de las unidades**

Las unidades se identificarán utilizando la notación de símbolos estándar creada por la OTAN (Organización del Tratado Atlántico Norte), ya que son los símbolos utilizados por la gran mayoría de países occidentales.

Estos símbolos son representados habitualmente por un cuadrado que encierra una imagen que representa el tipo de unidad que representa el cuadrado, así como su equipamiento (armamento), y en cuya parte superior se encuentra otro símbolo que representa el tipo de agrupación (regimiento, brigada, etc.), de la siguiente forma:



En este caso, la X significa que el regimiento es una brigada, y la imagen dentro del cuadro representa infantería mecanizada, una combinación de infantería y tanques.

#### **2.5 Simbología**

Existen dos clases de simbología a utilizar, los símbolos que representaran las agrupaciones y los símbolos que representaran el equipamiento, los cuales se describen a continuación:

### 2.5.1 Símbolos que representan las agrupaciones

A continuación se presenta un listado de los símbolos que representan cada tipo de agrupación, los cuales pueden ser ubicados en la parte superior de cada uno de los cuadros.

**Tabla i. Símbolos que representan las agrupaciones**

<b>Símbolo</b>	<b>Nombre</b>	<b>No. Personas</b>	<b>Unidades subordinadas</b>
XXXXXX	Teatro de operaciones	> 200,000	2+ grupos de ejército
XXXXX	Grupo de ejército	> 100,000	2+ ejércitos
XXXX	Ejército o frente	50,000 – 60,000	2+ cuerpos de ejército
XXX	Cuerpo de ejército	30,000	2+ divisiones
XX	División	10,000 – 20,000	2-4 brigadas
X	Brigada	2,000 – 5,000	2+ regimientos
III	Regimiento	2,000 – 3,000	3 - 4 batallones
II	Batallón	300 – 1,000	2 – 6 compañías
I	Compañía	60 – 250	2 – 6 pelotones
...	Pelotón	30 – 40	2+ patrullas
..	Patrulla	8 – 12	2+ equipos de fuego
.	Tripulación	8 – 12	2+ equipos de fuego
∅	Equipo de fuego	4 - 5	---

## 2.5.2 Símbolos que representan equipamiento

A continuación un listado con algunos de los principales símbolos que se utilizarán en el juego:

**Tabla II: Símbolos que representan equipamiento**

Símbolo	Descripción
	Defensa antiaérea
	Unidades anfibas
	Anti-tanques
	Tanques
	Tanques y unidades de reconocimiento
	Tanques + infantería, infantería mecanizada
	Aviación
	Artillería básica
	Infantería
	Reconocimiento, caballería
	Provisionamiento
	Lanza morteros
<b>Los siguientes símbolos se combinan con otros</b>	
	Morteros
	Artillería antiaérea
	Ametralladora antitanque

### **2.5.2.1 Color de las unidades**

Las unidades Azules se reconocerán como unidades aliadas, mientras que las unidades Rojas se reconocerán como unidades enemigas. Esto también de acuerdo a lo universalmente aceptado por los ejércitos occidentales.

### **2.5.2.2 Unidades al inicio del juego**

Según lo que hayan decidido en la fase estratégica los estudiantes del curso superior, se darán tropas a los participantes del juego según la orden de operaciones establecida.

### **2.5.2.3 Orden de operaciones**

La orden de operaciones incluye la estrategia que deben seguir los equipos contrincantes y las unidades que cada equipo tendrá. Al inicio del juego se mostrará la orden de operaciones detallada.

## **2.6 Escenarios**

La página de selección de escenarios permitirá buscar escenarios creados previamente en cualquier parte del disco duro, por medio de un cuadro de diálogo.

Los escenarios serán creados utilizando una herramienta que se desarrollará exclusivamente para esto, en la que se podrán cargar los mapas y definir las áreas con la misma facilidad que se haría en cualquier herramienta de dibujo básico (ejemplo, GIMP, Microsoft Paint).

Tal y como se muestra en el diagrama de actividades que esta anteriormente, si ya existe algún juego creado, se utilizará una pantalla igual a la de selección de escenarios para elegir a cual juego se va a unir (máximo 2 equipos activos por partida, pueden ingresar equipos observadores).

Al iniciarse la partida, cada jugador verá en azul sus unidades, y áreas del terreno marcadas en amarillo y en rojo. Las áreas marcadas en amarillo delimitan sus áreas de maniobra, y estarán detalladas en la orden de operaciones. Las áreas en rojo demarcan la posible área de maniobra del equipo contrario.

Los objetivos a conseguir también estarán dentro de la orden de operaciones. A la orden se puede acceder en cualquier momento desde el menú contextual del juego.

El menú contextual al iniciar el juego contiene las siguientes opciones:

- Mostrar orden de operaciones
- Información inteligencia
- Información unidades (si se da clic sobre una unidad)
- Listo para iniciar
- Salir

El primer paso dentro del desarrollo de la partida es posicionar las unidades que se tienen dentro del mapa. Para esto se arrastran utilizando un *Drag & Drop* a las posiciones en que se desean colocar. Estas posiciones iniciales estarán delimitadas por una de las áreas en amarillo.

Las unidades a utilizarse tendrán sus respectivos nombres y por medio del símbolo permitirán identificar el número de tropas y el tipo de unidad que és.

Las áreas asimismo, pueden tener nombres asignados, para facilitar su identificación en el desarrollo de las acciones.

### **3. ESPECIFICACIONES SUPLEMENTARIAS**

El presente documento contiene la especificación de los requerimientos que se tienen del sistema a implementar, pero que no son funcionales, tales como la disponibilidad, el rendimiento y la usabilidad del sistema.

Se debe tener especial consideración en requerimientos tales como la seguridad, la usabilidad y la variabilidad del sistema.

#### **3.1 Propósitos**

El propósito de este documento es la presentación de aquellos requerimientos que no son parte funcional del sistema, pero que son de soporte y gran importancia para el desarrollo de este. El documento contiene dichos requerimientos, pero pone un especial énfasis en la eficiencia, eficacia y seguridad.

#### **3.2 Alcances**

Este documento cubre aquellos requisitos no funcionales del sistema, tales como la confiabilidad, usabilidad, eficiencia, rendimiento y su completa descripción. Principalmente lo que se desea alcanzar con este documento es marcar las reglas que el sistema debe seguir para tener un alto valor de calidad.

#### **3.3 Definiciones y abreviaciones**

Las definiciones y abreviaciones a utilizar en el presente documento son:

- Modelo TCP/IP: modelo de capas utilizado en la Internet para la comunicación de datos entre computadoras.



- *Firewall*: mecanismo de control de información que se agrega a la red de un sistema para controlar la información que entra y sale de la red.
- Firma Digital: mecanismo de seguridad y encriptación de datos que brinda autenticidad de los clientes y de las empresas, y asegurándoles que los datos que envían por la red serán leídos única y exclusivamente por el receptor.
- HTTP: El Protocolo de Transferencia de HiperTexto (*Hypertext Transfer Protocol*) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores http.
- HTTPS: Versión segura del protocolo HTTP.

### **3.4 Seguridad**

Por el tipo de aplicación que se está construyendo, que viene a ser un programa cliente que se conectará a un servidor en el que se encuentra la información principal sobre las maniobras que se pueden realizar, existen varios ataques que pueden comprometer la seguridad de la información.

Por otro lado, la información contenida no requiere de una confidencialidad demasiado estricta, con lo que las medidas a tomarse para evitar que la información se fugue no deben ser demasiadas sin por eso dejar desprotegido el sistema.

Es por eso que es necesario tomar medidas administrativas para poder manejar de manera segura la información contenida en el sistema, verificando así que esta se mantenga consistente y completa, y que además pueda ser vista solamente por las personas a las que les está permitido hacerlo.

### **3.4.1 *Firewalls de hardware***

Los *firewalls* de *hardware* proveen un alto nivel de rendimiento ya que encapsulan en el mismo gabinete tanto el *software* como el *hardware*. Una herramienta de este tipo puede ayudarnos mucho a mantener la seguridad al proteger totalmente todas nuestras conexiones evitando así cualquier conexión sospechosa, indebida o totalmente peligrosa.

### **3.4.2 *Firewalls de software***

Los *firewall* de *software* servirán para aumentar el nivel de seguridad, principalmente en el servidor principal, ya que al ser una aplicación que podrá ser accedida de manera remota, quizás incluso a través del *Web*, serán pocas las máquinas clientes definidas dentro de la empresa en comparación con la cantidad de clientes que podrán acceder a nuestros servicios de manera remota.

## **3.5 Usabilidad**

La usabilidad representa la facilidad de uso y aprendizaje por parte de los usuarios del sistema, sin necesidad de una guía detallada de la forma de uso del mismo. Se detalla la administración del diseño para que la interacción entre el sistema y los usuarios sea intuitiva y sencilla.

### **3.5.1 Facilidad de uso**

Se proveerá de una interfaz sencilla y fácil de uso, lo más intuitiva posible, en la que la mayoría de las interacciones sean posibles únicamente utilizando movimientos y clics del ratón.

### **3.5.2 Desarrollo de tareas**

Se realizará un seguimiento de las tareas a realizar durante el desarrollo de un ejercicio de comando, así como de quién fue el encargado de realizarlas.

### **3.5.3 Ayuda para el usuario**

Se brindará ayuda desde el menú contextual (accesible por medio del clic derecho), sobre el objeto que este siendo apuntado por el ratón al momento de hacer clic. Además de la sección de ayuda presente en cada página, se debe encontrar un área en la que se incluya ayuda más general, tal como un FAQ (Frequently Asked Questions) para problemas que se den recurrentemente para los usuarios, y descripciones detalladas de las funciones más importantes del sistema.

## **3.6 Confiabilidad**

El sistema deberá estar levantado todo el tiempo que sea necesario para realizar la práctica de los cursos en que sea necesario utilizarlo. La conexión entre las aplicaciones cliente y la información contenida en el Servidor no debe perderse y debe tener una latencia baja para evitar problemas, sobre todo dado que se tiene que lograr una experiencia en tiempo real para todas las partes involucradas en el uso del sistema.

### **3.6.1 Disponibilidad**

El sistema debe estar levantado la mayor parte del tiempo posible, por lo que se debería de tener un servidor de respaldo para cada otro servidor que exista en el sistema, de modo que si alguno falla, el secundario entre a trabajar evitando la falta de servicio.

### **3.6.2 Disponibilidad de información**

La disponibilidad de información se trabaja en el ámbito de la base de datos. Por tanto, se tienen dos modalidades las cuales brindan el mayor nivel de seguridad para la información almacenada. Estas modalidades son:

#### **3.6.2.1 Base de datos en paralelo**

El esquema a utilizar será el tener varios nodos de información, con conexiones redundantes que permiten que la carga se comparta entre todos, mejorando el tiempo de respuesta y por tanto el rendimiento total del sistema. Esto queda sujeto al uso que se le dé al sistema y a la capacidad del manejador de base de datos para soportar esta característica.

#### **3.6.2.2 Recuperación de desastres**

Se basa en tener un respaldo de información pero en una locación distinta al lugar en que se encuentran los servidores principales, de manera que si algún desastre sucediera, la información no se pierde, al existir una copia de seguridad.

### **3.6.3 Precisión**

El sistema tendrá una precisión del 99% para así poder brindar a los usuarios la mejor confianza de respuesta y así poder trabajar de manera regular, sin interrupciones provocando así una comodidad y confianza en los servicios prestados por el sistema.

### **3.6.4 Requerimiento de confiabilidad**

Para poder mantener la experiencia de tiempo real, las conexiones de los

lados clientes deberán de ser conexiones que contengan al menos 756 kbps dedicados y totalmente disponibles para el sistema.

### **3.7 Desempeño**

El desempeño es la medida del tiempo de respuesta del sistema para un requerimiento funcional. Para un sistema de este tipo se requiere un alto desempeño por la cantidad de usuarios conectados a la vez y el tipo de operaciones a realizar.

#### **3.7.1 Velocidad**

Los contenidos incluidos en el sistema, específicamente en el servidor pueden verse bastante cargado, sobre todo en lo que al acceso constante a la base de datos se refiera, así como también a los cálculos que se hagan sobre el mapa.

Se requiere que el servidor tenga como mínimo 3 GB de memoria RAM y un procesador de aproximadamente 3 Ghz, con doble núcleo para garantizar poder procesar las instrucciones provenientes de los clientes en el menor tiempo posible. Las máquinas cliente deben poseer al menos 1 GB de memoria RAM, un procesador de 2.4 Ghz o superior, y una tarjeta gráfica con capacidad de aceleración de 128 Mb para el proceso de renderización de la imagen.

#### **3.7.2 Capacidad**

El servidor podrá manejar la información e instrucciones provenientes para el juego para 2 máquinas clientes que interactúen en el juego, permitiendo asimismo conexiones de otros clientes que actúen únicamente como evaluadores, y que por tanto no realicen envío de instrucciones durante el juego directamente al servidor. Nótese que entre más clientes se conecten al servidor,

independientemente de si se encuentran como observadores o como contendientes, la calidad de la conexión se deteriorará, por lo que es recomendable tener un ancho de banda dedicado de al menos 512 Kbps en el servidor por cada conexión entrante.

### **3.7.3 Tiempo de respuesta**

El tiempo entre que se ingresa una instrucción en un cliente y que esta se ejecuta no debería superar en el peor de los casos los 2 segundos, y debe mantenerse en un promedio inferior a 1 segundo.

### **3.7.4 Tiempo medio de recuperación**

Para mantener la confianza de los usuarios, precisamente los externos a la empresa, el sistema debe poder recuperarse de fallos en un tiempo promedio de 20 minutos.

## **3.8 Soportabilidad**

La soportabilidad es la capacidad del sistema de soportar nuevas modificaciones y cambios. Es conocida también como Variabilidad o la capacidad del sistema de ser modificable.

### **3.8.1 Adaptabilidad**

El sistema podrá ser portado a otros servidores, siempre y cuando cuenten con características similares a las enumeradas anteriormente. Además, deberá contar con las librerías SDL, Open CV y Perl como mínimo para ser utilizado.

### **3.8.2 Compatibilidad**

El sistema es dependiente de plataforma, debe correr sobre una máquina que utilice Windows XP, esto debido al manejo de los gráficos.

### **3.8.3 Extensibilidad**

La extensibilidad se manejará en dos modalidades:

#### **3.8.3.1 *Software***

Para añadir funcionalidad, en el caso del acceso a la base de datos, este se hará a través de clases de Java, facilitando la modificación. En el caso de funcionalidad adicional en el juego debe evaluarse si realmente es necesario ya que al estar escrito en su mayoría en C++, puede que haya que recompilar extensas partes de código.

#### **3.8.3.2 *Hardware***

La extensibilidad de *hardware* se manejará de manera horizontal, es decir que si se necesita más capacidad de procesamiento, almacenamiento o memoria, esta se realizará agregando más servidores, pero al mismo tiempo se combinará con la modalidad de extensibilidad vertical, por medio de la cual se agregará mayor capacidad a los mismos servidores, pero con cierto límite. Después de este límite, pasaremos a la modalidad horizontal.

### **3.8.4 Mantenibilidad**

El mantenimiento tanto de *software* como de *hardware* se realizará periódicamente, principalmente como medida de prevención para evitar fallas. El *software* será mantenido para evitar que pueda volverse lento en tiempo de respuesta o de procesamiento. El *hardware* será verificado una vez

mensualmente, y llevará un mantenimiento general al menos una vez cada tres meses.

### **3.8.5 Configurabilidad**

Las interfaces utilizadas pueden ser configuradas muy fácilmente con algunos asistentes del sistema, por lo que la configuración se realiza de la manera más fácil y ágil.

### **3.9 Restricciones de diseño**

Para garantizar la estabilidad y seguridad del sistema, se deberá establecer un servidor de *Web* con sistema operativo Windows XP, y en ese mismo servidor debe instalarse un servidor de aplicaciones Java para el manejo del acceso a la base de datos.

La red deberá estar estructurada para que las peticiones entren y salgan por una sola vía (full dúplex) para que el tráfico pueda ser analizado antes de contactar la información, y luego las peticiones serán atendidas.

Es necesario utilizar un *firewall* de *hardware* como medida de seguridad para evitar ataques de Internet, y por último, se recomienda altamente utilizar el esquema de base de datos en paralelo así como el esquema de recuperación de desastres.





## **4. ARQUITECTURA DEL SISTEMA**

### **4.1 Propósito**

El presente documento muestra la arquitectura del sistema propuesto, utilizando una serie de vistas que permiten visualizar diferentes aspectos del sistema. Se muestran las decisiones significativas sobre la arquitectura del sistema que se han hecho.

Este documento está dirigido a las personas que trabajen sobre el sistema, presentando de una forma clara el modo en que el sistema será construido.

### **4.2 Ámbito**

Se creará un sistema para utilizarse en el Comando Superior de Educación en los cursos de puesto de comando.

Esto permitirá mejorar la forma en que estos cursos se desarrollan, facilitando la evaluación.

### **4.3 Definiciones, acrónimos y abreviaciones**

- UML: Unified Modeling Language, lenguaje unificado de modelación, permite describir por medio de diagramas la forma en que el sistema funciona.
- COSEDE: Comando Superior de Educación

#### **4.4 Referencias**

Las referencias a utilizar serán los manuales obtenidos en el COSEDE.

#### **4.5 Vista general**

El documento de arquitectura de *software* contiene la arquitectura elegida para modelar el sistema a ser creado. Se utilizará un modelo de 4+1 vistas, por lo que a continuación se presentan las vistas de casos de uso, lógica, de procesos, de despliegue y de implementación en ese orden.

#### **4.6 Representación de la arquitectura**

En el presente documento, se presenta la arquitectura del sistema utilizando las 5 vistas que corresponden al método de 4+1. Las representaciones están hechas utilizando diagramas de UML, además de su correspondiente descripción.

El proyecto se divide entonces en los modelos de casos de uso e implementación, y las vistas de procesos, despliegue y lógica.

#### **4.7 Metas y límites de la arquitectura**

La aplicación estará hecha para cumplir los siguientes objetivos:

- El diseño y desarrollo de la documentación que apoyará la aplicación que se planea construir para que funcione en el Comando Superior de Educación.
- Sentar las bases para que al momento de implementarse la aplicación, esta sea confiable y eficiente.

- Cumplir con todos los requisitos que actualmente llenan los cursos de puesto de comando, mejorando la ejecución de ciertos procesos.
- Implementación del uso de tecnologías libres tales como SDL, Intel OpenCV, PostgreSQL y Java.

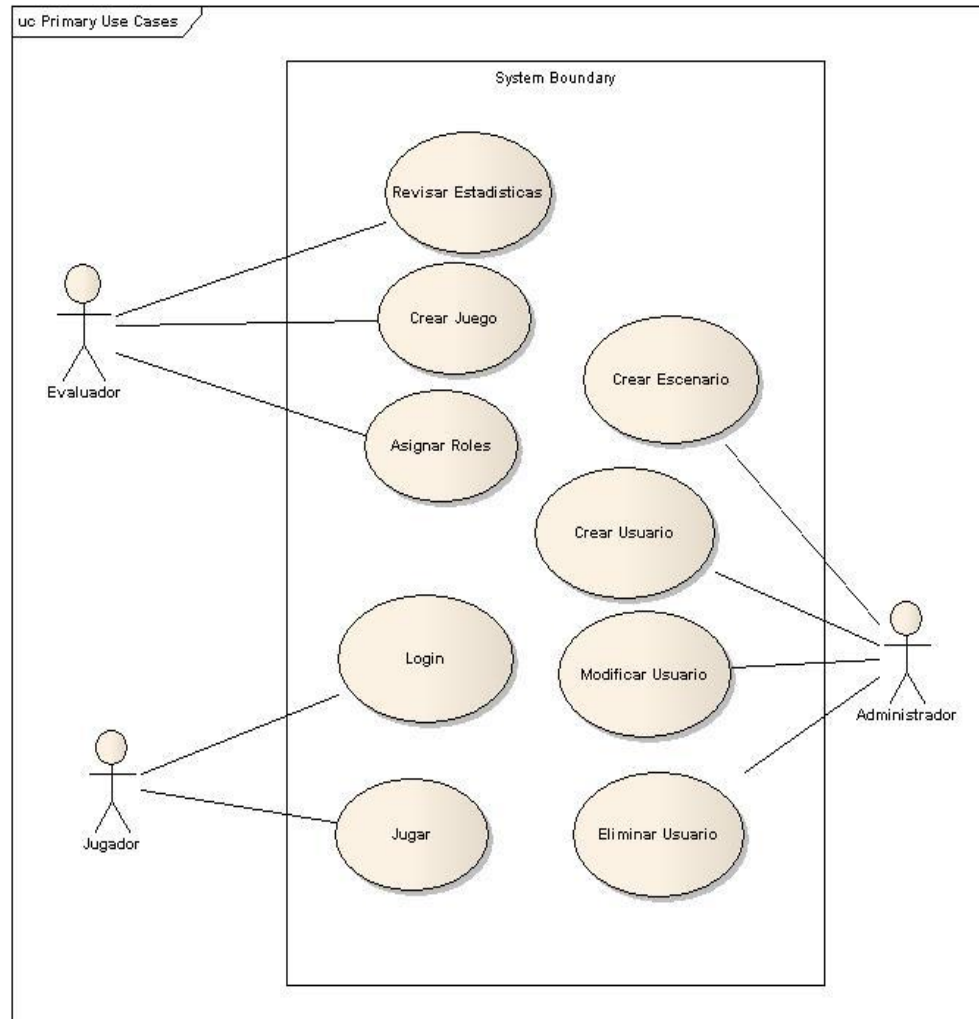
Los límites de la aplicación están dados por:

- Plataforma java J2EE
- Base de datos postgresQL
- Compatibilidad entre los distintos lenguajes a utilizar (C++, Perl y/o Python, Java)
- Plataforma Windows XP

#### **4.8 Vista de casos de uso**

Los casos de uso a utilizar en la arquitectura de la aplicación son los listados en el siguiente diagrama.

**Figura 16. Vista de casos de usos**

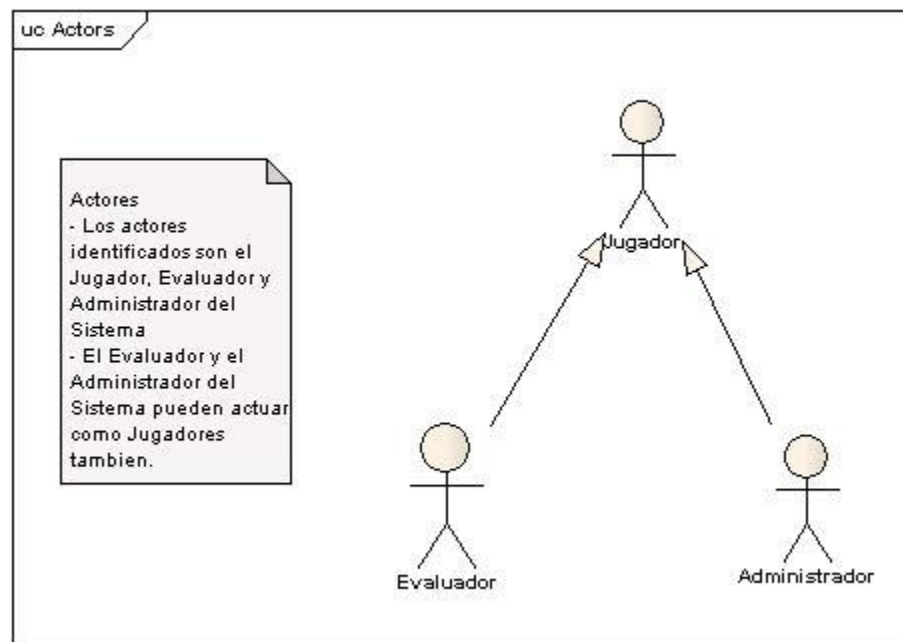


A continuación se presenta una breve descripción de los casos de uso que son significativos para la arquitectura.

### 4.8.1 Actores involucrados

Los actores involucrados en el sistema serán el jugador, el evaluador y el administrador. El evaluador y el administrador podrán actuar como jugadores también.

**Figura 17. Actores involucrados en el sistema**



Cada uno de los actores involucrados en el sistema tiene diversas actividades. Tanto el evaluador como el administrador tendrán acceso al sistema con privilegios de jugador añadidos para poder ingresar a observar el desarrollo de la partida en cualquier momento. Las actividades para cada actor se detallan a continuación.

#### 4.8.1.1 Actividades del actor jugador

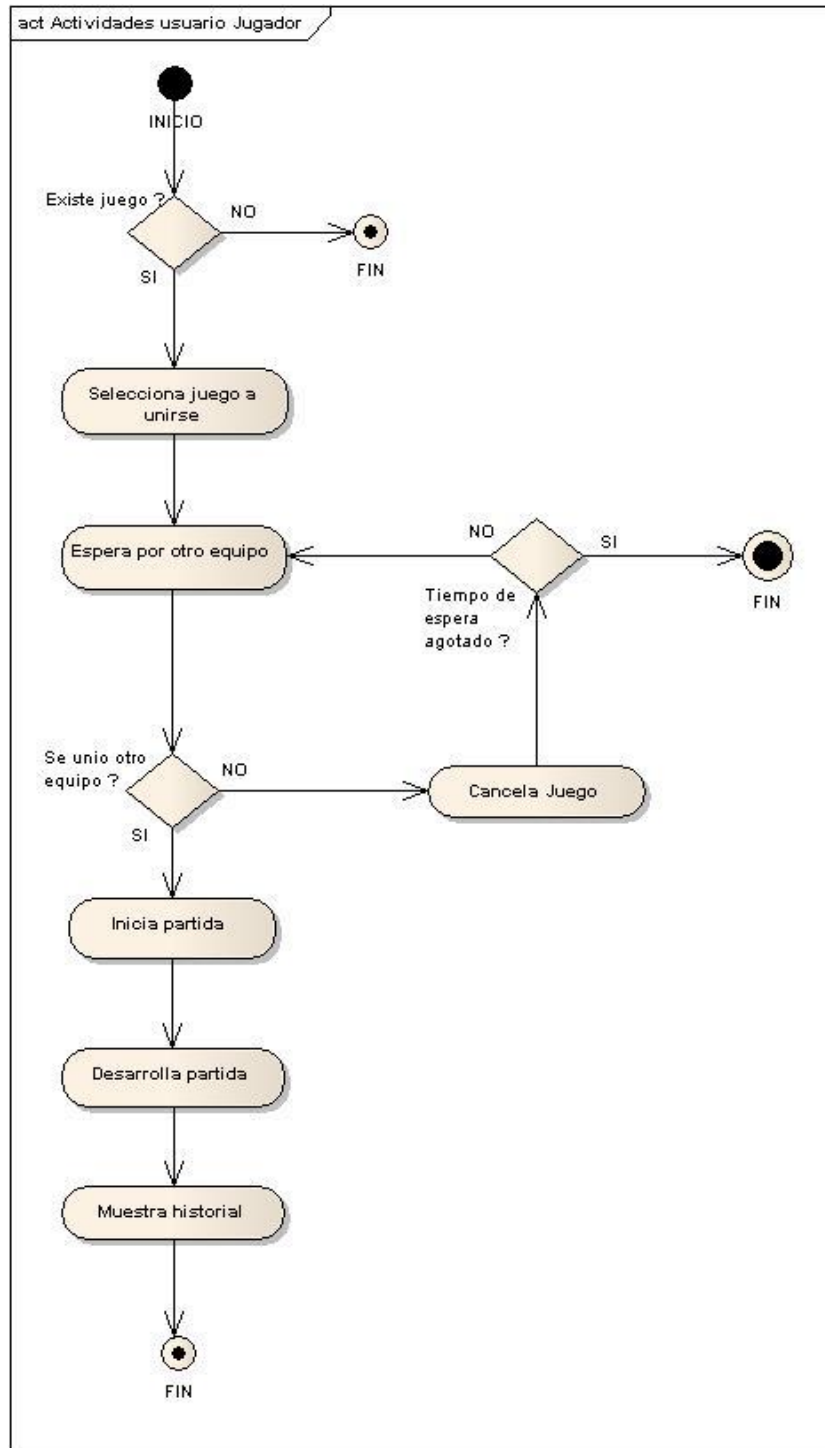
El actor jugador tiene permisos de ingresar al sistema y de jugar. En el diagrama se asume que ya está autenticado.

Una vez que este en el sistema, debe unirse a un juego que haya sido creado previamente por un usuario administrador y en el que hay un evaluador asignado.

A continuación queda en espera hasta que estén completos los contendientes o equipos, para iniciar el juego.

Después se desarrollan las distintas actividades del juego, y al finalizar este, se muestra al jugador el historial de las acciones que efectuó, así como sus resultados y estadísticas.

Figura 18. Actividades del jugador





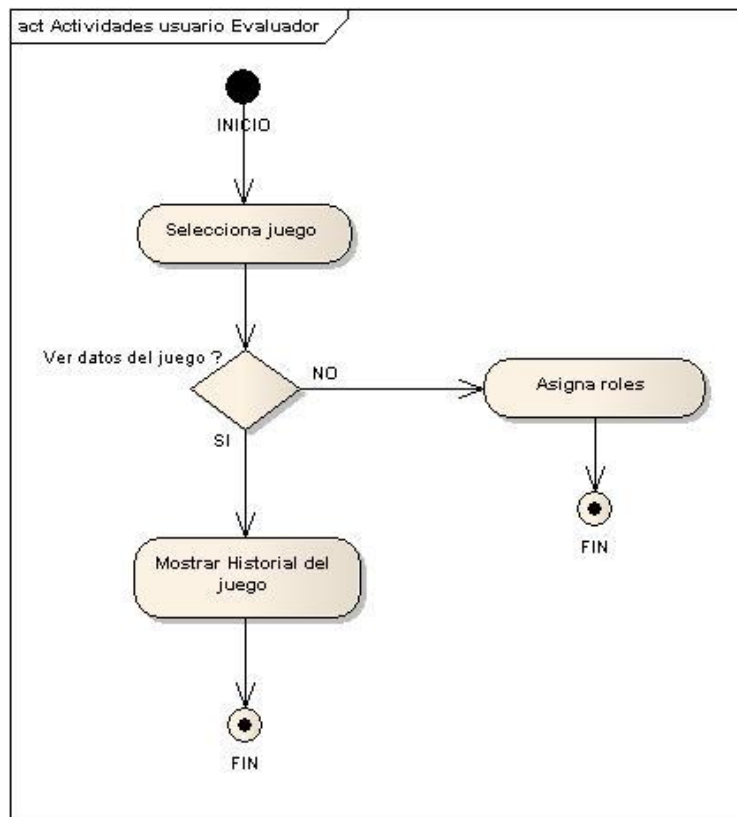
#### 4.8.1.2 Actividades del actor evaluador

El Evaluador, tras iniciar sesión debe de poder seleccionar un juego al cual unirse, a continuación ir dando los roles propios del escenario a los diversos jugadores que hayan ingresado.

También puede revisar el historial del juego, ya sea el que se está efectuando o cualquier otro cuyos datos estén guardados en el sistema.

El usuario Evaluador debe ser un instructor del COSEDE, experto en el juego de guerra.

**Figura 19. Actividades del evaluador**

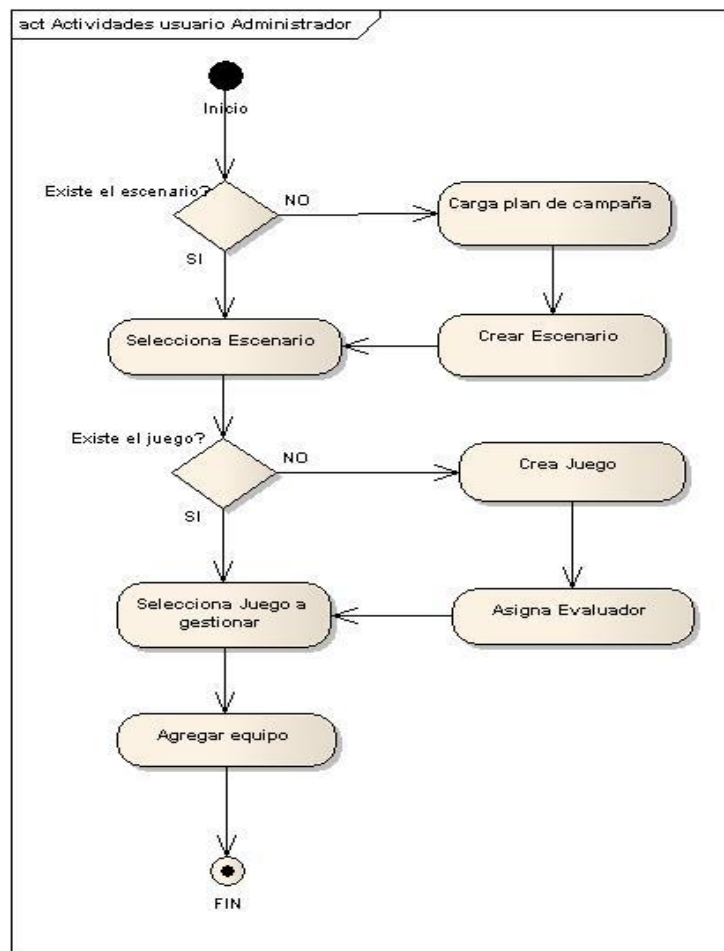


#### 4.8.1.1 Actividades del actor administrador

El Usuario Administrador se encarga de la parte técnica del sistema, la carga de datos de escenarios y unidades a la base de datos, así como de crear el juego, y gestionar los permisos de los distintos usuarios, tanto jugadores, evaluadores como otros administradores.

Idealmente el usuario administrador debe ser un experto en sistemas informáticos para poder lidiar con problemas que puedan darse.

**Figura 20. Actividades del administrador**



## **4.9 Casos de uso**

### **4.9.1 Login**

El caso de login está incluido para el actor Jugador, que lo hereda a su vez a los demás actores. Representa el inicio de sesión necesario para acceder al sistema y que se registre la actividad de cada usuario del mismo en una bitácora.

### **4.9.2 Jugar**

El caso de jugar representa la funcionalidad más importante del sistema, y engloba una serie de procesos que incluyen los movimientos y acciones de juego y la renderización de estos en las pantallas de los clientes.

Dichos procesos se encuentran descritos en detalle más adelante y en el documento de diseño del juego.

### **4.9.3 Crear juego**

En este caso de uso, el Instructor o Evaluador se encarga de crear el juego en base a un escenario antes cargado, y de ser necesario modificar las características que se encuentran configuradas por defecto.

### **4.9.4 Asignar Roles**

En este caso de uso, el instructor o evaluador asignara a cada uno de los usuarios que hayan iniciado sesión con rol de jugador un rol propio del escenario, para que pueda participar según su conocimiento específico.

#### **4.9.5 Revisar estadísticas**

Permite revisar las estadísticas de cada uno de los jugadores por parte del instructor o bien de un administrador. Esta revisión puede realizarse tanto dentro de un juego ya iniciado, con los resultados hasta el momento en que se hace la revisión, o bien revisar estadísticas históricas de un jugador o un juego específico.

#### **4.9.6 Crear escenario**

El Administrador del juego crea el escenario, incluyendo la carga a la base de datos del mapa, y la selección de las unidades que se van a proporcionar por defecto en los escenarios, así como la carga inicial de documentos de situaciones, y según el tipo de ejercicio pueden adjuntarse las ordenes de operaciones.

#### **4.9.7 Crear, modificar y eliminar usuario**

En estos casos de uso se administran los usuarios que pueden utilizar el sistema, dándoles sus diferentes roles según sea el caso.

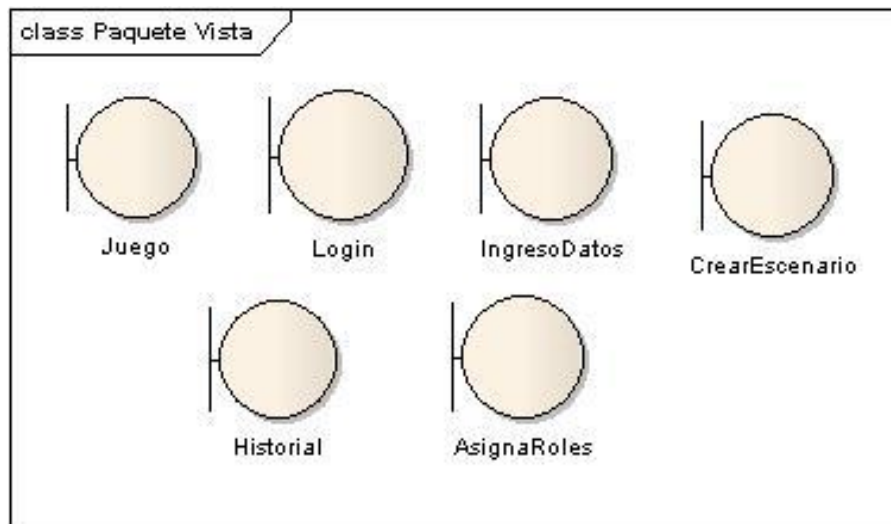
### **4.10 Vista lógica**

Las clases que se van a utilizar se han dividido en 3 partes, una de vista que representa la interfaz que se tendrá con los usuarios, una de control que representa las clases que se encargarán de la lógica del juego, y una de datos que representa los datos a utilizarse desde la base de datos a través de una herramienta de manejo de persistencia de datos.

#### 4.10.1 Clases del paquete de vista

En este paquete se representan los objetos que servirán para que los usuarios interactúen con el sistema, ya sea con ingresos de datos, configuraciones o bien el juego en sí.

**Figura 21. Clases de paquete de vista**



##### 4.10.1.1 Juego

No representa únicamente una clase, sino toda la pantalla de juego que verá el cliente y en el que podrá realizar sus acciones durante un ejercicio. Cualquier usuario puede entrar, ya sea que tenga un rol del escenario asignado o bien como observador. Debe ser hecha en C++ y Perl.

#### **4.10.1.2 Login**

Permite inicio de sesión, habrá 2 tipos de login, uno para ingreso de datos y otro para el juego.

#### **4.10.1.3 Ingreso de datos**

En esta página se ingresan los datos de doctrina contenidos en el Daplame, o bien de unidades nuevas que se desee agregar. Estará hecha en páginas JSP.

#### **4.10.1.4 Crear escenario**

Interfaz que el administrador debe utilizar para crear los escenarios, permite selección de mapa y de unidades que estarán por default para cada uno de los contendientes.

#### **4.10.1.5 Historial**

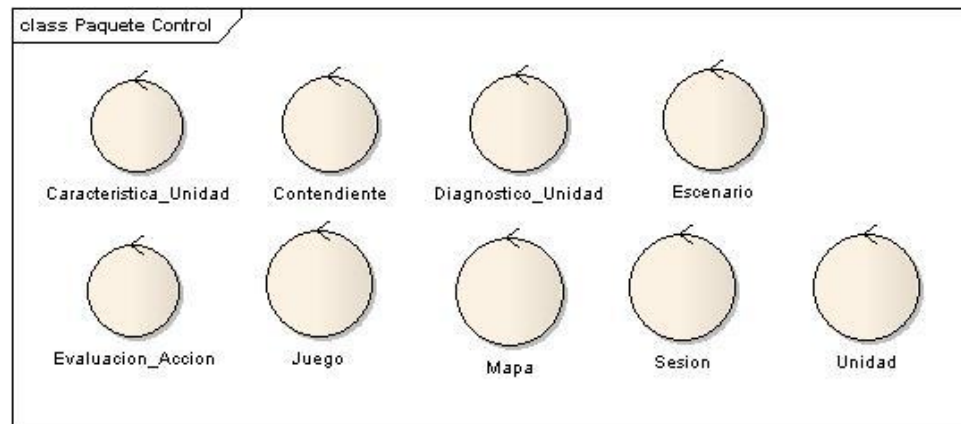
Despliega el historial y estadísticas de un jugador o un juego, dependiendo de los permisos del usuario que acceda.

#### **4.10.1.6 Asignar roles**

Permite al usuario evaluador asignar el rol del escenario a cada usuario jugador logueado en el sistema y conectado en el juego.

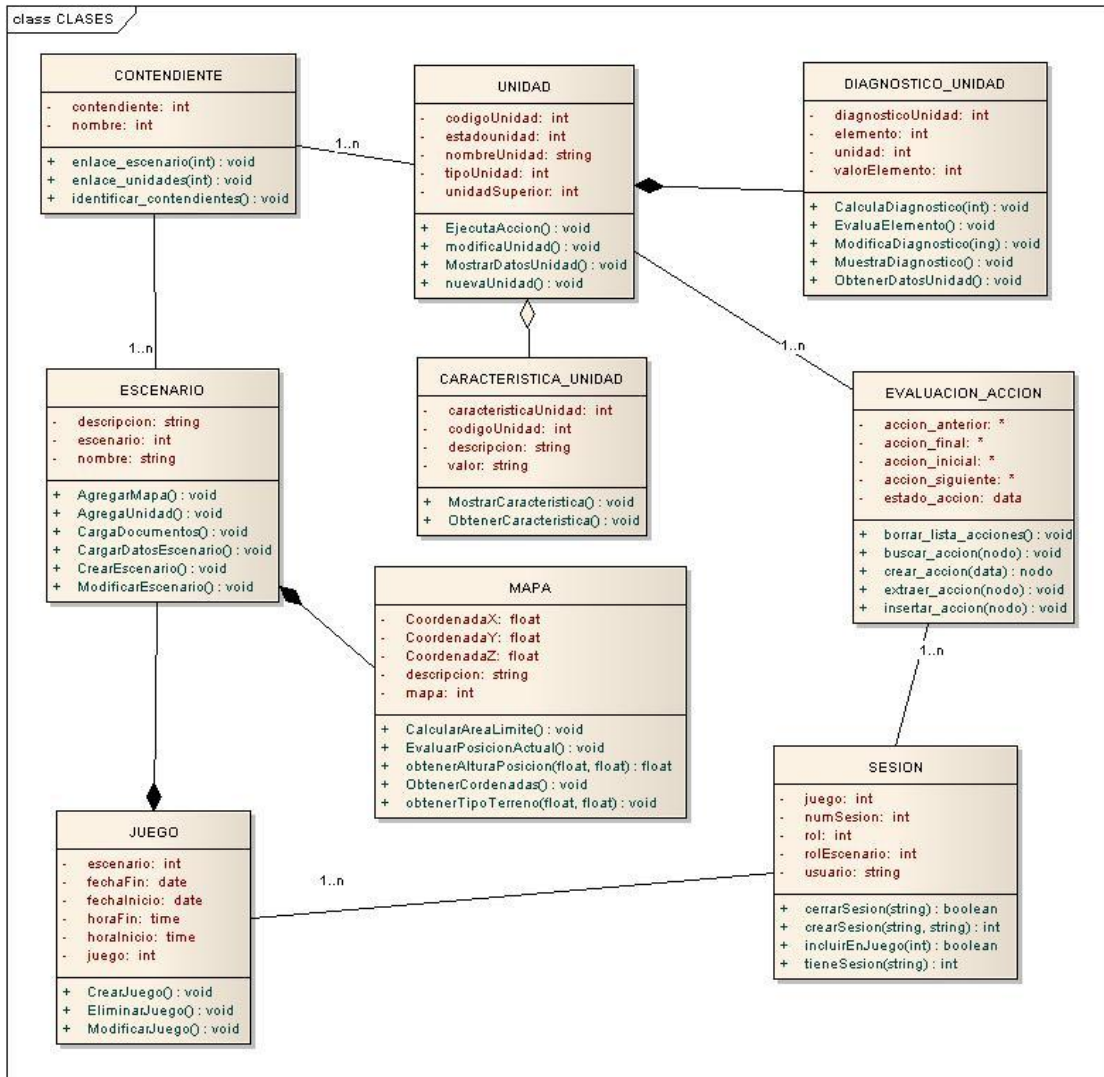
#### 4.10.2 Clases del paquete de vista

Figura 22. Clases paquete de control



En este paquete se representan los objetos que servirán para que los usuarios interactúen con el sistema, ya sea con ingresos de datos, configuraciones o bien el juego en sí.

Figura 23. Diagrama de clases



#### 4.10.2.1 Unidad

Representa unidades, ya sea individual o bien conjuntos de unidades (batallón, pelotón, etc.).



#### **4.10.2.2 Característica unidad**

Almacena las características que tiene una unidad, sobre todo capacidades de visibilidad, movimiento sobre terreno, rangos de disparo, etcétera, para poder evaluar si las acciones que da el jugador son válidas.

#### **4.10.2.3 Diagnóstico unidad**

Representa el estado actual de la unidad dentro del juego, así como si se han reducido algunas de sus capacidades, según el terreno, clima o el daño recibido.

#### **4.10.2.4 Evaluación acción**

Es una lista de acciones, se tendrán 3 instancias de la misma. Una para unidades que estén haciendo fuego (disparando), otra para unidades que estén en movimiento y una tercera para unidades estacionarias. De esta forma se priorizará como se evaluarán las acciones de cada unidad. Esta lista debe ser recorrida por un proceso a cada segundo del juego para obtener los resultados en tiempo real.

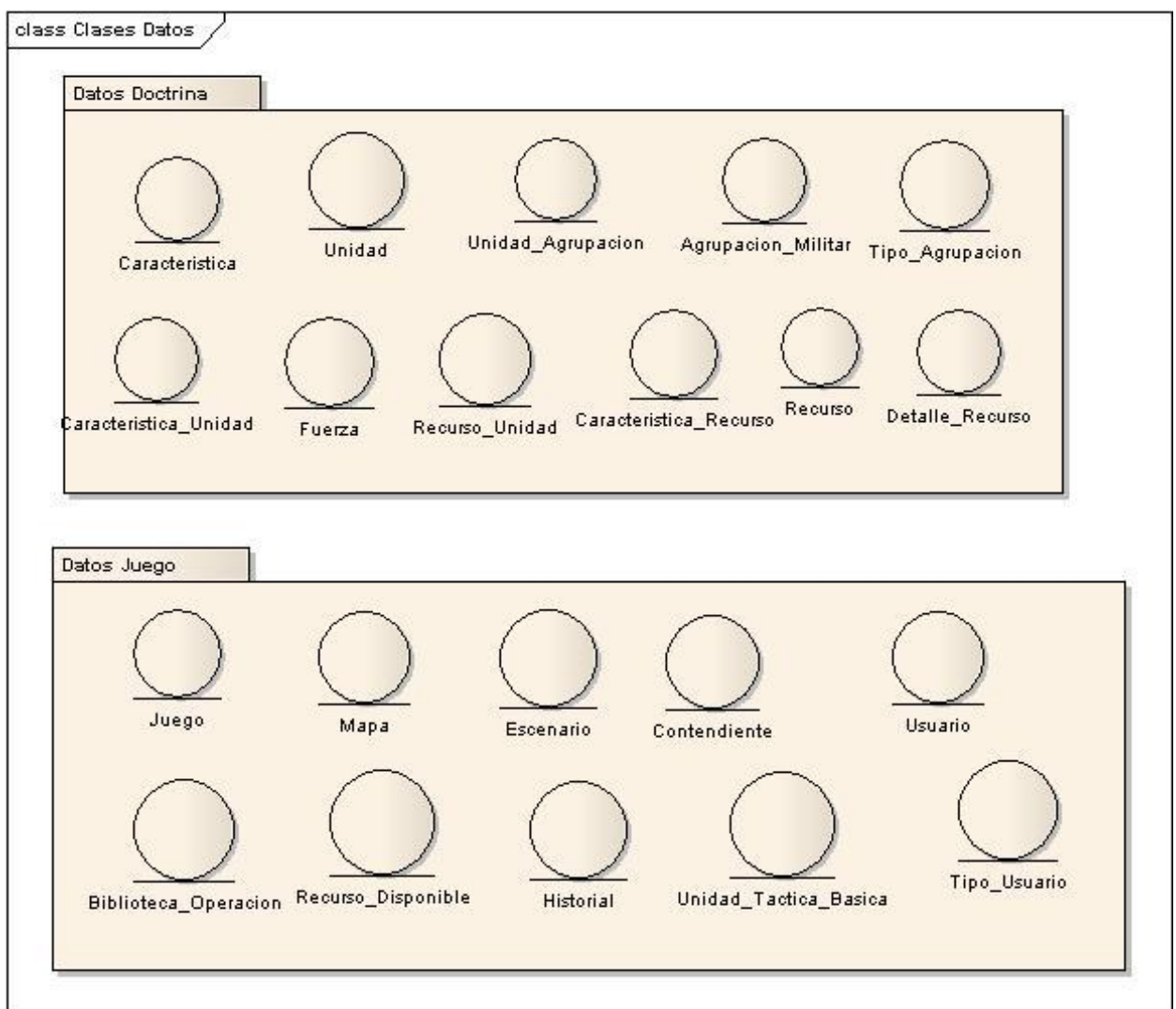
#### **4.10.3 Paquete de datos**

En este paquete se encuentra una representación de las clases que contendrán los datos del sistema. Esta capa será presentada a la capa superior por medio de un *software* que permita la persistencia de datos en Java en el caso de los datos de la doctrina. Este *software* debe pertenecer al grupo de los siguientes frameworks, o bien ser uno con una funcionalidad similar: Spring, Hibernate, EclipseLink, TopLink, etc.

En el caso de los datos del juego, estos serán presentados como clases en C++.

Cada una de las clases está basada directamente en una tabla de la base de datos y actúan como intermediario directo entre la aplicación y la base de datos, manteniendo una conexión abierta.

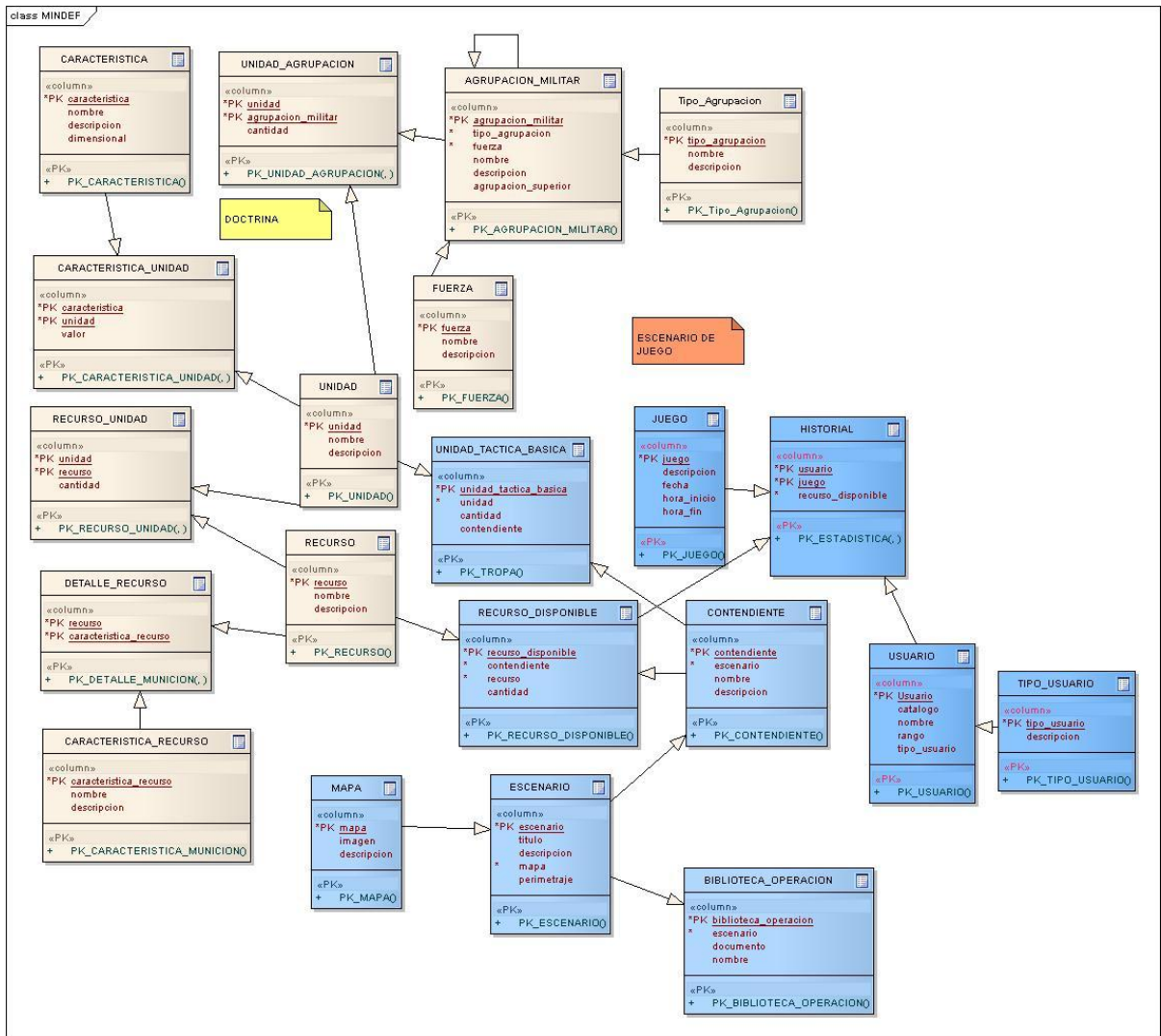
**Figura 24. Clases del paquete de datos**



### 4.10.3.1 Modelo entidad-relación

El diagrama de base de datos muestra el modelo entidad-relación que debe construirse en el manejador de base de datos y al que se accederá desde la aplicación a través del paquete de datos. El modelo es el siguiente:

Figura 25. Diagrama de base de datos



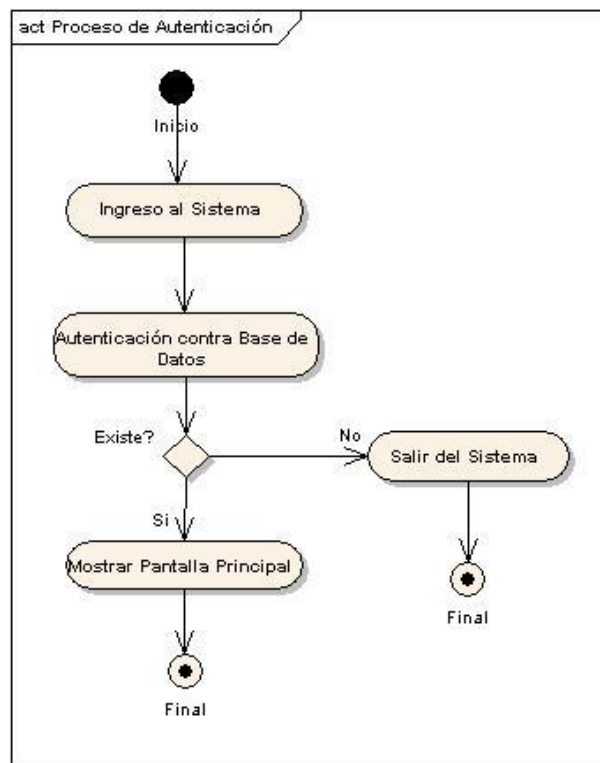
## 4.11 Vista de procesos

En esta sección se organizan los procesos y capas que se comunican e interactúan entre sí. Se describen además, los principales modos de comunicación entre los procesos.

### 4.11.1 Proceso de ingreso al sistema y autenticación

El proceso de autenticación es importante en términos de que es necesario para realizar cualquier otra operación dentro del sistema. Además, permite llevar control de quién efectúa cada acción. La autenticación se realiza contra el servidor de base de datos.

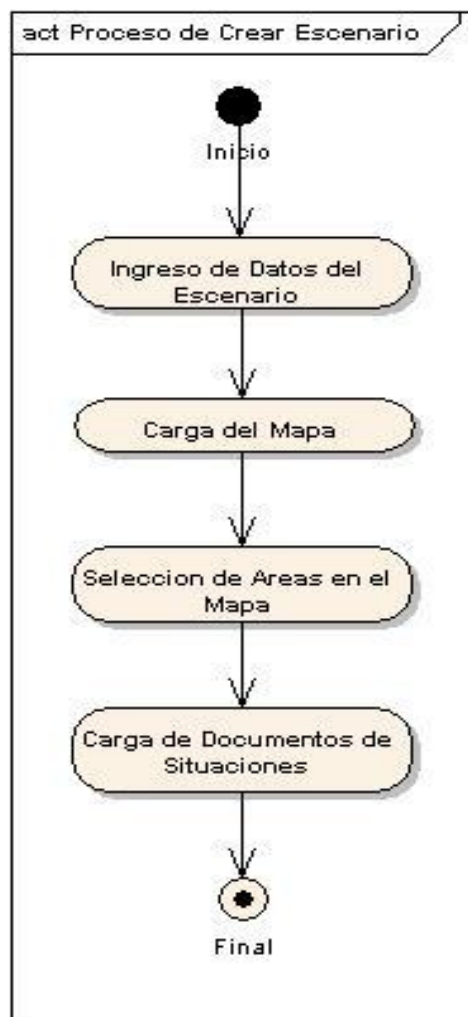
**Figura 26. Proceso de ingreso y autenticación del sistema**



#### 4.11.2 Proceso de crear escenario

Este proceso se efectúa para crear los escenarios sobre los que se realizarán las prácticas o juegos de guerra.

**Figura 27. Proceso de crear escenario**



Se incluyen aquí dos de las pantallas que tentativamente se van a utilizar.

a) Para crear la definición del escenario:

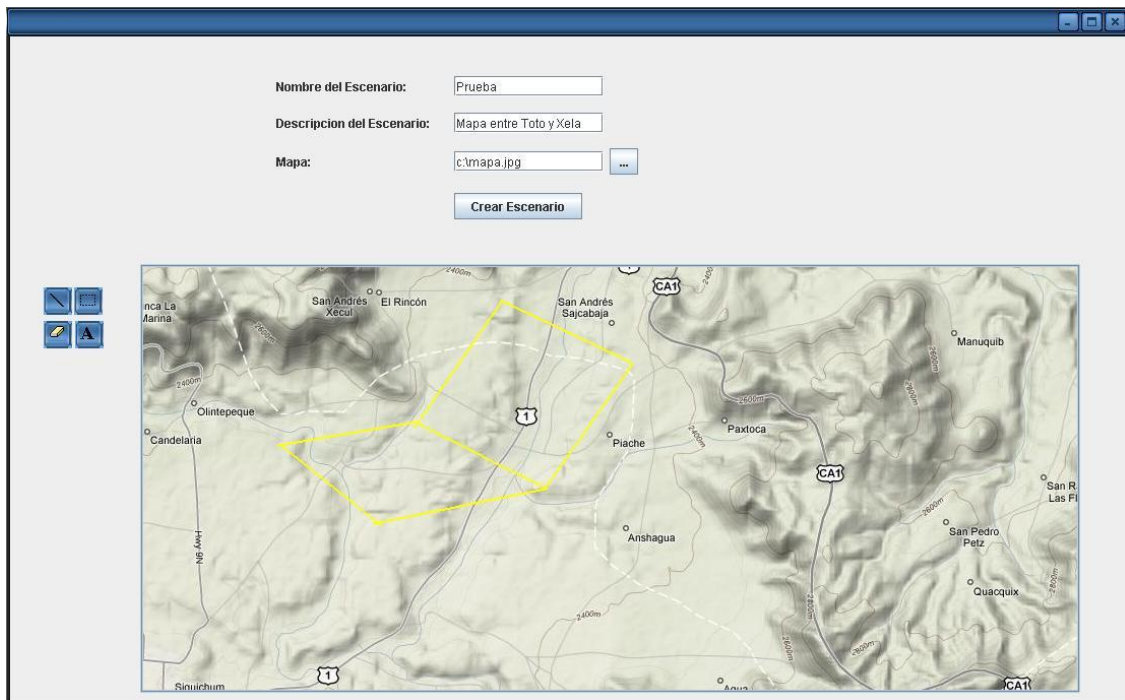
**Figura 28. Definición de escenario**



The image shows a screenshot of a software window with a blue title bar and standard window controls (minimize, maximize, close). The window contains a form for defining a scenario. It features three input fields: 'Nombre del Escenario:', 'Descripcion del Escenario:', and 'Mapa:'. The 'Mapa:' field includes a small button with three dots. Below the input fields is a button labeled 'Crear Escenario'.

b) Para seleccionar las áreas que se van a trabajar en el mapa.

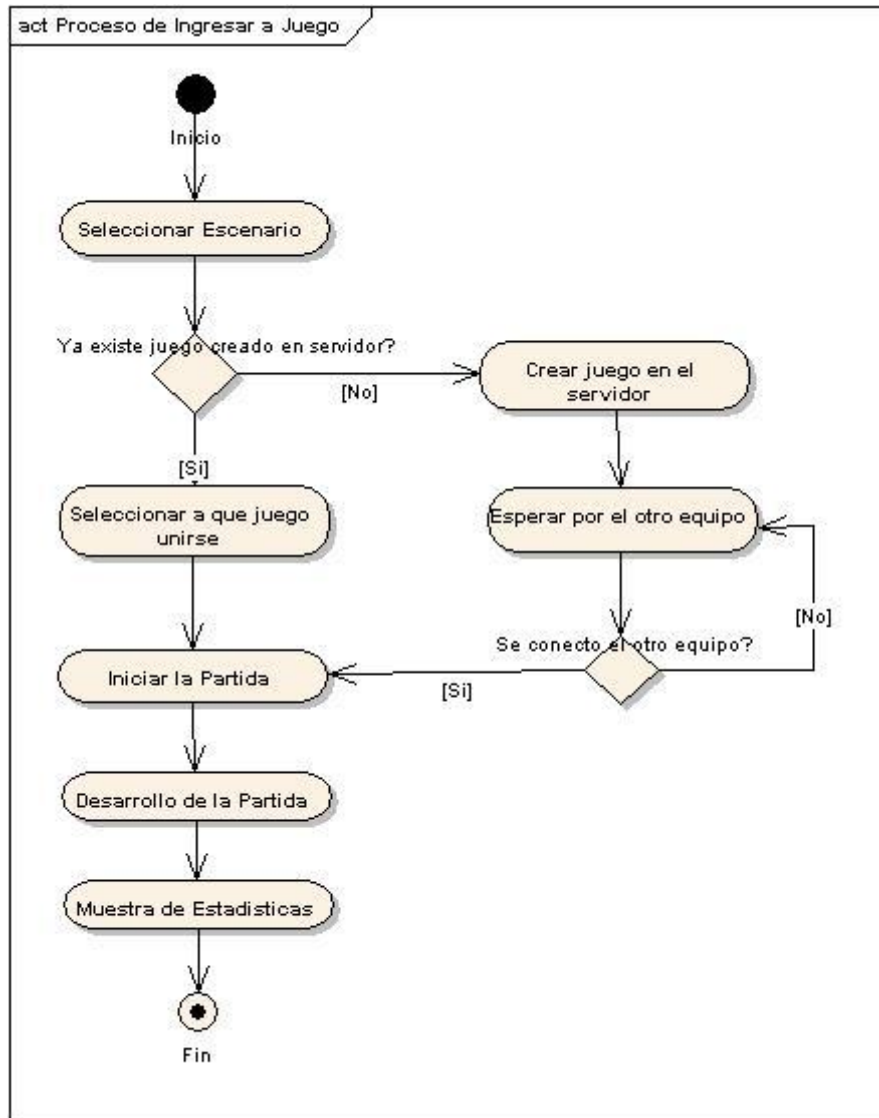
**Figura 29. Selección de áreas a trabajar.**



#### 4.11.3 Proceso de ingreso a un juego

Una vez que ya existen escenarios, se puede crear un juego, ingresando ya sea como jugador o bien como observador.

**Figura 30: Proceso de ingreso a un juego**

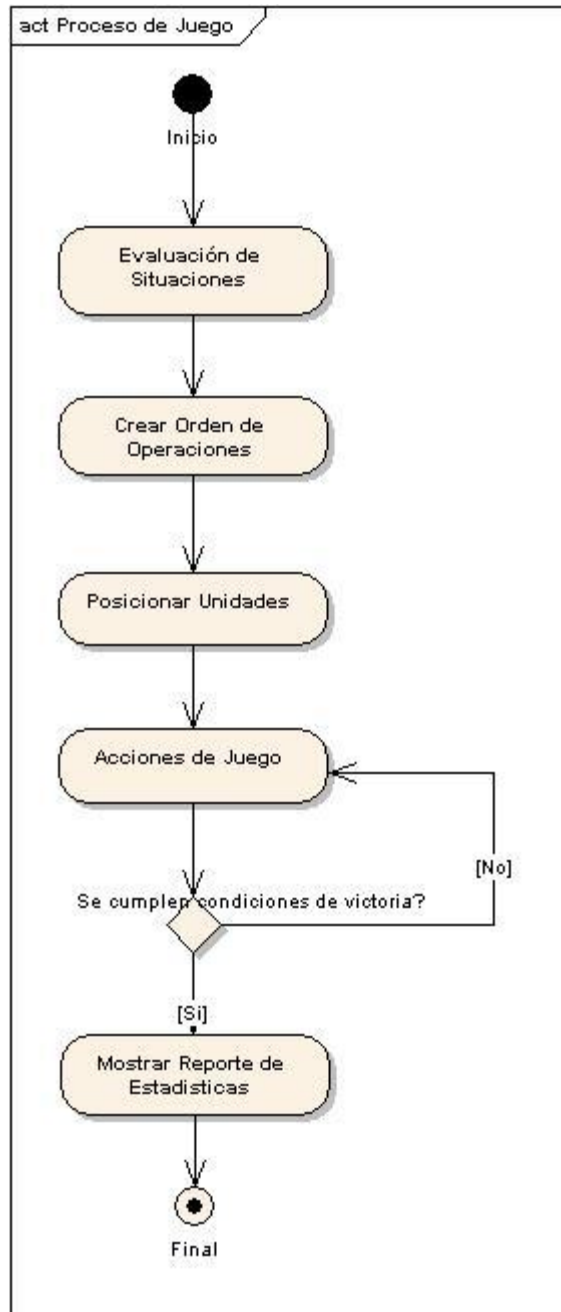


#### 4.11.4 Proceso de un juego

Dentro de un juego, se realizan los siguientes pasos:



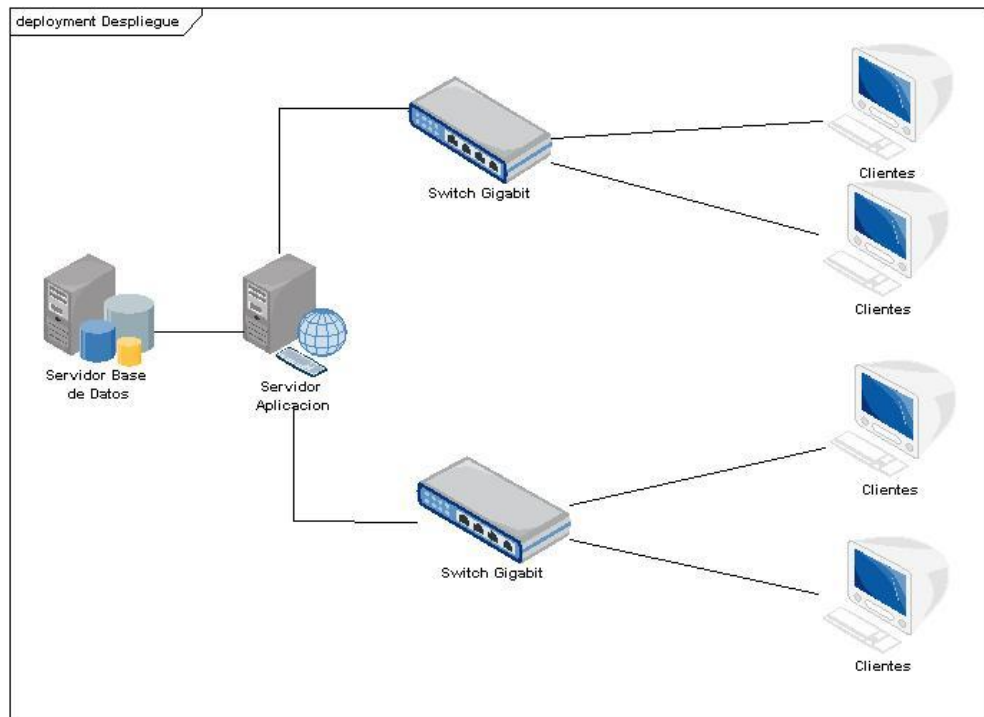
Figura 31. Proceso de juego



## 4.12 Vista despliegue

La arquitectura física utilizará un servidor principal, en el que se encontrará la base de datos, y el modelo que nos permitirá acceder a la misma. Este modelo estará hecho en Java, por lo que se requiere un servidor de aplicaciones para Java instalado también (Jboss, Glassfish o Tomcat). Además, deben estar instaladas las librerías SDL y OpenCV para el correcto manejo de imágenes y eventos. Los clientes tendrán instaladas aplicaciones pesadas que se encargarán de ir a traer la información al modelo y de enviar instrucciones al servidor sobre las posiciones en el mapa en que generarán eventos.

**Figura 32. Vista de despliegue**



#### **4.12.1 Ambiente de *hardware***

El *hardware* a utilizar debe de tener una capacidad mínima en función del número de clientes a atender. Lo recomendable en este caso debido al tamaño de la aplicación es utilizar servidores con recursos altos para el servidor central de base de datos y el de aplicación, con características tales como:

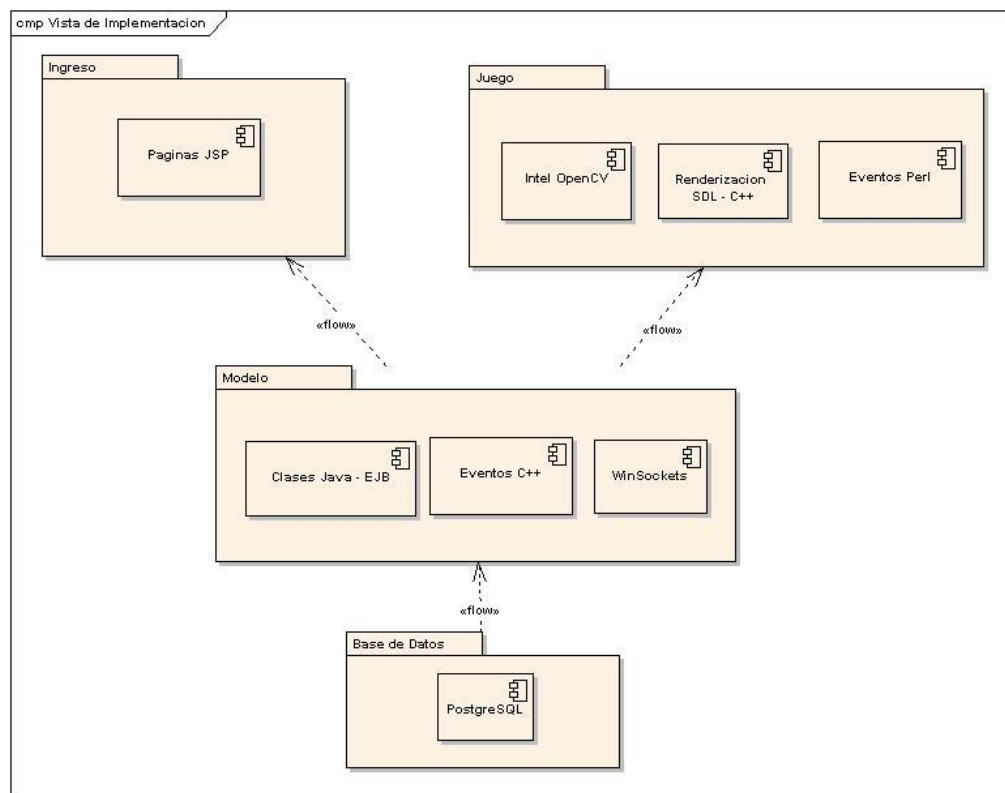
- Procesador Pentium 4 HT de 3 Ghz como mínimo.
- GB de memoria RAM.
- 512 Kbps de ancho de banda por conexión.
- Los clientes deberían tener una configuración:
- Procesador Pentium 4 de 2.4 Ghz.
- Tarjeta de Video aceleradora de 128 Mb como mínimo.
- 1 GB de memoria RAM.

## 4.13 Vista de implementación

### 4.13.1 Vista general

En el diagrama se puede observar el mapeo de las respectivas capas a los componentes que se utilizarán en el proyecto.

**Figura 33. Vista de implementación**



Como se puede ver, se utilizarán páginas JSP para el ingreso de datos a la base de datos. En el caso del paquete de juego que se puede ver, representa los clientes pesados, y parte de la aplicación que se ejecutará en el servidor. Esta parte es la que utilizará la librería OpenCV.

En la capa media, las clases de Java se deben utilizar como el medio de comunicación principal con la base de datos para el manejo de persistencia. En el caso de las aplicaciones cliente, estas utilizaran los WinSockets para comunicarse con el programa en C++, que a su vez utilizará también las clases de Java para obtener datos.

#### **4.13.2 Manejo de interfaz**

Para el manejo de interfaz, se utilizará JSP y JSF en el caso de los ingresos de datos, y Visual C++ en el caso de las aplicaciones cliente.

## CONCLUSIONES

1. Se logró reunir suficiente información por parte del Comando Superior de Educación del Ministerio de la Defensa Nacional. Dicha información lleva a concluir que se pudieron identificar claramente cuáles eran los requerimientos necesarios para poder llevar a cabo el diseño del Sistema de Simulación de Estrategias Militares.
2. Se logró analizar y mapear los requerimientos del sistema de lenguaje unificado, en el cual se definen claramente el funcionamiento de los procesos de elaboración de simulación de estrategias militares.
3. Se logró definir un documento de arquitectura del sistema, que proporciona todos los aspectos técnicos necesarios para la implementación de la propuesta tecnológica.
4. Se definieron claramente los procesos en el diseño del Sistema de Simulación de Estrategias Militares, como la parte principal de la solución tecnológica propuesta.
5. Se concluyó con éxito el diseño del funcionamiento y de la arquitectura del sistema que servirá de apoyo para los estudiantes del curso del Comando Superior de Educación del Ministerio de la Defensa.



## RECOMENDACIONES

1. Utilizar la propuesta de herramientas tecnológicas que se especifican dentro del diseño de la arquitectura del sistema, ya que al implementar y utilizar la aplicación, se puede llegar a tener un gran volumen de datos que serviría como una consistente minería de datos para poder extender el alcance del sistema hacia el ambiente de la simulación y predicción de estrategias militares.
2. Asignar los recursos necesarios para la implementación del sistema propuesto, ya que con el mismo el Ministerio de la Defensa, específicamente los estudiantes del Comando Superior de Educación, se verá beneficiado en la reducción de tiempo para la enseñanza y aprendizaje de soluciones de estrategias militares.





## BIBLIOGRAFÍA

1. LETIER TORRES, PATRICIO. **Desarrollo de software orientado a objetos usando UML.** (Documento electrónico). Departamento de Sistemas Informáticos, Universidad Politécnica de Valencia, España. (Cita 15 de Agosto de 2008). Disponible en: [http://eisc.univalle.edu.co/materias/Material\\_Desarrollo\\_Software/UML\\_Historia\\_Intro.pdf](http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/UML_Historia_Intro.pdf)
2. SENN, JAMES. **Análisis y diseño de sistemas de información.** Edición 1998. McGraw-Hill. México 1998. 942 pp.
3. KENDALL, KENNETH E. Y OTROS. **Análisis y diseño de sistemas.** 6ta. Edición. Pearson Educación. México 2005. Vol. 1. 726pp.
4. FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA.. **Solution architecture book.** TATA Consultancy Services. s.n., sf.
5. LAGUNA, MIGUEL A. **Modelado de sistemas software.** (Documento Electrónico). Departamento de Informática, Universidad de Valladolid, España 2003. (Cita 10 de marzo de 2009). Disponible en [www.infor.uva.es/~mlaguna/cd/Modelado%20de%20sistemas%20software.pdf](http://www.infor.uva.es/~mlaguna/cd/Modelado%20de%20sistemas%20software.pdf).
6. IRIBARNE MARTÍNEZ, LUIS F. **Glosario, modelo un modelo de mediación para el desarrollo de software basado en componentes COTS.** (Documento Electrónico). Departamento de Lenguajes y

- Computación, Universidad Almería, España. 2003. (Cita 10 de marzo de 2009). Disponible en: <http://www.cotstrader.com/thesis/glosary.pdf>.
7. WIKIPEDIA. **Modelo de plataforma específico**. (Página Web). WIKIPEDIA 2009. (Cita 10 de marzo de 2009). Disponible en: [http://en.wikipedia.org/wiki/Platform-specific\\_model](http://en.wikipedia.org/wiki/Platform-specific_model).
  8. WIKIPEDIA. **Modelo de plataforma independiente**. (Página Web). WIKIPEDIA 2009. (Cita 10 de marzo de 2009). Disponible en: [http://en.wikipedia.org/wiki/Platform-independent\\_model](http://en.wikipedia.org/wiki/Platform-independent_model).
  9. WIKIPEDIA. **Lenguaje unificado de modelado**. (Página Web). WIKIPEDIA 2009. (Cita 10 de marzo de 2009). Disponible en: [http://es.wikipedia.org/wiki/Lenguaje Unificado de Modelado](http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado).
  10. GUERRERO, JORGE Y PAVEZ, JOSÉ. **Manejo de crisis a nivel continental**. (Documento Electrónico). Biblioteca del Colegio Interamericano de Defensa, Chile 1996. (Cita 10 de marzo de 2009). Disponible en <http://library.jid.org/en/mono35/guerrero.pdf>
  11. EJÉRCITO DE GUATEMALA. **Manual de datos de planificación militar escolar**. Comando de Educación y Doctrina 2004. (Cita 10 de marzo de 2009).