SABIA: SOFTWARE DE APOYO AL APRENDIZAJE EN LÍNEA DE ALGORITMOS DE BÚSQUEDA PARA APLICACIONES DE INTELIGENCIA ARTIFICIAL

MARY LUZ JAIMES CHANAGÁ
CARLOS ALBERTO MATEUS PINZÓN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICO MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS
BUCARAMANGA
2004

SABIA: SOFTWARE DE APOYO AL APRENDIZAJE EN LÍNEA DE ALGORITMOS DE BÚSQUEDA PARA APLICACIONES DE INTELIGENCIA ARTIFICIAL

MARY LUZ JAIMES CHANAGÁ
CARLOS ALBERTO MATEUS PINZÓN

Proyecto de grado presentado como requisito para optar al título de Ingeniero de Sistemas.

Directora
Dra. MARTHA VITALIA CORREDOR MONTAGUT

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICO MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS
BUCARAMANGA
2004

DEDICATORIA

A mis padres Hipólito y María Alicia.

A mis hermanos, Lito, Blanca, Tere, Nico, Cely, Eduardo y Gilma.

A mis sobrinos.

MARY J.

DEDICATORIA

A mi mamá Ana María, a mi papá Jesús Alberto.

A mis hermanos, Lady, Blanca, Olga, Edward y Oscar.

A todos mis familiares y amigos.

CARLOS.

AGRADECIMIENTOS

A todos los que contribuyeron para el desarrollo de este proyecto.

A Dios quien nos regala sus bendiciones todos los días.

A la profesora Martha Vitalia quien fue nuestra directora de proyecto.

Al grupo GENTE por la ayuda que nos ofrecieron.

A nuestras familias que siempre creyeron en nosotros.

A todos, muchas gracias.

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

TITULO: SABIA: SOFTWARE DE APOYO AL APRENDIZAJE EN LINEA DE ALGORITMOS DE BUSQUEDA PARA APLICACIONES DE INTELIGENCIA ARTIFICIAL. [*]

AUTORES: MARY LUZ JAIMES CHANAGÁ
CARLOS ALBERTO MATEUS PINZÓN[**]

PALABRAS CLAVES: Inteligencia Artificial, Algoritmos de Búsqueda, Tecnologías de la Información y la Comunicación (TIC), Aula Virtual.

DESCRIPCIÓN.

SABIA se plantea como parte de un conjunto de herramientas que ayudará a los estudiantes de la asignatura de Inteligencia Artificial (IA) en el refuerzo de conceptos relacionados con Algoritmos de Búsqueda (AB), colocando a su disposición información que apoye la comprensión de los AB básicos que se aplican en la implementación de juegos, agentes inteligentes, sistemas expertos, etc. La herramienta abarca: descripción clara y concisa de los algoritmos para hacer que sean más fáciles de entender, simulación del recorrido en árboles de búsqueda para cada uno de éstos, ejemplos de su aplicación a problemas específicos, ejercicios propuestos que sirven para auto evaluar los conocimientos adquiridos y juegos para ver su uso y aplicación.

La realización de este proyecto apoya la definición de políticas institucionales y un escenario pedagógico para la educación en línea en la Universidad Industrial de Santander (UIS). Es un aporte a la experiencia de inicio de desarrollo de módulos software para apoyar experiencias de educación en línea en diferentes carreras de la UIS. Además, el desarrollo de este tipo de herramientas educativas con la característica de facilitar la actualización de contenidos, permite mantener al día la información que se ofrece a los estudiantes para apoyar sus procesos de aprendizaje.

Este libro contiene la descripción del proceso que se llevo a cabo para la elaboración de SABIA. En el capítulo 2 se hace la presentación del proyecto. En el capítulo 3 se encuentra el resumen de los fundamentos teóricos necesarios para el desarrollo del proyecto. En el capítulo 4 se explica con base en la metodología utilizada, el proceso de desarrollo de SABIA. En los capítulos posteriores se encuentra las conclusiones, las recomendaciones, la bibliografía y los anexos.

[*] Trabajo de grado.

[**] Facultad de Ciencias Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Directora: Dra. Martha Vitalia Corredor Montagut.

TITLE: SABIA: HELPING SOFTWARE FOR ON-LINE LEARNING OF SEARCHING ALGORITHMS FOR ARTIFICIAL INTELLIGENCE APPLICATIONS.[*]

AUTHORS: MARY LUZ JAIMES CHANAGÁ
CARLOS ALBERTO MATEUS PINZÓN[**]

KEY WORDS: Artificial Intelligence, searching algorithms, information and communication technologies (TIC), virtual classroom.

DESCRIPTION.

This software is set out as a group of tools which will help the students of Artificial Intelligence subject (AI) to reinforce the concepts related to the Searching Algorithms (SA), giving to them the required information in the comprehension of the basic SA that are applied in the implementation of games, intelligent agents, expert systems, etc. This tool involves: a clear brief description of the algorithms in order to make it easy to understand, a simulation of the searching trees route for each one of the algorithms, an example of the application of specific problems, some exercises to auto evaluate the acquired knowledge and games in order to see the use and the application of the algorithms.

The realization of this project supports the definition of the institutional policies, a pedagogic environment for the on-line education at Universidad Industrial de Santander (UIS). This is a contribution to the starting development of software units to support the on-line education experience of the different careers at UIS. Besides that, the development of this kind of tools that facilitate the data updating allows to bring the information which is offered to the students up to date in order to support their learning process.

This book contains the description of the process that was used in order to create the software SABIA. Chapter number two presents the project, chapter number three exposes the summary of the required theoretical basis for the development of the project, chapter number four explains the development process according to the methodology used; and the next chapters contains conclusions, suggestions, biography and annexes.

[*] Work of degree.
[**] Physical Mechanic Sciences Faculty. Systems and Informatics Engineering School. Director: Dr. Martha Vitalia Corredor Montagut.

1. INTRODUCCIÓN

El mundo se mueve aceleradamente como resultado de las exigencias cada vez mayores para sobrevivir. Por esto las universidades, comprometidas con la calidad, en especial la Universidad Industrial de Santander (UIS), se esfuerzan en formar personas, ciudadanos, profesionales y científicos de alto nivel, de forma que tengan ventajas al momento de incorporarse a la vida laboral. Este propósito se logra ofreciendo experiencias de aprendizaje mediado que facilitan el desarrollo de competencias cognitivas, procedimentales y actitudinales, que aseguran a los estudiantes el aprender a aprender. Para el apoyo de estas experiencias se hace necesaria la innovación permanente en las metodologías de aprendizaje y enseñanza.

En este sentido, se han buscado métodos de enseñanza para tratar de aprovechar y distribuir el tiempo dispuesto para las asignaturas, de modo que los estudiantes logren captar, organizar y analizar la gran cantidad de información disponible en la actualidad. Así, dadas las ventajas que ofrecen las Tecnologías de la Información y Comunicación (TIC), en cuanto a la posibilidad de acceso, organización y procesamiento de la información, las facilidades para la interacción y comunicación permanente entre los diversos actores educativos, la representación hipermedial de contenidos, la simulación de procesos y el apoyo al trabajo colaborativo, se han planteado, herramientas que

puedan dar apoyo a los procesos de formación integral en el ámbito de la educación superior.

En la UIS particularmente se han iniciado experiencias en este sentido con apoyo de diversos entornos, dentro de los cuales se encuentra "Aula Virtu@l"[1], entorno que permite ofrecer contenidos y herramientas de aprendizaje, con las cuales los estudiantes podrán reforzar los conocimientos adquiridos en clase. SABIA se plantea como parte de ese conjunto de herramientas que apoyará a los estudiantes de la asignatura de Inteligencia Artificial (IA) en el refuerzo de conceptos relacionados con algoritmos de búsqueda.

SABIA surge como respuesta a una necesidad de colocar a disposición de los estudiantes información, ejemplos y ejercicios y apoyar la comprensión de los Algoritmos de Búsqueda básicos que se aplican en la implementación de juegos, agentes inteligentes, sistemas expertos, etc. Está enfocada principalmente a los estudiantes de IA, pero también queda disponible como herramienta específica para los profesores que consideren oportuno colocarla a disposición de sus estudiantes. Presenta contenidos, simulaciones y ejemplos que explican de manera clara y sencilla cada algoritmo; ejercicios propuestos que sirven para auto evaluar los conocimientos adquiridos; y juegos para ver su uso y aplicación.

Este libro contiene la descripción del proceso que se llevo a cabo para la elaboración de SABIA. En el capítulo 2 se hace la presentación del

[1] Plataforma para el montaje de experiencias virtuales de aprendizaje, elaborada como proyecto de grado de Erwin Meza Vega, bajo la dirección del Grupo de Estudio e iNvestigación en Tecnologías y Educación (GENTE).

proyecto, que incluye el planteamiento del problema, los objetivos y la justificación. En el capítulo 3 se encuentra el resumen de los fundamentos teóricos necesarios para el desarrollo del proyecto como: TIC, especialmente aula virtual; ingeniería del software donde se hace énfasis en UML como leguaje de modelado, prototipado evolutivo como metodología de desarrollo escogida, POO, Java y Dreamweaver como herramientas de desarrollo; e Inteligencia Artificial teniendo como tema principal los algoritmos de búsqueda. En el capítulo 4 se explica con base en la metodología utilizada, el proceso de desarrollo de SABIA. En los capítulos posteriores se encuentra las conclusiones, las recomendaciones, la bibliografía y los anexos.

2. PRESENTACION

2.1. PLANTEAMIENTO DEL PROBLEMA

En la Escuela de Ingeniería de Sistemas e Informática de La Universidad Industrial de Santander, se dicta como electiva técnica profesional la asignatura de Inteligencia Artificial (IA). Dentro de los contenidos de ésta se plantea la revisión y el estudio, una vez vistos algunos conceptos básicos, de la temática sobre los Algoritmos de Búsqueda, tema cuyo desarrollo presenta algunos problemas en los cuales nos centraremos para el desarrollo de este proyecto.

Los Algoritmos de Búsqueda son procedimientos que permiten encontrar una secuencia de operadores para hallar solución(es) a un problema específico, soluciones que pueden ser las óptimas o simplemente buenas para el usuario. Existen muchos Algoritmos de Búsqueda, algunos de ellos con un grado de complejidad elevado, lo cual hace que su estudio exija una dedicación alta de tiempo a su análisis y comprensión, situación que combinada con las pocas horas asignadas a la asignatura de IA, ocasionan dificultades en el aprendizaje y la aplicación de conceptos relacionados, por parte de los estudiantes matriculados en ella.

Las falencias se presentan principalmente en aspectos que se refieren a la comprensión de las características de su funcionamiento, estructura,

comportamiento, comprensión de su lógica, análisis de los parámetros de complejidad y la aplicación en solución de problemas prácticos.

Dado lo anterior consideramos pertinente aportar, desde nuestra perspectiva de ingenieros, a contrarrestar la situación anterior para mejorar procesos educativos de los estudiantes que cursen la asignatura de IA, así como promover el uso de plataformas y experiencias virtuales para apoyar aprendizajes significativos y dinámicos, donde el estudiante tendrá libertad para acceder a información que estará disponible en todo momento en el "Aula Virtu@l".

2.2. OBJETIVO GENERAL

Desarrollar una herramienta software, que facilite la comprensión de las características del funcionamiento y la aplicación de los Algoritmos de Búsqueda como técnicas para construir Agentes Inteligentes, de forma que apoye la adquisición de conceptos utilizados en los desarrollos de Inteligencia Artificial.

2.3. OBJETIVOS ESPECÍFICOS

- Identificar las características principales de los Algoritmos de Búsqueda en Amplitud, Profundidad, Profundidad con retroceso, Grafos, Irrevocable, Primero el mejor, A*, Poda Alfa-Beta y Mínimax.

- Analizar la lógica de los Algoritmos de Búsqueda mencionados para identificar su estructura, comportamiento y complejidad, de manera que se facilite su comprensión e implementación como objeto de estudio de la Inteligencia Artificial.

- Implementar los Algoritmos de Búsqueda mediante módulos en línea para el entorno "Aula Virtu@l", que faciliten a quienes los usen:

  - Disponer de manera permanente de contenidos, actividades y experiencias que les apoye el aprendizaje sobre el tema.

  - La comprensión de la lógica de los algoritmos y de la estrategia de selección de caminos que cada uno de éstos plantea para la solución de un problema.

  - El análisis de su eficiencia y de los campos de aplicación en la solución de problemas.

  - El desarrollo de ejercicios que permitan evaluar la comprensión de la lógica de la estrategia de búsqueda de cada uno de los algoritmos y las posibilidades de aplicación de éstos.

2.4. JUSTIFICACIÓN

"No es mi intención la de sorprenderlos ni espantarlos – pero la mejor manera de hacer un resumen es señalar que ahora existen en el mundo máquinas que pueden pensar, que pueden aprender y que pueden crear. Aún más, la habilidad para obtener esos atributos va a crecer con rapidez hasta que – en un futuro ya visible – el rango de los problemas capaces de ser gestionados va a ser coextensivo con el rango de aplicación de la mente humana"

Herbert Simon

La IA tiene en cuenta los últimos descubrimientos en cuanto a ejecución y simulación de procesos inteligentes en máquinas y por máquinas, las nuevas tecnologías de la Informática, Ciencia de Materiales y la Electrónica. Los problemas de investigación de la IA constituyen una frontera en continuo movimiento. De manera general y un poco más precisa podría decirse que la IA trata de construir máquinas con comportamiento aparentemente inteligente.

Desde su origen han sido muchas las ramas que han brotado de la IA, se habla de sistemas expertos, sistemas basados en el conocimiento (SBC), vida artificial, algoritmos genéticos, computación molecular o redes neuronales, comprensión del habla y modelado semántico, visión artificial, robótica, entre otras.

A pesar de las dudas que hay con respecto a donde se llegará en los desarrollos de IA, sus avances son una realidad. Como muestra de ello podemos observar la gran cantidad de robots o máquinas que se han creado para la realización de trabajos, cuya ejecución puede representar un riesgo o resultar muy tediosos para ser desarrollados por un ser

humano; en muchos de estos trabajos se necesita de la toma de decisiones, claro, sin esperar que una máquina tome decisiones como de pronto lo haría una persona en una situación inesperada.

Todo lo dicho anteriormente deja entrever la importancia que tiene la IA en nuestras vidas, por esto nuestro proyecto quiere hacer énfasis en los Algoritmos de Búsqueda, los cuales son pieza clave en el desarrollo de las aplicaciones de las diferentes ramas de la IA. Todo en IA se reduce a búsquedas; cada problema es un árbol "virtual" de todas las posibilidades, de buen o mal éxito, el truco reside en encontrar estrategias de búsqueda eficientes y óptimas para cada situación particular.

Debido a la gran cantidad de problemas que aborda la IA, existe también gran variedad de Algoritmos de Búsqueda a tenerse en cuenta para su solución. En libros, revistas, sitios de Internet, etc., se ofrecen documentos para su respectivo estudio pero que resultan enfocados a personas que ya tienen conocimiento de la temática, situación que torna difícil el estudio de los Algoritmos de Búsqueda a los estudiantes matriculados en la asignatura de Inteligencia Artificial de la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander, ya que ésta, en la mayoría de los casos, constituye su primer acercamiento a la IA.

Para contribuir a la mejora de esta situación desarrollaremos una herramienta software enfocada al aprendizaje de los Algoritmos de Búsqueda, que tendrá en cuenta el contexto, los intereses y las necesidades de los estudiantes de la asignatura y de todos aquellos

que la quieran consultar. Esta herramienta abarcará: descripción clara y concisa de los algoritmos para hacer que sean más fáciles de entender, simulación del recorrido del árbol para cada uno de éstos, ejemplos de su aplicación a problemas específicos y ejercicios propuestos.

La realización de este proyecto apoya la definición de políticas institucionales y un escenario pedagógico para la educación en línea en la UIS. Será un aporte a la experiencia de inicio de desarrollo de módulos software para apoyar experiencias de educación en línea en diferentes carreras de la UIS.

3. MARCO TEORICO

3.1. TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN (TIC)[2]

Las nuevas Tecnologías de la Información y Comunicación (TIC) son definidas por los expertos como "tecnología disruptiva" allí donde se ha aplicado. Rompen e innovan, pero reforzando lo fundamental en cada ámbito. Así ha sucedido, por ejemplo, en las finanzas, en el comercio electrónico o en la comunicación interpersonal.

También en educación las TIC rompen esquemas o didácticas tradicionales, pero reforzando los aspectos básicos de la enseñanza y del aprendizaje. ¿Qué aspectos? Primero, la información, permanentemente disponible y fácilmente accesible, con la que se construye el conocimiento. Segundo, la actividad o intervención permanente del alumno; el programa útil educativamente es el que hace a la máquina "tonta" para que el alumno actúe inteligentemente. Y finalmente, el más importante, la interacción.

Las TIC son una herramienta que potencia la intervención del profesor: la comunicación con los alumnos y la dirección del aprendizaje. Sin esta interacción del profesor con el alumno la información puede ser simple ruido y la actividad del alumno corre el riesgo de ir a ciegas o resultar

[2] http://www.educastur.princast.es/Revistanueva

inútil. Es en la interacción donde se hila la información y se teje el conocimiento. Cambian los telares, pero lo básico de la relación entre maestro y aprendices para generar saberes y destrezas permanece: la guía y la comunicación.

Como apoyo al uso de las TIC en los procesos educativos se ofrecen las experiencias de enseñanza virtual. En esta dirección "Aula Virtu@l" pretende facilitar el desarrollo de ambientes virtuales de aprendizaje. Las experiencias que se facilitan mediante los servicios de "Aula Virtu@l" consideran: el uso de las TIC como instancias de mediación; la resolución de problemas y el aprendizaje colaborativo como estrategias para favorecer el aprendizaje significativo, pues apoyan el desarrollo de competencias cognitivas, actitudinales y procedimentales; el estudiante como protagonista del aprendizaje y el profesor como facilitador o guía, desde el lado, de las experiencias de aprendizaje, y la caracterización de las experiencias virtuales como apoyo al modelo pedagógico donde se utilizan las estrategias de aprendizaje señaladas[3].

3.2. INGENIERIA DEL SOFTWARE

Es un enfoque sistemático del desarrollo, la operación, el mantenimiento y retiro del software, en otras palabras, se considera que la ingeniería del software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones

[3] **CORREDOR** Montagut, Marta Vitalia y otros. Aula Virtu@l: "una alternativa en educación superior". Bucaramanga. Ediciones UIS. 2004.

costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software[4].

La ingeniería de software como disciplina ha evolucionado significativamente en lo que se refiere a modelos conceptuales y herramientas de trabajo, que hacen del proceso de desarrollo y mantenimiento de software una actividad cada vez menos dependiente del arte de quienes llevan a la práctica un diseño elaborado.

3.2.1. Metodología de Desarrollo de Software

La generación de cualquier producto es un proceso que involucra la ejecución de distintas etapas o fases de producción. Al considerarse una aplicación académica un producto, concreto y tangible, su generación también debe atravesar por distintas etapas. Es indispensable, además, organizar el trabajo, y con los subproductos generados en cada etapa, y haciendo un seguimiento lógico a las actividades, lograr que la aplicación cumpla los objetivos que orientaron su creación para que su utilización sea exitosa.

Un proceso define *quién* está haciendo *qué*, *cuándo* y *cómo*; y su trabajo consiste en transformar los requisitos de un cliente en un producto software.

[4] **PRESSMAN**, Roger S. Ingeniería del software. Un enfoque práctico. Editorial Mc. Graw Hill. Cuarta edición. España, 1998.

Existen diferentes modelos de proceso para la ingeniería del software, también llamados "Ciclo de vida de desarrollo software", cada uno representa un intento por ordenar una actividad inherentemente caótica y han sido caracterizados de tal forma que ayuden al control y coordinación de un proyecto software real. Como estrategia de desarrollo se debe seleccionar un modelo de proceso según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y las entregas que se requieren.

Después de analizar cada uno de los modelos de ciclos de vida del desarrollo de software y teniendo una visión de sus características comunes, se determinó que el paradigma apropiado para el desarrollo de este proyecto era el Prototipado Evolutivo. Los demás se descartaron porque presentaban insuficiencia en algunas de las siguientes características: trabajar con poca identificación de los requerimientos, generar un sistema con amplio desarrollo, permitir modificaciones durante el desarrollo y ofrecer a los clientes signos visibles de progreso.

La elección del Prototipado Evolutivo se fundamentó en las ventajas que tiene sobre los demás modelos, y que se evidencian en las características mencionadas, así mismo en que: se tiene contacto con el cliente a lo largo del desarrollo, constantemente se realizan pruebas para la detección de errores y de este modo es posible mejorar el prototipo y lograr que el producto final cumpla con los requerimientos del cliente.

Puesto que durante el proceso de desarrollo se necesita tener una perspectiva del sistema y el artefacto más común para representarla es

el modelo, de manera adicional se construyeron modelos representados por diagramas, los cuales son manejados por el lenguaje UML.

3.2.1.1. Prototipado Evolutivo

El Prototipado Evolutivo es un modelo que toma sus bases del prototipado simple (ver Figura 1), pero posee mayores controles sobre la calidad y desarrolla primero las áreas de mayor riesgo del sistema, de tal forma que el prototipo se convierte en el producto final, una vez se llegue a su fin. Diciendo con esto que el concepto del sistema se desarrolla a medida que avanza el proyecto. Sus etapas consisten en: concepto inicial, diseño e implementación del prototipo inicial, iteración para refinar el prototipo hasta que sea aceptable y completar y entregar el prototipo (ver Figura 2).

Figura 1. Modelo de Prototipado Simple.[5]

[5] **GOMEZ** Flórez, Luis Carlos. Proyectos Informáticos. Notas de Clase. UIS. 2001 – 2003.

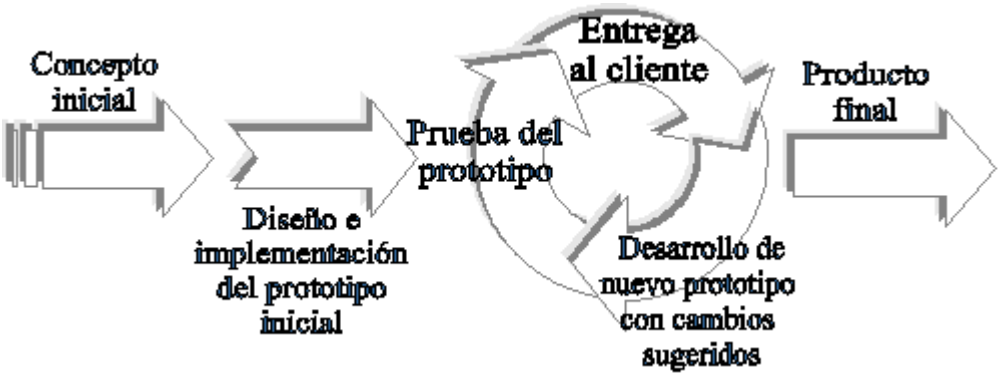

Figura 2. Modelo de Prototipado Evolutivo.[6]

En la primera etapa del Prototipado Evolutivo, concepto inicial, se hace la recolección de la información necesaria para desarrollar el proyecto, también un estudio de las herramientas de desarrollo, así como la recopilación y el análisis de requerimientos. En la segunda etapa, diseño e implementación del prototipo inicial, se procede a la elaboración de diagramas mediante UML, diseño de la interfaz gráfica de usuario y construcción del prototipo inicial. En la tercera etapa ó fase iterativa, se realizan pruebas para detectar posibles errores de implementación, se hace entrega al cliente quien da las sugerencias necesarias relacionadas con el logro de los objetivos del sistema. Teniendo en cuenta las observaciones de los usuarios se desarrollan nuevos prototipos, hasta que se llegue a uno que sea aceptado como producto final.

[6] Modificado a partir de modelo encontrado en:
GOMEZ Flórez, Luis Carlos. Proyectos Informáticos. Notas de Clase. UIS. 2001 – 2003.

3.2.1.2. Lenguaje Unificado de Modelamiento (UML)

Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. UML pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar[7].

UML está compuesto por diversos diagramas: de Clases, de Casos de Uso, de Componentes, de Despliegue, de Estados, de Actividad, de Secuencia y de Colaboración. UML permite definir solo los diagramas necesarios, ya que no todos son indispensables en todos los proyectos, por lo cual se escogieron tres (Casos de Uso, Actividad y Secuencia) como ayuda para el desarrollo del proyecto.

Diagramas de Casos de Uso: representan la forma como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden como los elementos interactúan (operaciones o casos de uso). Son muy importantes para modelar y organizar el comportamiento del sistema.

Diagramas de Actividades: pueden dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos. Se

[7] **LARMAN**, Craig. UML y patrones. Introducción al análisis y diseño orientado a objetos. Editorial Prentice Hall. Primera edición. México, 1999.

utilizan para modelar el funcionamiento del sistema y el flujo de control entre objetos.

Diagramas de Secuencia: forman parte del modelado dinámico del sistema. Un diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian, ordenados según su secuencia en el tiempo.

3.2.2. Lenguajes para el Desarrollo

3.2.2.1. Java

Java es un lenguaje de programación con el que podemos realizar cualquier tipo de programa. En la actualidad su uso es muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet, como en la informática en general.

Una de las principales características por las que Java se ha hecho famoso es que es un lenguaje independiente de la plataforma, lo que quiere decir que si hacemos un programa en Java podrá funcionar sobre cualquier sistema operativo.

Java cuenta con grupos de clases que pueden ser fácilmente adaptadas a las necesidades del desarrollador de software. Estas clases se pueden incluir en los programas Java, sin temor a fallos de portabilidad.

Además, están bien documentadas (mediante páginas Web), y organizadas en paquetes y en un gran árbol de herencia. A estos grupos de clases se les conoce comúnmente como APIs (Application Programming Interface), y los hay para diferentes propósitos, que van desde la implementación de una interfaz grafica para el usuario, como seria Swing, hasta la conectividad de bases de datos. También proporciona una colección de clases para su uso en aplicaciones de red, que permiten trabajo en red con TCP/IP, WWW y HTML, programas distribuidos y acceso a bases de datos.

A partir de la explosión de Internet en 1994, se empezó a hablar de Java y de sus aplicaciones, conocidas como applets. Un applet es un mini programa que corre solamente bajo un navegador y es descargado automáticamente como parte de una página Web, al igual que cualquier gráfico; cuando se activa, ejecuta un programa[8].

3.2.2.2. Dreamweaver

Dreamweaver es la opción profesional para la creación de sitios y aplicaciones Web. Proporciona una combinación potente de herramientas visuales de disposición, características de desarrollo de aplicaciones y soporte para la edición de código.

Gracias a las robustas características para la integración y el diseño, Dreamweaver permite que los diseñadores y desarrolladores Web creen

[8] **CEBALLOS**, Francisco Javier. Java 2. Curso de programación. Editorial Alfaomega. México, 2000.

y manejen cualquier sitio Web con toda facilidad. Cumple perfectamente el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, además de ser muy fáciles de usar.

3.3. INTELIGENCIA ARTIFICIAL (IA)

Dentro del campo de la Informática una de las áreas que mas ha hecho evolucionar los problemas que se pueden resolver por la utilización de las computadoras ha sido la IA. El objetivo de la IA consiste en la construcción de sistemas, tanto hardware como software, que sean capaces de replicar aspectos de lo que se suele considerar "inteligencia". Evidentemente este objetivo está muy ligado a la definición de la palabra "inteligencia". Esto nos obliga a adoptar un punto de vista práctico y definir la IA como el conjunto de técnicas, métodos, herramientas y metodologías, que nos ayudan a construir sistemas que se comportan igual que un humano en la resolución de problemas concretos[9].

Según esta definición la IA involucra muchos campos de investigación y desarrollo diferentes, tales como la Robótica, la Visión Artificial, la Resolución de Problemas, los Sistemas Expertos, la Traducción Automática, etc.

En los desarrollos de IA, la búsqueda es un proceso de gran importancia en la resolución de problemas difíciles para los que no se dispone de

[9] **ALER**, Ricardo. **BORRAJO**, Daniel. **SILVA**, Andrés. Unidad didáctica: Inteligencia Artificial. Archivo pdf.

técnicas más directas. Los procesos de búsqueda están cercanamente relacionados con los procesos de optimización.

3.3.1. Algoritmos de Búsqueda

Las técnicas de solución de problemas en IA, en general, incorporan un proceso de búsqueda. Todo proceso de búsqueda puede ser visualizado como el recorrido por un árbol en el que cada nodo representa un estado y cada rama representa las relaciones entre los estados cuyos nodos conecta.

Hay una gran variedad de métodos de búsqueda. El más trivial y demorado es el de búsqueda del óptimo por fuerza bruta (revisar sistemáticamente todo el espacio de problema) y los más sofisticados, más breves, apelan a matemáticas que pueden ser tan complicadas como el autor quiera.

En inteligencia artificial el tema de búsquedas es central, dado que, por ejemplo, realizar acciones mecanizadas o resolver problemas, se reduce a buscar en un espacio de estados. En esa disciplina se estudian búsquedas sin información (o ciegas), como Búsqueda en Amplitud, en Profundidad Limitada, Profundidad con Retroceso, etc. y búsquedas con información (o inteligentes), como Búsqueda en Escalada Irrevocable, Primero el Mejor, A*, Mínimax, Poda Alfa – Beta, etc.

Búsqueda en Amplitud: este algoritmo visita cada nodo del árbol por niveles, es decir, visita todos los nodos de un nivel antes de visitar los del siguiente. Para cada nodo visitado genera todos sus sucesores y los guarda en una estructura tipo cola FIFO.

Búsqueda en Profundidad Limitada: este algoritmo continúa por una rama del árbol hasta decidir terminar la búsqueda por esa dirección, ya sea porque llegó al estado final o porque superó un límite de profundidad determinado. Cuando fracasa una ruta se devuelve, continuando la exploración en el paso inmediatamente anterior. Para cada nodo visitado genera todos sus sucesores y los guarda en una estructura tipo pila LIFO.

Búsqueda en Profundidad con Retroceso: Este algoritmo hace el mismo recorrido que la Búsqueda en Profundidad Limitada, diferenciándose en que para cada nodo sólo genera un sucesor.

Búsqueda en Escalada Irrevocable: Los métodos de escalada (o de ascensión a la colina) trabajan, en su versión más habitual, con funciones de evaluación en las que los valores superiores son preferibles. De ahí su nombre: se trata de elegir en cada paso un estado cuyo valor heurístico sea mayor que el del estado activo en ese momento. Como utiliza estrategia irrevocable cada estado genera a lo más un nuevo estado.

Búsqueda Primero el Mejor: Consiste en recorrer el grafo de búsqueda eligiendo en cada momento el nodo que tenga mejor valor para una determinada función heurística. A diferencia del método del gradiente,

cuando el camino actual se aleja de la meta, se pueden retomar caminos de exploración abandonados anteriormente.

Búsqueda A*: Se usa para juegos simples unipersonales. Este algoritmo implementa una búsqueda primero el mejor, utilizando una función de evaluación estática f = g + h; donde g es una función de coste de llegar del estado inicial al estado evaluado y h es una estimación del coste de llegar desde el estado evaluado al estado final u objetivo del juego.

Búsqueda Minimax: Se usa en juegos bipersonales. El nombre del algoritmo deriva de considerar que, dada una función estática que devuelve valores con relación al jugador maximizante, éste procura maximizar su valor mientras que su oponente procura minimizarlo. En un árbol de juego donde los valores de la función estática están con relación al jugador maximizante, se maximiza y minimiza alternadamente de un nivel a otro; y viceversa si están con relación al jugador minimizante.

Búsqueda Poda Alfa – Beta: La estrategia de poda del algoritmo Minimax es llamada Poda Alfa - Beta, puesto que dado que existen dos jugadores, existen dos valores umbrales Alfa y Beta para acotar la búsqueda de cada uno respectivamente: el valor Alfa representa la cota inferior del valor que puede asignarse en último término a un nodo maximizante y el valor Beta representa la cota superior del valor que puede asignarse en último término a un nodo minimizante.

Las principales diferencias que pueden aparecer en las diferentes técnicas de búsqueda, son:

- La dirección en la cual se conduce la búsqueda (hacia adelante o hacia atrás).
- La estrategia de control, o forma de seleccionar las reglas que pueden ser aplicables.

4. DISEÑO Y CONSTRUCCIÓN DE SABIA

FASE INICIAL

4.1. CONCEPTO INICIAL

4.1.1. Recolección de la Información

En esta etapa, como primer paso para el desarrollo de SABIA, se realizó una entrevista con la directora del proyecto, quien mostró su punto de vista sobre un problema que se estaba presentando en la asignatura de Inteligencia Artificial, debido a la falta de comprensión por parte de los estudiantes, de algunos conceptos relacionados con el tema de algoritmos de búsqueda.

Después de conocer el problema y decidir que SABIA contribuiría a su solución, se procedió a una búsqueda de información inicial relacionada con los temas que incluiría el proyecto. A medida que se iba desarrollando el proyecto surgía la necesidad de hacer nuevas búsquedas de información, tanto para el contenido como para la realización de la herramienta. Toda esta información fue recopilada de diversas fuentes como libros, artículos, Internet y proyectos de grado.

4.1.2. Recopilación y Análisis de Requerimientos

Los requerimientos son las características deseables que tendrá el producto. Teniendo en cuenta el planteamiento del problema y la información preliminar se definieron los requerimientos relacionados con el contenido, el entorno y el lenguaje de programación que tendría la herramienta.

Para los contenidos se decidió que la herramienta llevaría teoría, simulaciones, ejemplos, ejercicios y juegos, de cada uno de los algoritmos de búsqueda. Los algoritmos elegidos fueron: Amplitud, Profundidad limitada, Profundidad con retroceso, Búsqueda general en grafos, Escalonada irrevocable, Primero el mejor, A*, Poda α – ß y Mínimax.

Con respecto al entorno de programación, se había establecido previamente utilizar un entorno virtual ya existente como es el caso del "Aula Virtu@l"[10].

[10] http://tic.uis.edu.co/aula

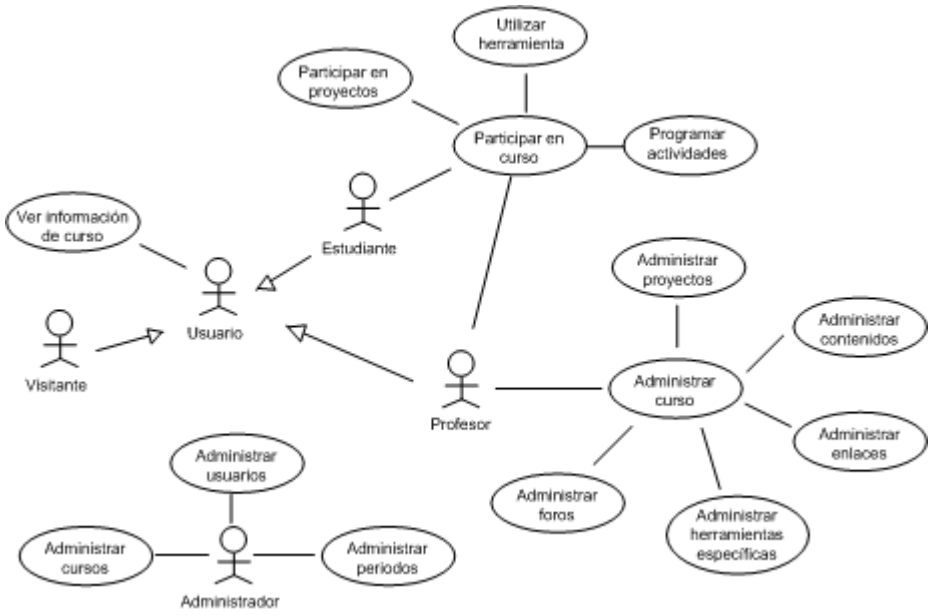

Figura 3. Diagrama de casos de uso generales del "Aula Virtu@l".

Debido a que SABIA sería desarrollada para un ambiente virtual de aprendizaje, se acordó Java como el lenguaje de programación para el desarrollo de ciertos applets que incluiría la herramienta. Por otra parte se decidió trabajar con Dreamweaver para la elaboración de páginas web en las cuales irían embebidos gran parte de los applets.

4.1.3. Conocimiento de Herramientas

Se inicio con el estudio de la parte básica del lenguaje Java (j2sdk), para después hacer énfasis en la parte que realmente interesaba, la creación de *applets* y el manejo de los *threads*. Luego se alternó con el estudio de UML y finalmente se trabajó con Dreamweaver.

4.2. DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO INICIAL

4.2.1. Elaboración de Diagramas de Casos de Uso, de Secuencia y de Actividades

En primer lugar se desarrolló el diagrama de casos de uso general (ver Figura 4), para mostrar cómo se deseaba que trabajara el sistema.

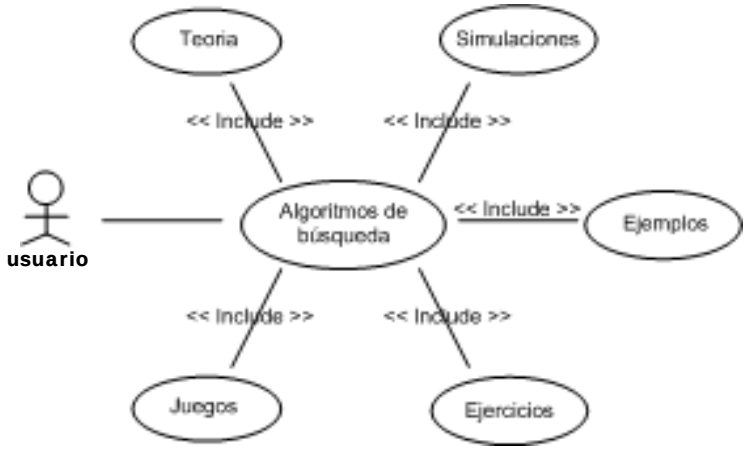

Figura 4. Diagrama de casos de uso general.

Después se crearon diagramas de secuencias (ver Figuras 5, 6, 8, 9, 11 y 12) de los algoritmos búsqueda en amplitud, búsqueda en profundidad limitada y búsqueda en profundidad con retroceso; para observar la perspectiva cronológica de las interacciones y la secuencia explícita de mensajes entre objetos. También diagramas de actividades de estos mismos algoritmos (ver Figuras 7, 10 y 13), para entender el comportamiento de la ejecución del algoritmo, sin profundizar en los detalles internos de los mensajes.

La nomenclatura que se tiene en cuenta para los diagramas de secuencias y de actividades es la siguiente:

 Usuario

 Interfaz

 Sistema

 Árbol

[ ] Marcador de condición

*[ ] Marcador de Iteración

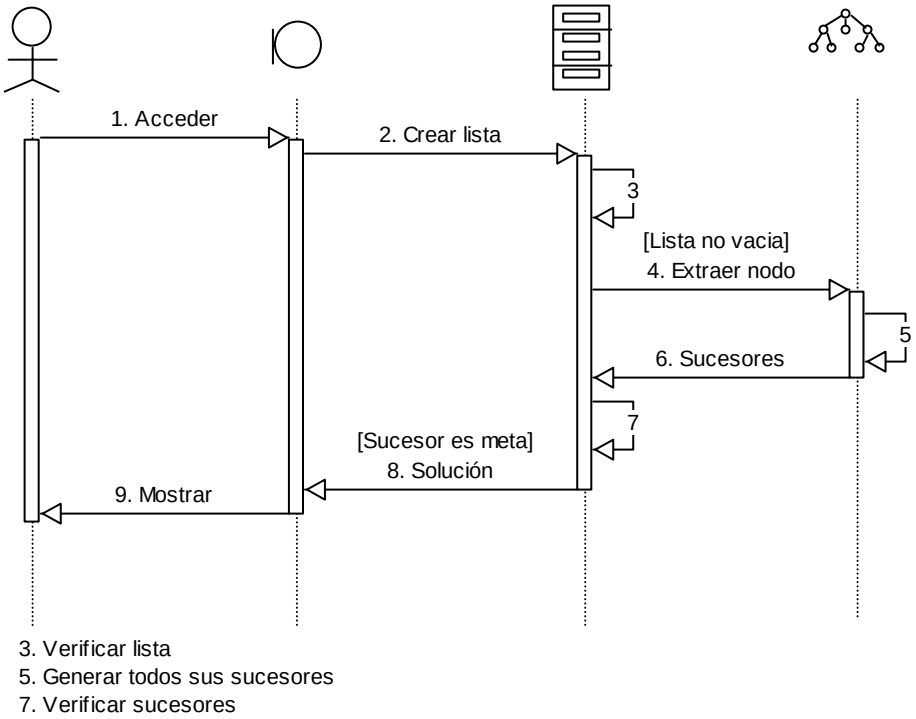

Figura 5. Diagrama de secuencias: búsqueda en amplitud cuando primer sucesor es meta.

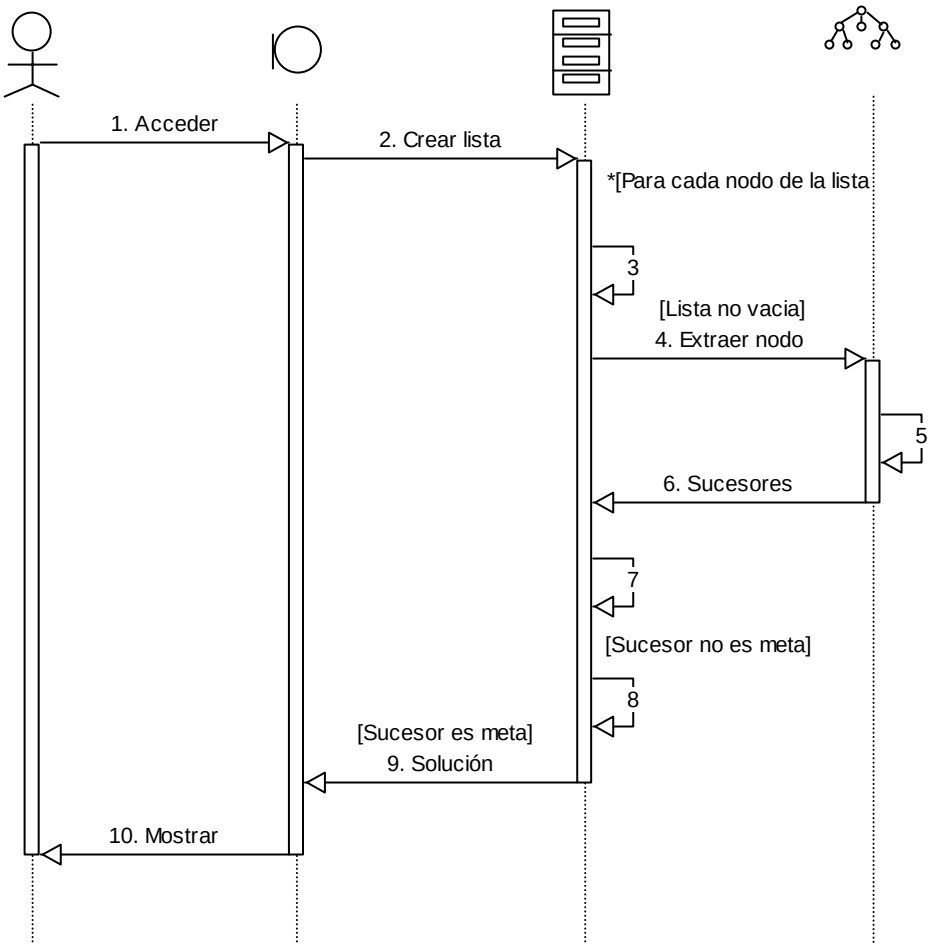

Figura 6. Diagrama de secuencias: búsqueda en amplitud cuando primer sucesor no es meta.

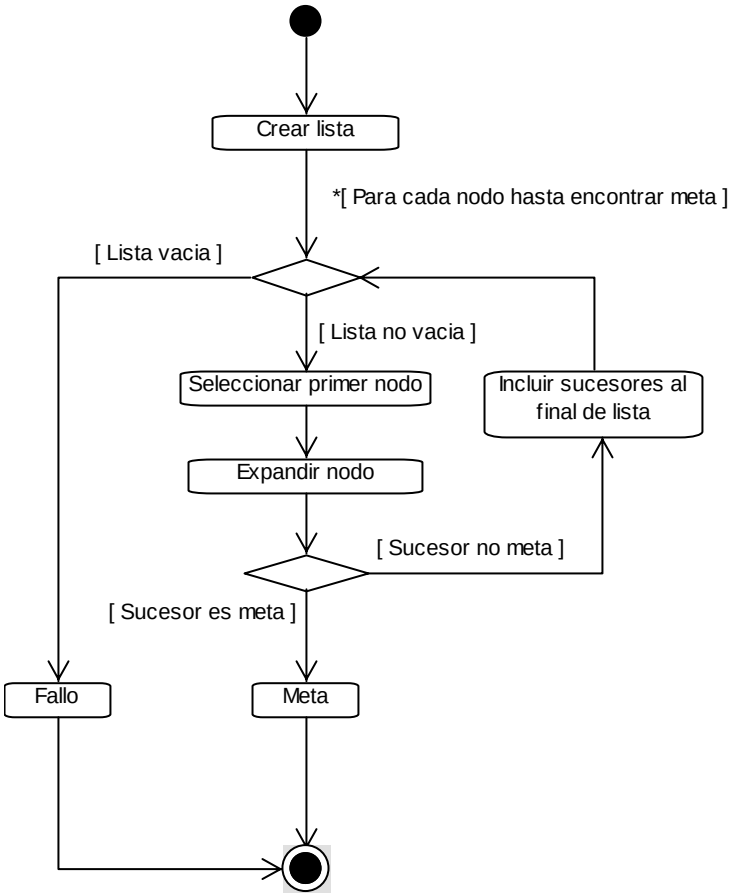

Figura 7. Diagrama de actividades: búsqueda en amplitud.

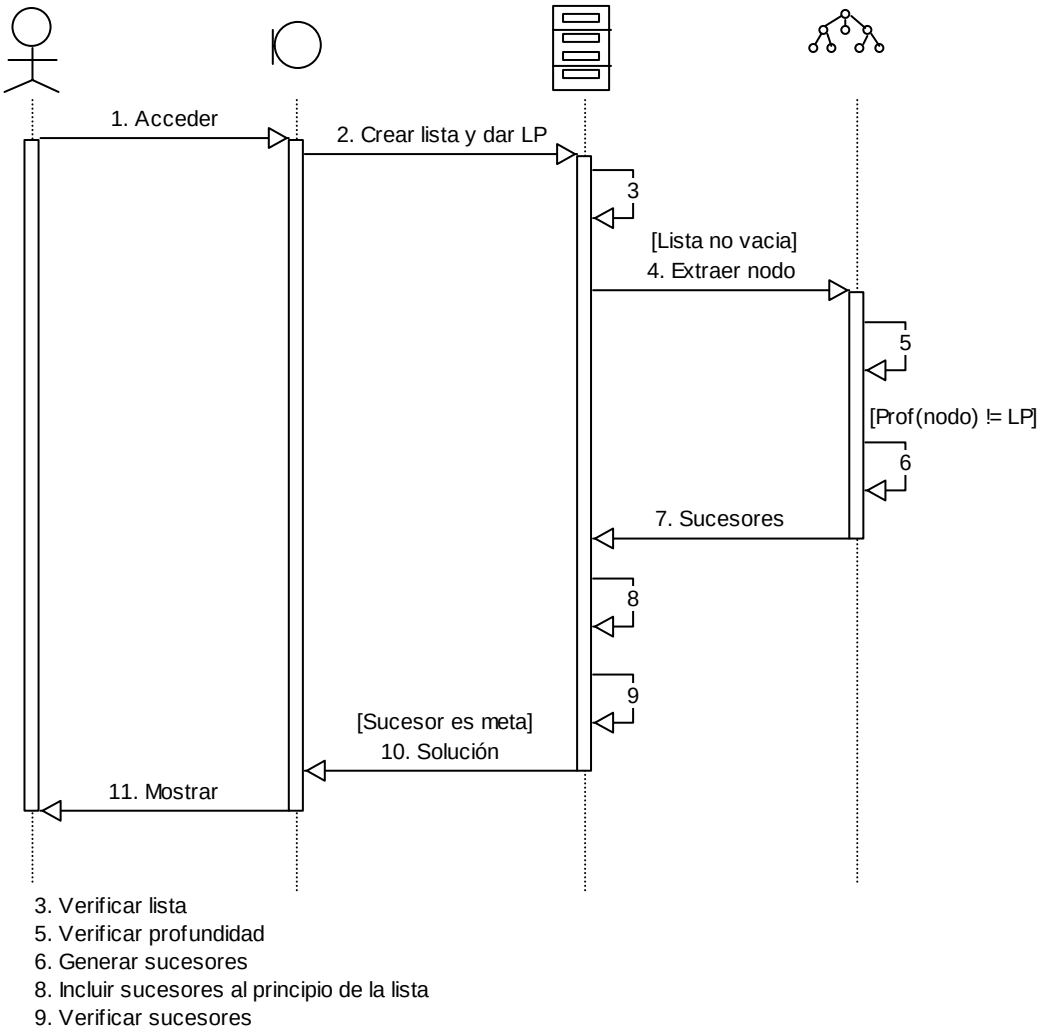

Figura 8. Diagrama de secuencias: búsqueda en profundidad limitada cuando primer sucesor es meta.

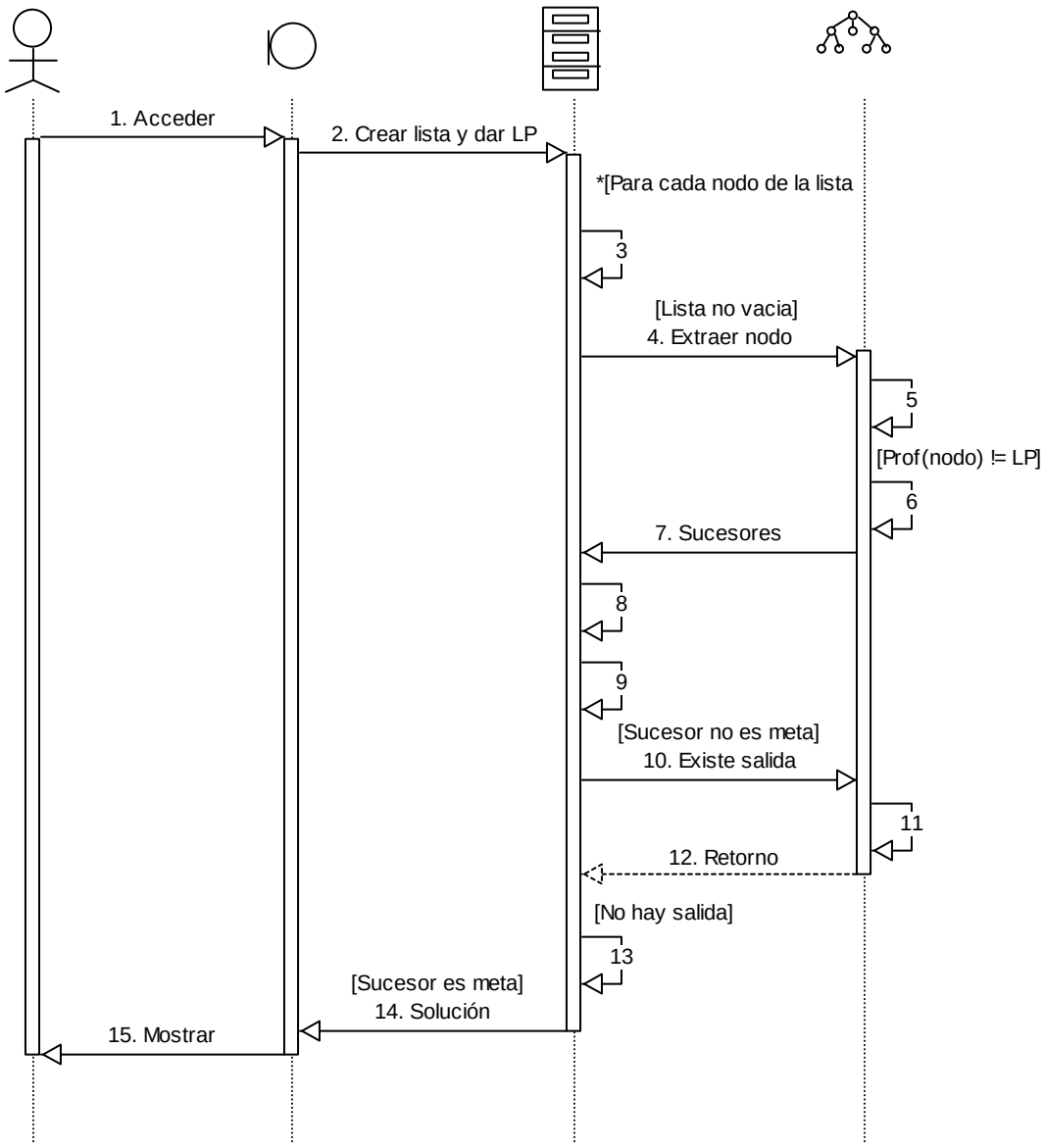

Figura 9. Diagrama de secuencias: búsqueda en profundidad limitada cuando primer sucesor no es meta.

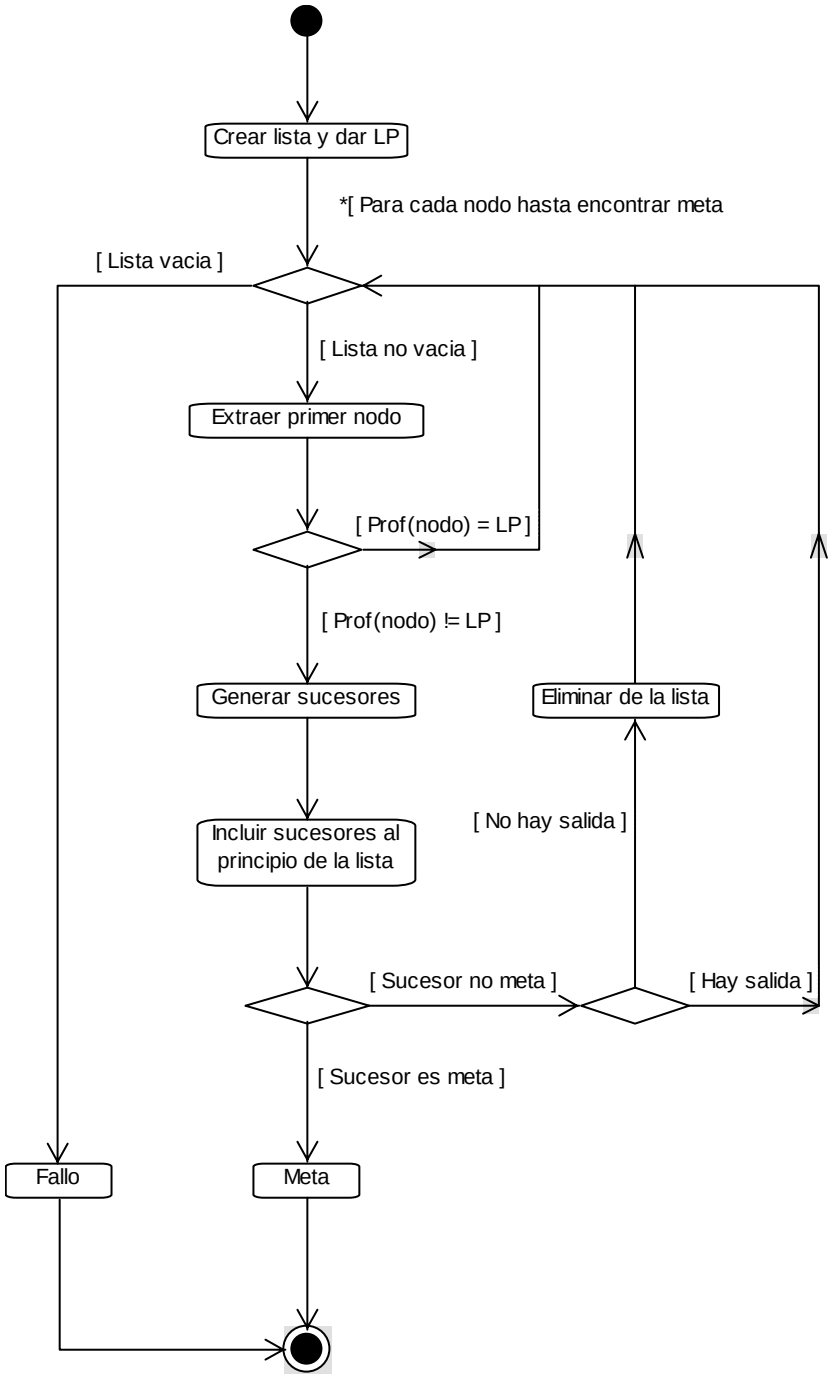

Figura 10. Diagrama de actividades: búsqueda en profundidad limitada.

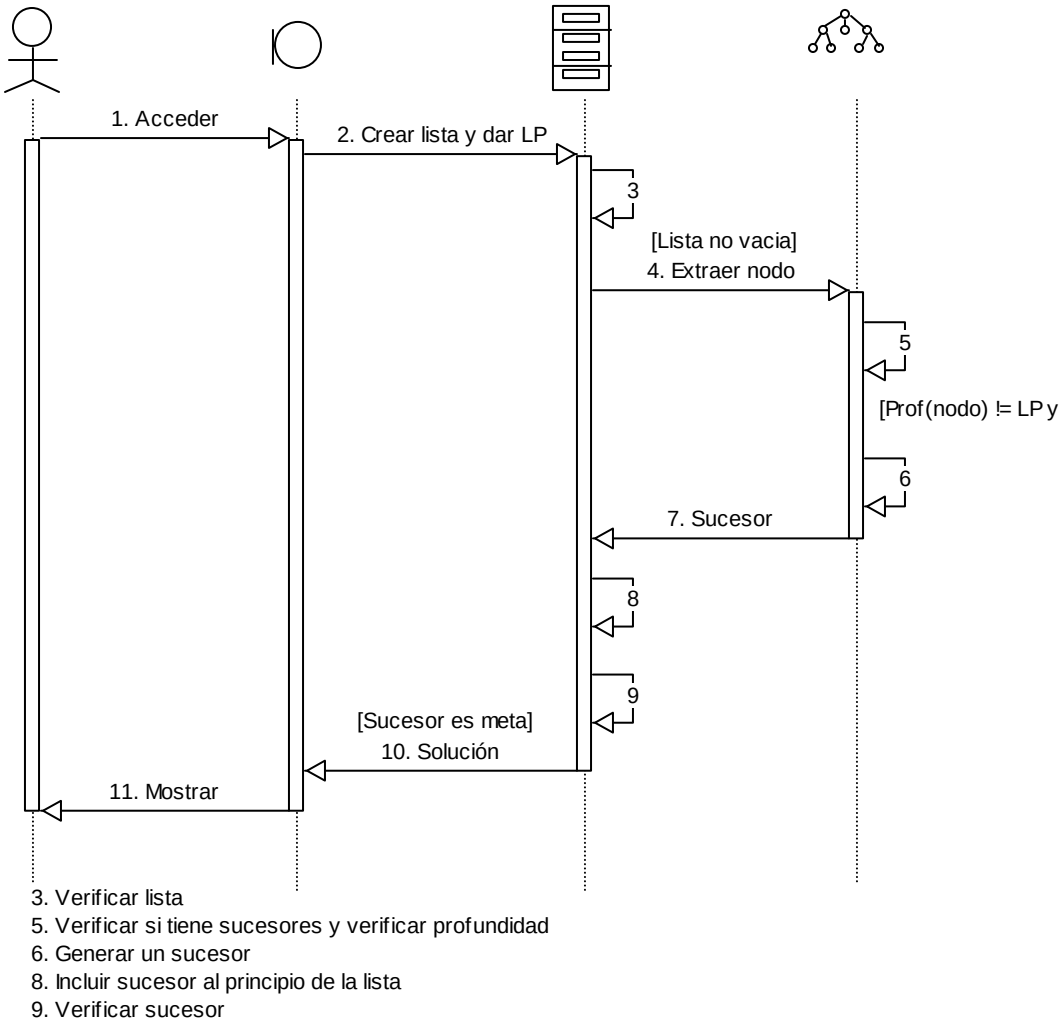

Figura 11. Diagrama de secuencias: búsqueda en profundidad con retroceso cuando primer sucesor es meta.

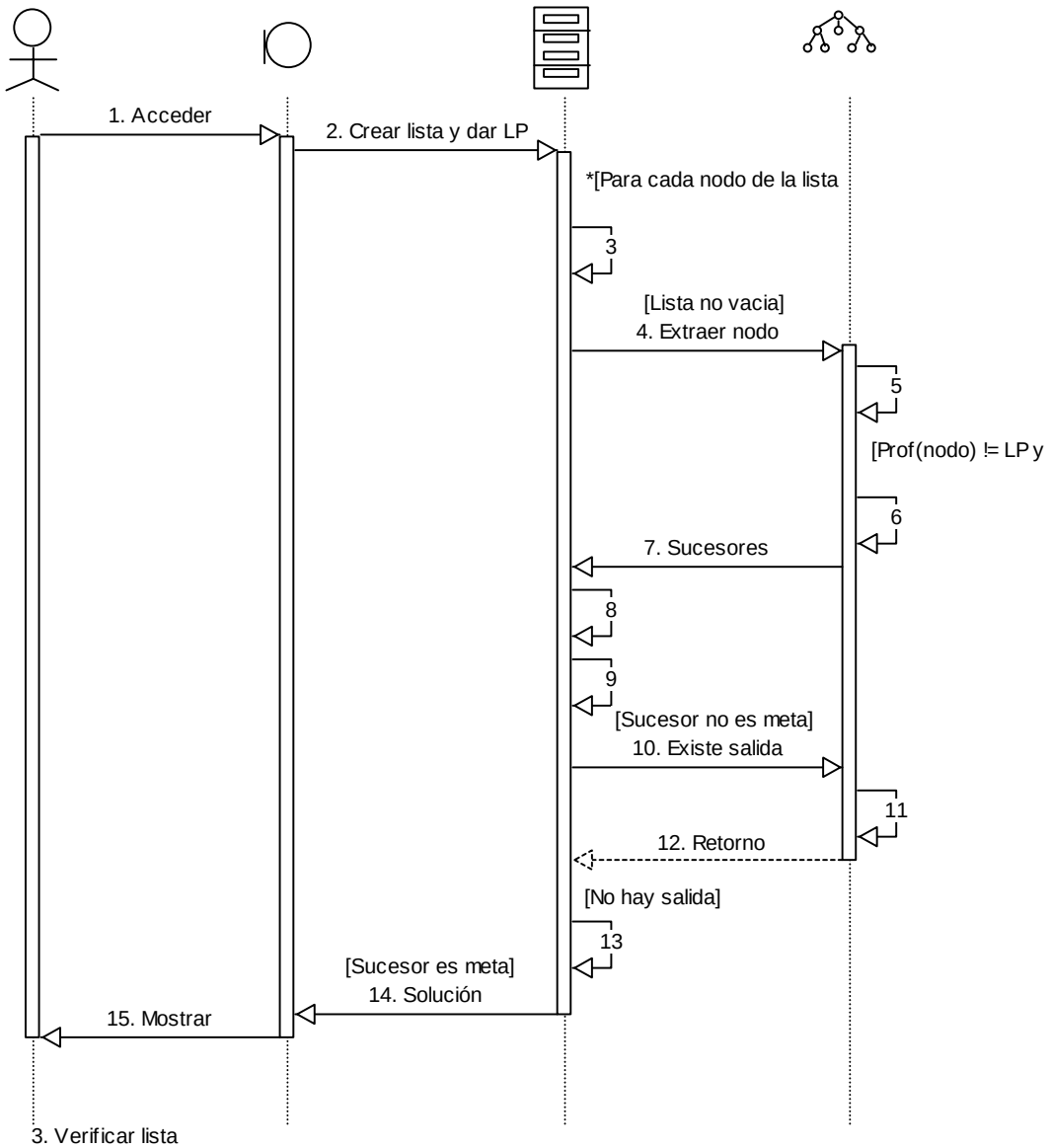

Figura 12. Diagrama de secuencias: búsqueda en profundidad con retroceso cuando primer sucesor no es meta.

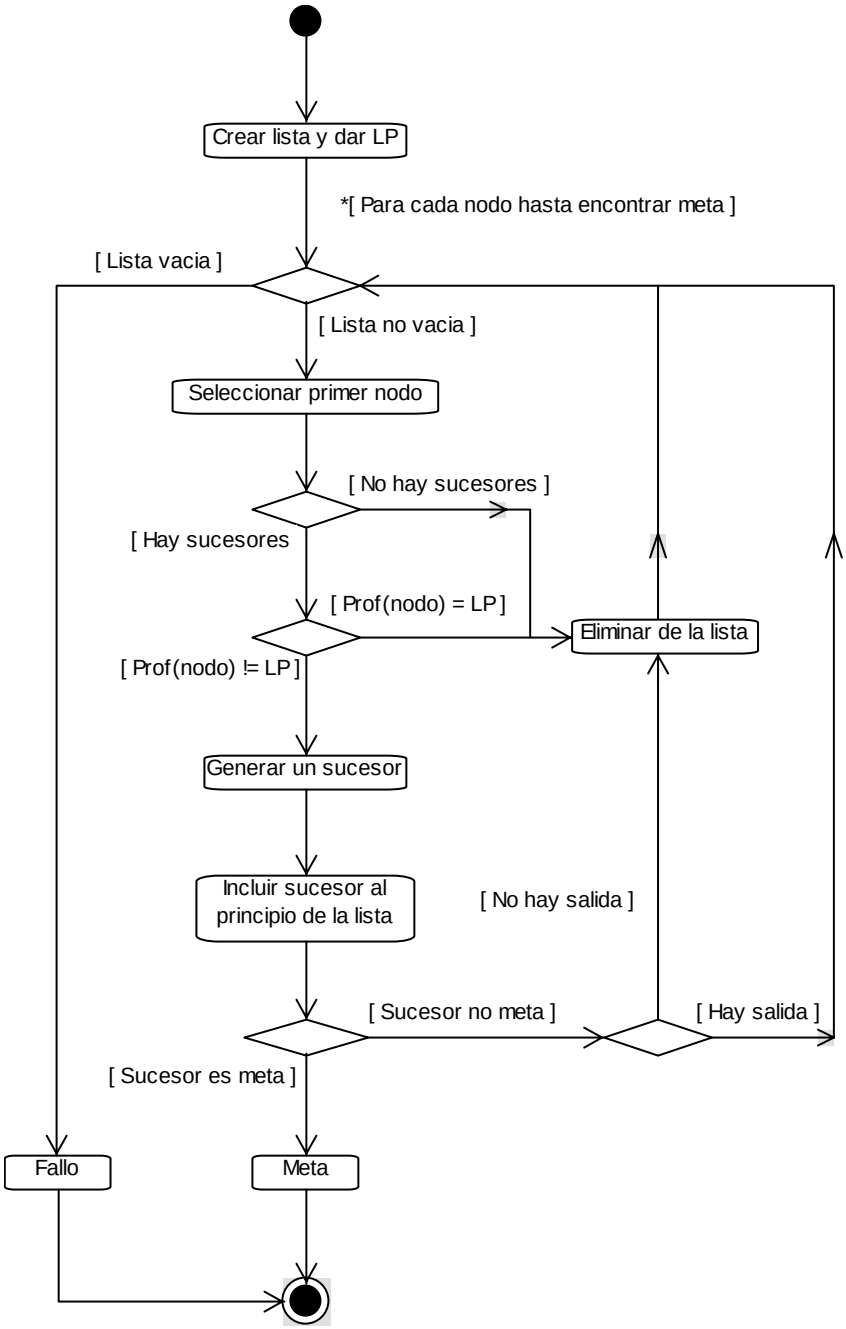

Figura 13. Diagrama de actividades: búsqueda en profundidad con retroceso.

4.2.2. Diseño de la Interfaz

En la Figura 14 se observa la interfaz gráfica del "Aula Virtu@l" donde se encuentran una serie de íconos dentro de los cuales está el Icono Herramientas. Al hacer clic sobre este ícono se abre una pequeña ventana desde donde se puede acceder a las distintas herramientas que ofrece cada curso. En el curso Inteligencia Artificial, en la ventana de herramientas se encontrará un enlace a SABIA.

Se hicieron dos bosquejos de la interfaz gráfica de usuario, una inicial (ver Figura 15) donde el usuario encuentra el respectivo logo y nombre de la herramienta, logotipos de las entidades interesadas y responsables de su desarrollo, así como un enlace que invita al usuario a explorar la herramienta. La segunda interfaz (ver Figura 16) posee los elementos necesarios, dispuestos de tal forma, que dan al usuario la facilidad y comodidad para la exploración de los contenidos.

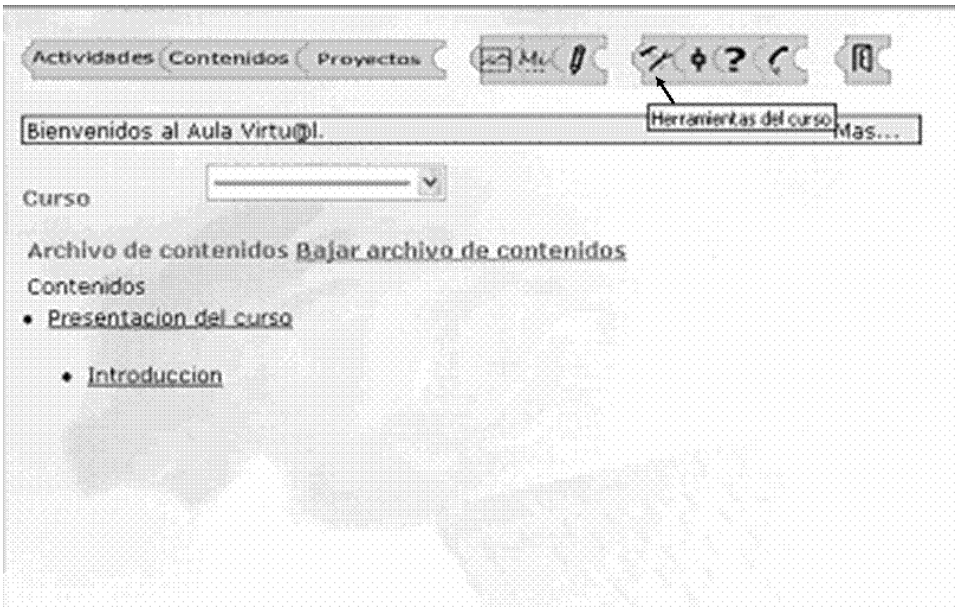

Figura 14. Interfaz gráfica de "Aula Virtu@l".

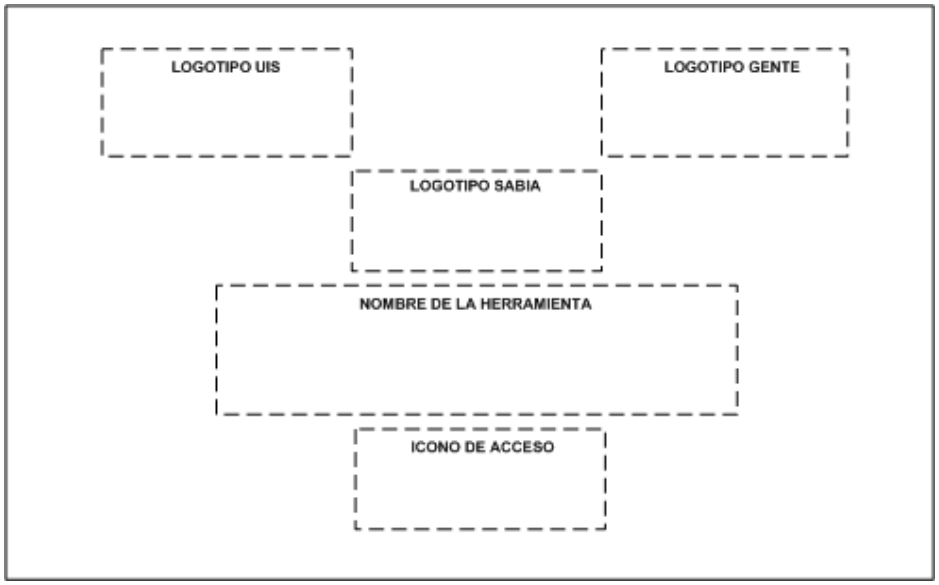

Figura 15. Interfaz gráfica de usuario inicial.

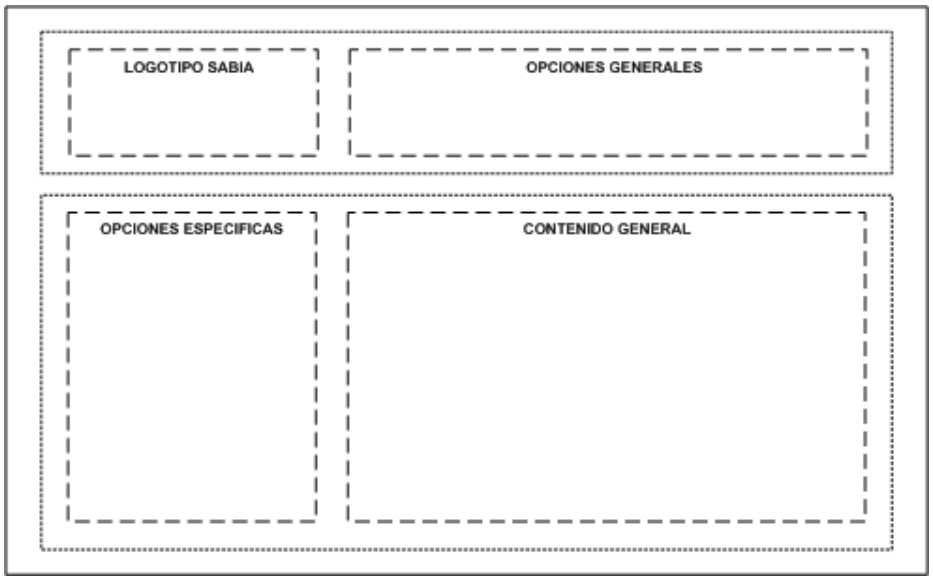

Figura 16. Interfaz gráfica de usuario general.

4.2.3. Construcción del Prototipo Inicial

La información recopilada se clasificó y se agrupó de acuerdo con los temas que llevaría la herramienta, luego se realizó el proceso de extraer de cada grupo la parte relevante. Para afianzar el aprendizaje de los contenidos fundamentales, así como el desarrollo de las habilidades, se elaboraron diagramas, y se incluyeron ejercicios y ejemplos que sirven para aclarar los conceptos expuestos y para fijarlos mediante su aplicación en casos concretos. Igual propósito tienen las simulaciones, las cuales son applets.

Teniendo en cuenta el diseño de la interfaz gráfica de usuario, se crearon páginas HTML donde se ubicaron los contenidos.

Haciendo referencia al "Aula Virtu@l" y a uno de sus requisitos, el cual especifica que las herramientas montadas necesariamente tienen que ser applets, se procedió a la elaboración de un applet que es el responsable de permitir el acceso a SABIA.

4.2.3.1. Código de los Applets

ACCESO A SABIA

Applet

```
import java.awt.*;
import java.applet.*;
import java.net.*;

public class Sabia extends Applet {
        String directorio;

        public void init() {
                directorio = getParameter("PATH");
           try {
               URL URLSabia = new URL(directorio + "/" + "index.htm");
                   getAppletContext().showDocument(URLSabia);
                } catch (MalformedURLException mue) {
} } }
```

Página HTML

```
<HTML><BODY>
<APPLET CODE = Sabia.class width="500" height="250">
<PARAM NAME = PATH VALUE = "http://tic.uis.edu.co/aula/tools/sabia">
</APPLET>
</BODY></HTML>
```

SIMULACIONES

Búsqueda

```
import java.awt.*;
import java.awt.event.*;
import java.util.Vector;

abstract public class SearchApplet extends java.applet.Applet {

    public void init() {
        String width = getParameter("pwidth");
        if (width == null) {
            xSizePixels = 80;
        } else {
            xSizePixels = Integer.valueOf(width).intValue();
        }
        String height = getParameter("pheight");
        if (height == null) {
            ySizePixels = 80;
        } else {
            ySizePixels = Integer.valueOf(height).intValue();
        }
        System.out.println("width=" + xSizePixels + ", height=" + ySizePixels);
        setLayout(null);
        startChoice = new Choice();
        goalChoice  = new Choice();
        for (int i=0; i<numNodes; i++) {
            startChoice.add(nodeNames[i]);
            goalChoice.add(nodeNames[i]);
        }
        startChoice.select(0);
        goalChoice.select(numNodes - 1);
        Label l1 = new Label("Starting node:");
        Label l2 = new Label("Goal node:");
        l1.setBounds(5, 10, 90, 24);
        startChoice.setBounds(100, 10, 50, 30);
        l2.setBounds(5, 50, 90, 24);
        goalChoice.setBounds(100, 50, 50, 30);
        timeDelayChoice = new Choice();
        timeDelayChoice.add("No plot time delay");
        timeDelayChoice.add("Plot time delay");
        timeDelayChoice.setBounds(160, 10, 90, 24);
        timeDelayChoice.select(1);
        add(l1); add(l2); add(startChoice); add(goalChoice); add(timeDelayChoice);
        Button b = new Button("Run");
        b.setBounds(160, 50, 90, 32);
        add(b);
        b.addMouseListener(new java.awt.event.MouseAdapter()
```

```
                        {
                            public void mouseClicked(MouseEvent me) {
                                int i1 = startChoice.getSelectedIndex();
                                int i2 = goalChoice.getSelectedIndex();
                                startNodeIndex = i1;
                                goalNodeIndex = i2;
                                System.out.println("mouse clicked " + i1 + " " + i2);
                                findPath(i1, i2); }
                        }
                        );
        canvas = new Canvas();
        add(canvas);
        canvas.setBounds(5, 80, xSizePixels + 20, ySizePixels + 20);
        add(canvas);
        repaintPath(null, 0);
        setBackground(Color.white);
}
   protected Choice startChoice, goalChoice, timeDelayChoice;
   private Canvas canvas;
   private int goalNodeIndex = -1, startNodeIndex = -1;

   public void repaintPath(int [] path, int num) {
        System.out.print("repaintPath(");
        boolean time_delay = timeDelayChoice.getSelectedIndex() == 1;
        for (int i=0; i<num; i++) {
            System.out.print(path[i]);
            if (i < (num - 1)) System.out.print(", ");
        }
        System.out.println(")");
        Graphics g = canvas.getGraphics();
        g.setColor(Color.black);
        float x_scale = (float)xSizePixels / (x_max - x_min);
        float y_scale = (float)ySizePixels / (y_max - y_min);
        for (int i=0; i<numLinks; i++) {
            int i1 = link_1[i];
            int i2 = link_2[i];
            int x1 = (int)(node_x[i1] * x_scale);
            int x2 = (int)(node_x[i2] * x_scale);
            int y1 = (int)(node_y[i1] * y_scale);
            int y2 = (int)(node_y[i2] * y_scale);
            g.drawLine(x1 + 10, y1 + 10, x2 + 10, y2 + 10);
            g.drawLine(x1 + 10, y1 + 11, x2 + 10, y2 + 11);
        }
        for (int i=0; i<numNodes; i++) {
            int x1 = (int)(node_x[i] * x_scale);
            int y1 = (int)(node_y[i] * y_scale);
            if (i == startNodeIndex)     g.setColor(Color.green);
            else if (i == goalNodeIndex) g.setColor(Color.blue);
            else                         g.setColor(Color.black);
            paintNode(g, nodeNames[i], x1 + 10, y1 + 10);
        }
```

```
    if (path == null)  return;
    g.setColor(Color.red);
    for (int i=0; i<num - 1; i += 2) {
       int x1 = (int)(node_x[path[i]] * x_scale);
       int x2 = (int)(node_x[path[i+1]] * x_scale);
       int y1 = (int)(node_y[path[i]] * y_scale);
       int y2 = (int)(node_y[path[i+1]] * y_scale);
       g.drawLine(x1 + 10, y1 + 10, x2 + 10, y2 + 10);
       g.drawLine(x1 + 10, y1 + 11, x2 + 10, y2 + 11);
    }
    if (time_delay) {
       try {
            Thread.sleep(1000);
       } catch (Exception e) { }
    }  }

public void repaintPath(int [] path) {
    repaintPath(path, path.length);
}

protected void paintNode(Graphics g, String name, int x, int y) {
    int len = name.length() * 10 + 6;
    int x1 = x - (len / 2);
    int x2 = x + (len / 2);
    int y1 = y - 10;
    int y2 = y + 10;
    g.setColor(Color.cyan);
    g.fill3DRect(x1, y1, len, 20, true);
    g.setColor(Color.black);
    g.drawString(name, x1 + 4, y2 - 6);
}

public void paint(Graphics g) {
    super.paint(g);
    repaintPath(null, 0);
}

public void addNode(String name, float x, float y) {
    System.out.println("Adding node: " + name + ", " + x + ", " + y);
    nodeNames[numNodes] = name;
    node_x[numNodes] = x;
    node_y[numNodes] = y;
    numNodes++;
    if (x < x_min) x_min = x;
    if (x > x_max) x_max = x;
    if (y < y_min) y_min = y;
    if (y > y_max) y_max = y;
}

public String getNodeName(int index) {
    try {
```

```
            return nodeNames[index];
        } catch (Exception e) {
            System.out.println("Error in getNodeName: " + e);
        }
        return "no name";
    }

    public float getNodeX(int index) {
        try {
            return node_x[index];
        } catch (Exception e) {
            System.out.println("Error in getNodePosition: " + e);
        }
        return 0.0f;
    }

    public float getNodeY(int index) {
        try {
            return node_y[index];
        } catch (Exception e) {
            System.out.println("Error in getNodePosition: " + e);
        }
        return 0.0f;
    }

    public float getPathLength(int index) {
        return lengths[index];
    }
    public void addLink(int node1, int node2) {
        link_1[numLinks] = node1;
        link_2[numLinks] = node2;
        float dist_squared =
            (node_x[node1] - node_x[node2]) * (node_x[node1] - node_x[node2]) +
            (node_y[node1] - node_y[node2]) * (node_y[node1] - node_y[node2]);
        lengths[numLinks] = (float)Math.sqrt(dist_squared);
        numLinks++;
    }

    public void addLink(String name1, String name2) {
        int index1 = -1, index2 = -1;
        for (int i=0; i<numNodes; i++) {
            if (name1.equals(nodeNames[i])) index1 = i;
            if (name2.equals(nodeNames[i])) index2 = i;
        }
        if (index1 != -1 && index2 != -1)  addLink(index1, index2);
    }

    abstract public int [] findPath(int node_1, int node_2);

    protected int getNodeIndex(String name) {
        for (int i=0; i<numNodes; i++) {
```

```
            if (name.equals(nodeNames[i])) return i;
        }
        return -1;
    }

    final public static int MAX = 50;
    private String [] nodeNames = new String[MAX];
    private float [] node_x = new float[MAX];
    private float [] node_y = new float[MAX];
    protected int [] link_1 = new int[MAX];
    protected int [] link_2 = new int[MAX];
    private float [] lengths = new float[MAX];
    protected int numNodes = 0;
    protected int numLinks = 0;
    private int xSizePixels = 0, ySizePixels = 0;
    private float x_min = 0.0f, x_max = 0.1f;
    private float y_min = 0.0f, y_max = 0.1f;
}
```

Búsqueda en Amplitud

```
import java.util.Vector;

public class BreadthFirstSearch extends SearchApplet {

    public void init() {
        addNode("0", 0.0f, 0.0f);
        addNode("1", 1.0f, 1.0f);
        addNode("2", 5.0f, 2.0f);
        addNode("3", 2.0f, 5.0f);
        addNode("4", 7.0f, 5.0f);
        addNode("5", 8.0f, 8.0f);
        addNode("6", 10.0f, 5.0f);
        addNode("7", 8.0f, 2.0f);
        addNode("8", 12.0f, 8.0f);
        addNode("9", 13.0f, 5.0f);
        addLink(0,1);
        addLink(1,2);
        addLink(2,3);
        addLink(2,4);
        addLink(4,5);
        addLink(4,6);
        addLink(6,8);
        addLink(8,9);
        addLink(2,7);
        addLink(7,9);
        super.init();
    }
```

```
    public int [] findPath(int node_1, int node_2) {
        System.out.println("Entered BreadthFirstSearch.findPath(" +
                           node_1 + ", " + node_2 + ")");
        if (node_1 == node_2)  {
           repaintPath(null, 0);
           return null;
        }
        boolean [] visitedFlag = new boolean[numNodes];
        float [] distanceToNode = new float[numNodes];
        int [] predecessor = new int[numNodes];
        Queue queue = new Queue(numNodes + 2);
        int [] display_path = new int[2 * numNodes + 1];
        int num_display_path = 0;
        for (int i=0; i<numNodes; i++) {
           visitedFlag[i] = false;
           distanceToNode[i] = 10000000.0f;
           predecessor[i] = -1;
        }
        visitedFlag[node_1] = true;
        distanceToNode[node_1] = 0.0f;
        queue.enqueue(node_1);
outer:  while (queue.isEmpty() == false) {
           int head = queue.head();
           int [] connected = connected_nodes(head);
           if (connected != null) {
                for (int i=0; i<connected.length; i++) {
                   if (visitedFlag[connected[i]] == false) {
                        distanceToNode[connected[i]] = distanceToNode[head] + 1.0f;
                        predecessor[connected[i]] = head;
                        queue.enqueue(connected[i]);
                        display_path[num_display_path++] = head;
                        display_path[num_display_path++] = connected[i];
                        repaintPath(display_path, num_display_path);
                        if (connected[i] == node_2) break outer;
                   }
                }
                visitedFlag[head] = true;
                queue.dequeue();
           }
        }
        int [] ret = new int[numNodes + 1];
        int count = 0;
        ret[count++] = node_2;
        for (int i=0; i<numNodes; i++) {
           ret[count] = predecessor[ret[count - 1]];
           count++;
           if (ret[count - 1] == node_1)  break;
        }
        num_display_path = 0;
        int [] ret2 = new int[count];
        for (int i=0; i<count; i++) {
```

```
            ret2[i] = ret[count - 1 - i];
            if (i > 0) {
                display_path[num_display_path++] = ret2[i-1];
                display_path[num_display_path++] = ret2[i];
            }
        }
        repaintPath(display_path, num_display_path);
        return ret2;
    }

    protected class Queue {
        public Queue(int num) {
            queue = new int[num];
            head = tail = 0;
            len = num;
        }
        public Queue() {
            this(400);
        }
        private int [] queue;
        int tail, head, len;
        public void enqueue(int n) {
            queue[tail] = n;
            if (tail >= (len - 1)) {
                tail = 0;
            } else {
                tail++;
            }}
        public int dequeue() {
            int ret = queue[head];
            if (head >= (len - 1)) {
                head = 0;
            } else {
                head++;
            }
            return ret;
        }
        public boolean isEmpty() {
            return head == (tail + 1);
        }
        public int head() {
            return queue[head];
        }
    }

    protected int [] connected_nodes(int node) {
        int [] ret = new int[SearchApplet.MAX];
        int num = 0;
        for (int n=0; n<numNodes; n++) {
            boolean connected = false;
            for (int i=0; i<numLinks; i++) {
```

```
                if (link_1[i] == node) {
                    if (link_2[i] == n) {
                        connected = true;
                        break;
                }}
                if (link_2[i] == node) {
                    if (link_1[i] == n) {
                        connected = true;
                        break;
                  }}
            }
            if (connected) {
                ret[num++] = n;
            }}
        if (num == 0)  return null;
        int [] ret2 = new int[num];
        for (int i=0; i<num; i++) {
            ret2[i] = ret[i];
        }
        return ret2;
    }
    private int [] path = new int[SearchApplet.MAX];
    private int num_path = 0;

    private int [] copy_path(int [] path, int num_to_copy) {
        int [] ret = new int[SearchApplet.MAX];
        for (int i=0; i<num_to_copy; i++) {
            ret[i] = path[i];
        }
        return ret;
    } }
```

Búsqueda en Profundidad Limitada

```
import java.util.Vector;

public class DepthFirstSearch extends SearchApplet {

    public void init() {
        addNode("0", 0.0f, 0.0f);
        addNode("1", 1.0f, 1.0f);
        addNode("2", 5.0f, 2.0f);
        addNode("3", 2.0f, 5.0f);
        addNode("4", 7.0f, 5.0f);
        addNode("5", 8.0f, 8.0f);
        addNode("6", 10.0f, 5.0f);
        addNode("7", 8.0f, 2.0f);
        addNode("8", 12.0f, 8.0f);
```

```
        addNode("9", 13.0f, 5.0f);
        addLink(0,1);
        addLink(1,2);
        addLink(2,3);
        addLink(2,4);
        addLink(4,5);
        addLink(4,6);
        addLink(6,8);
        addLink(8,9);
        addLink(2,7);
        addLink(7,9);
        super.init();
    }

    public int [] findPath(int node_1, int node_2) {
        System.out.println("Entered DepthFirstSearch.findPath(" +
                           node_1 + ", " + node_2 + ")");
        num_path = 1;
        path[0] = node_1;
        return findPathHelper(path, 1, node_2);
    }
    public int [] findPathHelper(int [] path, int num_path, int goal_node) {
        System.out.println("Entered DepthFirstSearch.findPathHelper(...," +
                           num_path + ", " + goal_node + ")");
        System.out.println("ici1");
        repaintPath(path, num_path);
        if (goal_node == path[num_path - 1]) {
           int [] ret = new int[num_path];
           for (int i=0; i<num_path; i++) ret[i] = path[i];
           System.out.println("ici2");
           repaintPath(ret);
           return ret;
        }
        int [] new_nodes = connected_nodes(path, num_path);
        if (new_nodes != null) {
           for (int j=0; j<new_nodes.length; j++) {
                int [] new_path = copy_path(path, num_path);
                new_path[num_path] = new_nodes[j];
                int [] test = findPathHelper(new_path, num_path + 1, goal_node);
                if (test != null) {
                   if (test[test.length - 1] == goal_node) {
                        return test;
                   }
        } } }
        return null;
    }

    public void repaintPath(int [] path, int num) {
        int [] path2 = new int[2 * num + 1];
        int count = 0;
        for (int i=0; i<(num - 1); i++) {
```

```
            path2[count++] = path[i];
            path2[count++] = path[i+1];
        }
        super.repaintPath(path2, count);
    }

    protected int [] connected_nodes(int [] path, int num_path) {
        int [] ret = new int[SearchApplet.MAX];
        int num = 0;
        int last_node = path[num_path - 1];
        for (int n=0; n<numNodes; n++) {
            boolean keep = true;
            for (int i=0; i<num_path; i++) {
                if (n == path[i]) {
                    keep = false;
                    break;
             }   }
            boolean connected = false;
            if (keep) {
                for (int i=0; i<numLinks; i++) {
                    if (link_1[i] == last_node) {
                        if (link_2[i] == n) {
                            connected = true;
                            break;
                        }
                    }
                    if (link_2[i] == last_node) {
                        if (link_1[i] == n) {
                            connected = true;
                            break;
                    }  } }
                if (connected) {
                    ret[num++] = n;
                }
            } }
        if (num == 0)  return null;
        int [] ret2 = new int[num];
        for (int i=0; i<num; i++) {
            ret2[i] = ret[i];
        }
        return ret2;
    }
    private int [] path = new int[SearchApplet.MAX];
    private int num_path = 0;
    private int [] copy_path(int [] path, int num_to_copy) {
        int [] ret = new int[SearchApplet.MAX];
        for (int i=0; i<num_to_copy; i++) {
            ret[i] = path[i];
        }
        return ret;
    } }
```

FASE ITERATIVA

4.3. PRUEBA DEL PROTOTIPO

4.3.1. Detección de Errores

Disponiendo de un prototipo inicial se procedió a realizar las pruebas correspondientes. Inicialmente se montó la herramienta en el "Aula Virtu@l" y se accedió a esta sin inconvenientes, luego se hizo la prueba de consulta, obteniendo como resultado un único error en los applets de las simulaciones, los cuales presentaban problemas en el momento de minimizar o maximizar la página, pues como es sabido los applets se encuentran embebidos dentro de páginas HTML.

4.4. ENTREGA AL CLIENTE

4.4.1. Recopilación de Cambios Sugeridos

Se entregó el prototipo al cliente, en este caso la directora del proyecto, para que realizara las respectivas pruebas de desempeño, conociera el manejo de la interfaz y finalmente diera a conocer su opinión y sugerencias respecto al mismo.

La directora, quien está capacitada para evaluarlo y decidir si éste es adecuado para todos los usuarios (profesores y estudiantes de la asignatura Inteligencia Artificial), dio su visto bueno sobre la

herramienta, sugiriendo modificaciones con respecto al diseño gráfico de la interfaz y la inclusión de un tema en la parte teórica.

4.5. DESARROLLO NUEVO PROTOTIPO

4.5.1. Modificación de Diagramas de Casos de Uso, de Secuencia y de Actividades

No hubo modificaciones en los diagramas elaborados inicialmente, los cuales corresponden a los algoritmos de búsqueda en amplitud, en profundidad limitada y en profundidad con retroceso. Lo que se hizo en esta parte fue crear los diagramas faltantes (ver Figuras 17 - 34).

La nomenclatura que se tiene en cuenta para los diagramas de secuencias y de actividades es la siguiente:

 Usuario

 Interfaz

 Sistema

 Árbol

[ ] Marcador de condición

*[ ] Marcador de Iteración

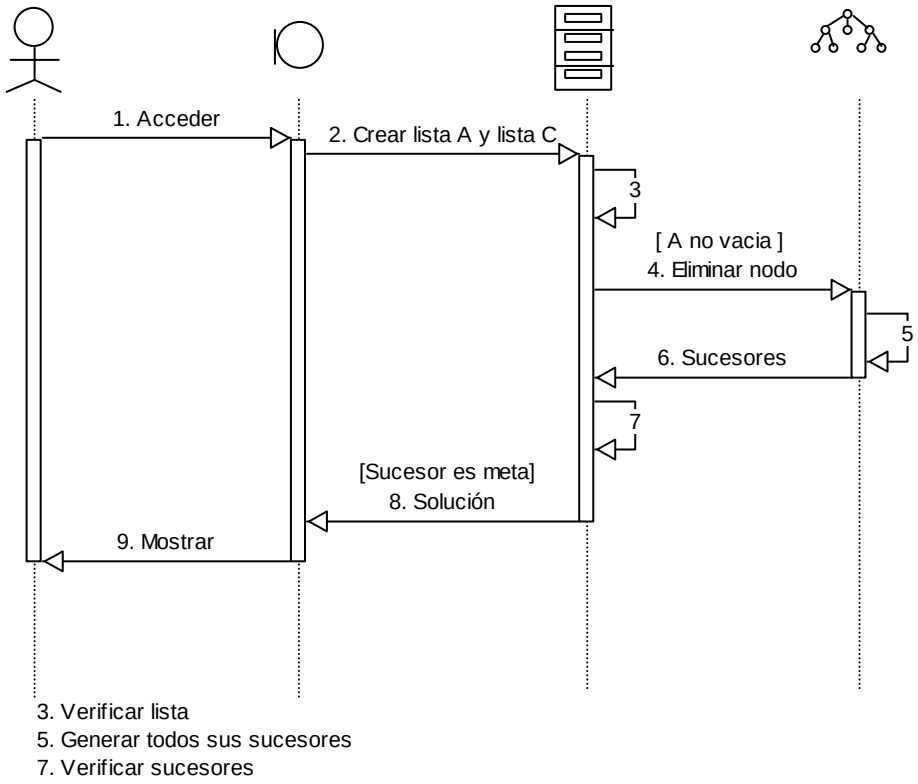

Figura 17. Diagrama de secuencias: búsqueda general en grafos cuando primer sucesor es meta.

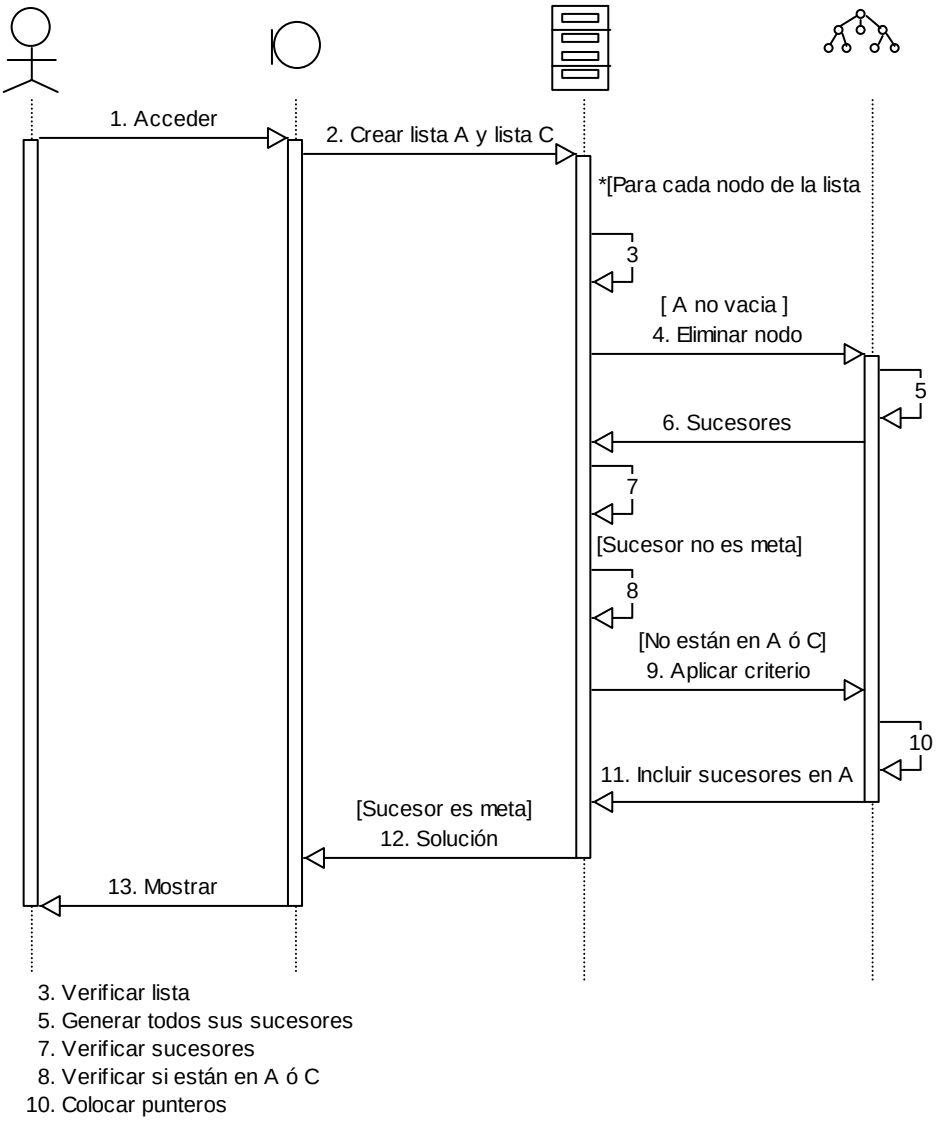

Figura 18. Diagrama de secuencias: búsqueda general en grafos cuando primer sucesor no es meta.

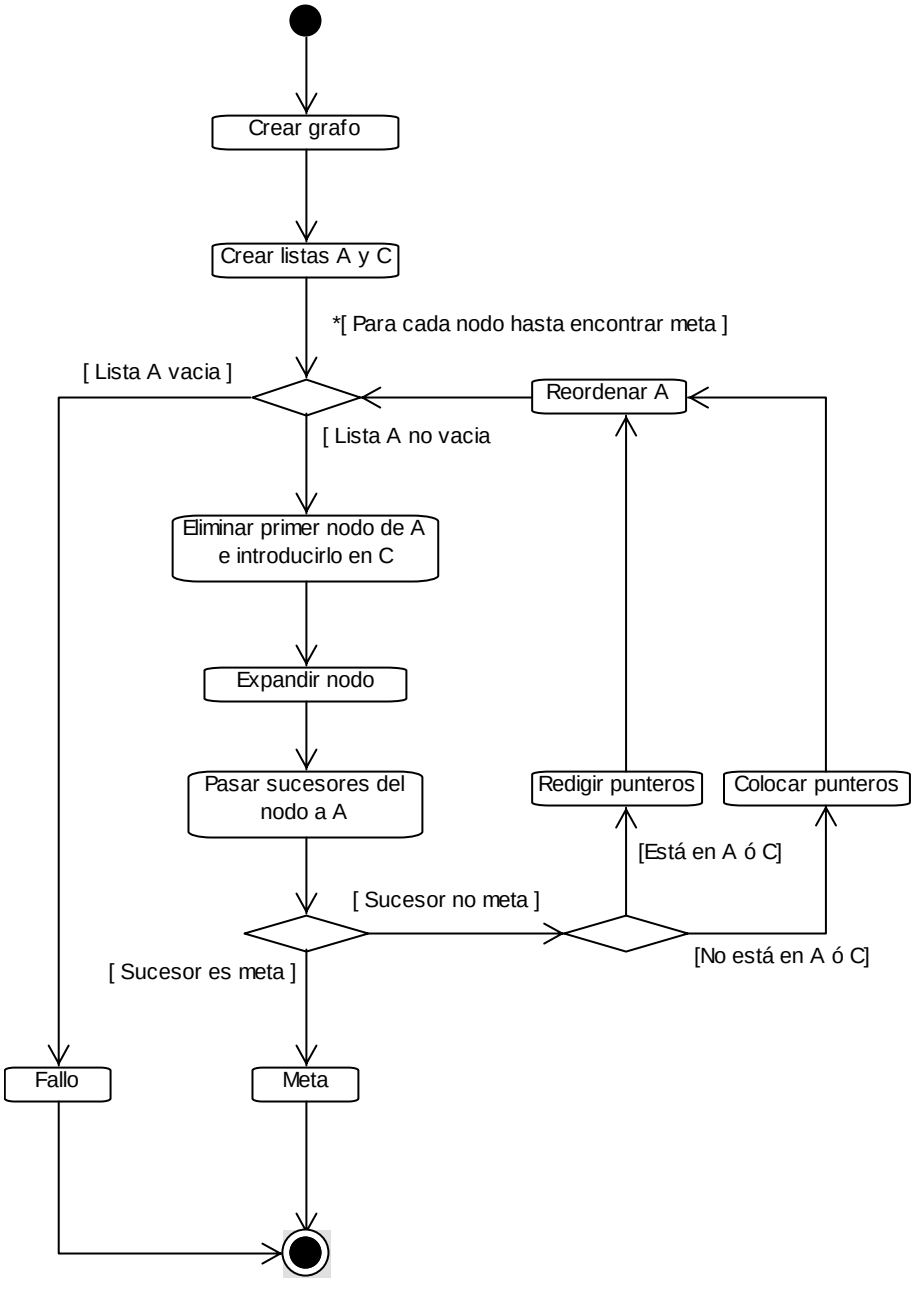

Figura 19. Diagrama de actividades: búsqueda general en grafos.

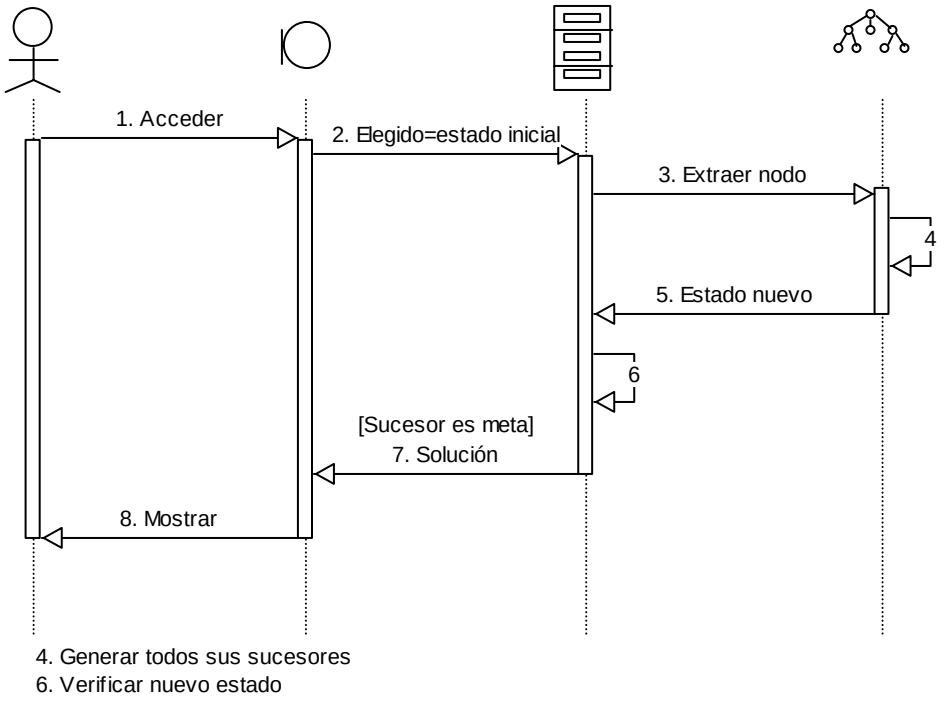

Figura 20. Diagrama de secuencias: búsqueda en escalada irrevocable cuando primer sucesor es meta.

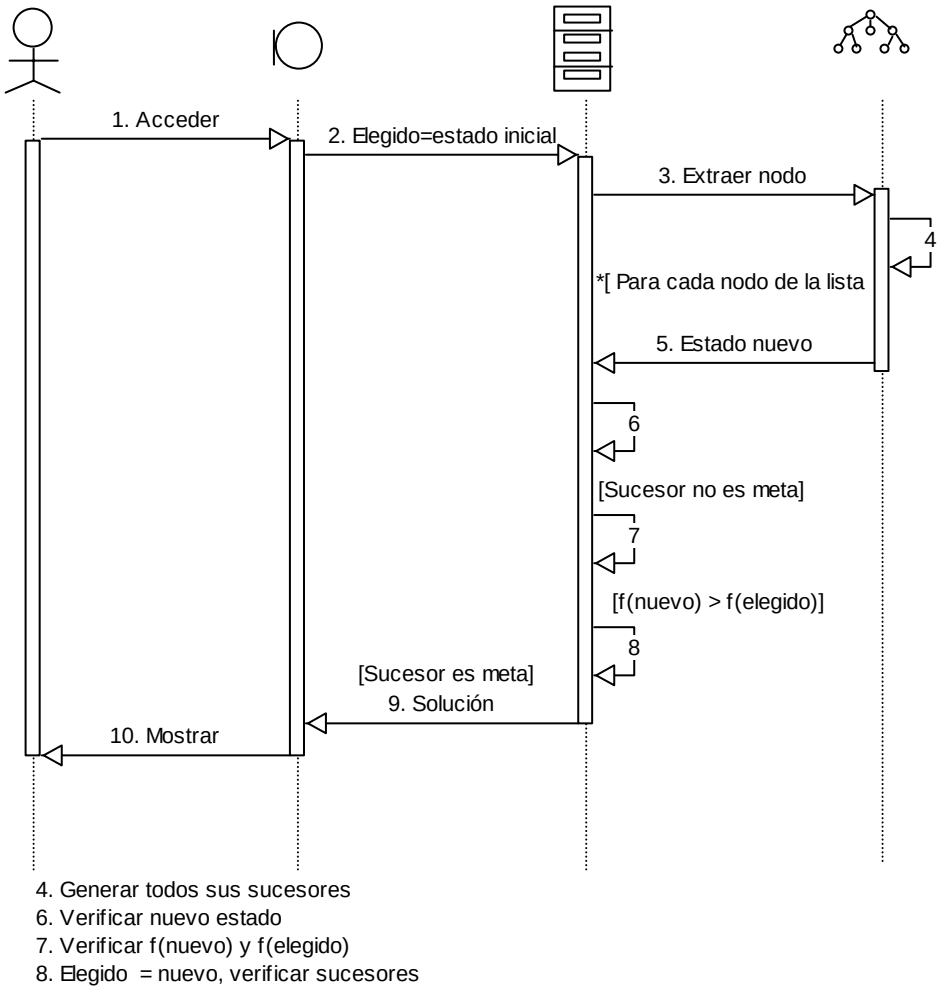

Figura 21. Diagrama de secuencias: búsqueda en escalada irrevocable cuando primer sucesor no es meta.

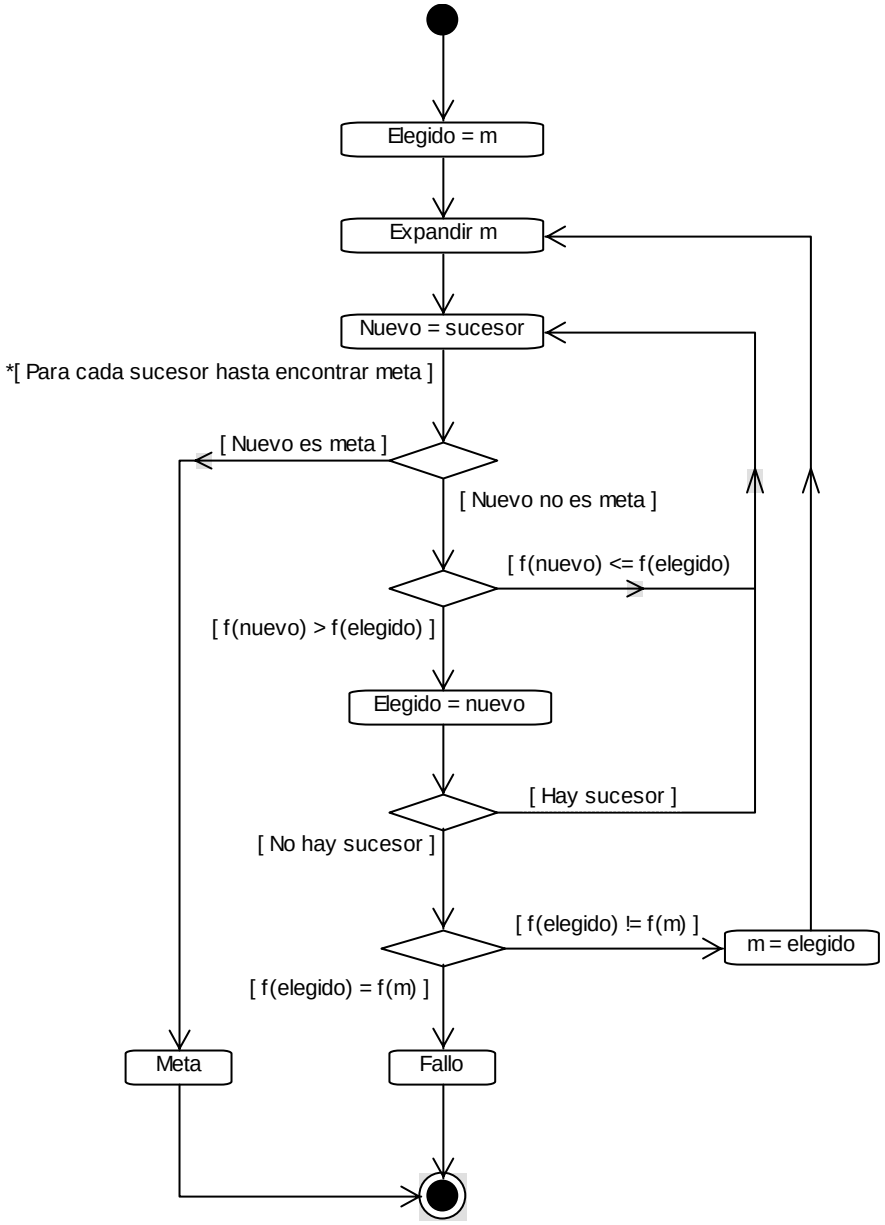

Figura 22. Diagrama de actividades: búsqueda en escalada irrevocable.

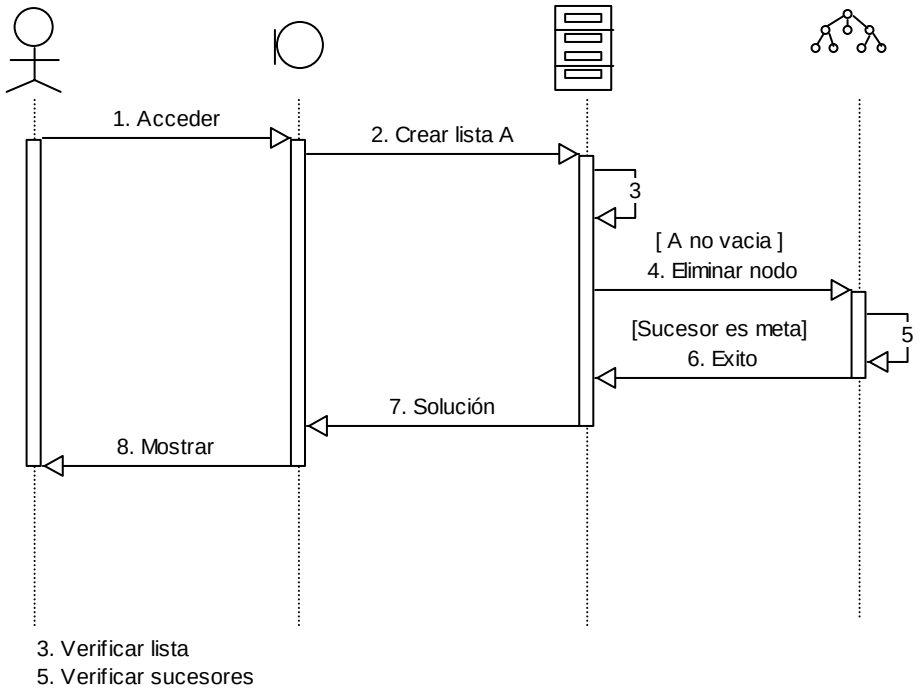

Figura 23. Diagrama de secuencias: búsqueda primero el mejor cuando primer sucesor es meta.

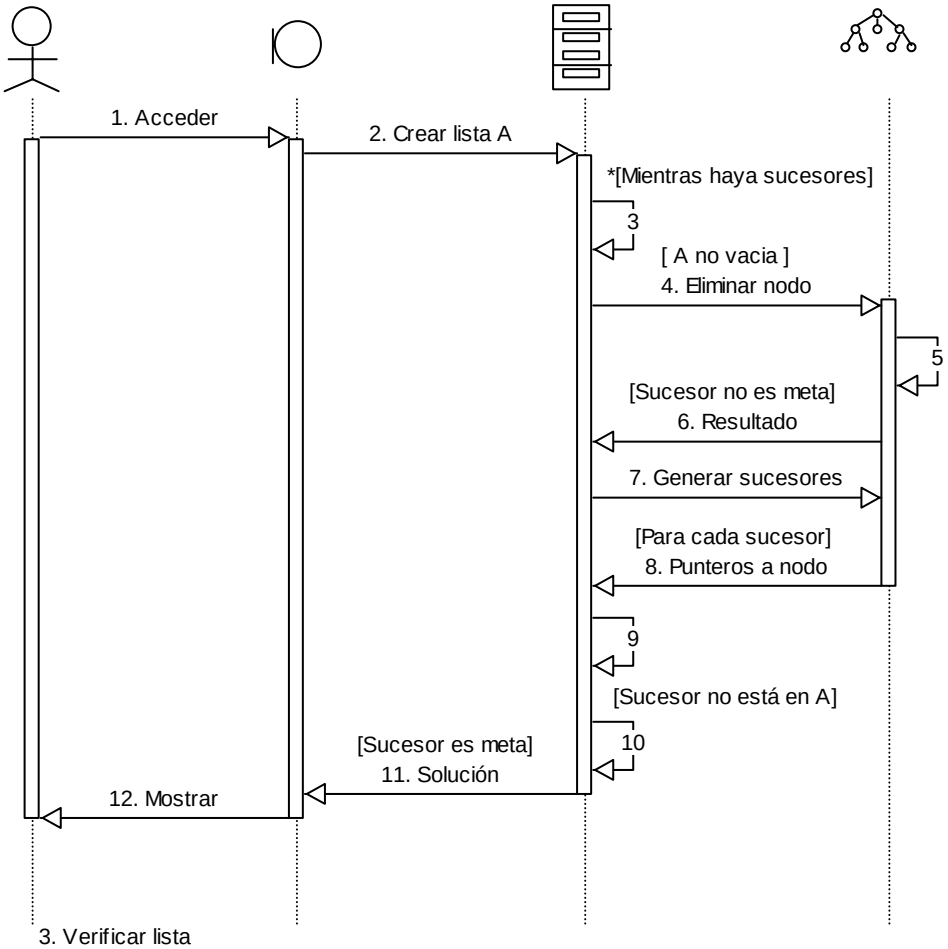

Figura 24. Diagrama de secuencias: búsqueda primero el mejor cuando primer sucesor no es meta.

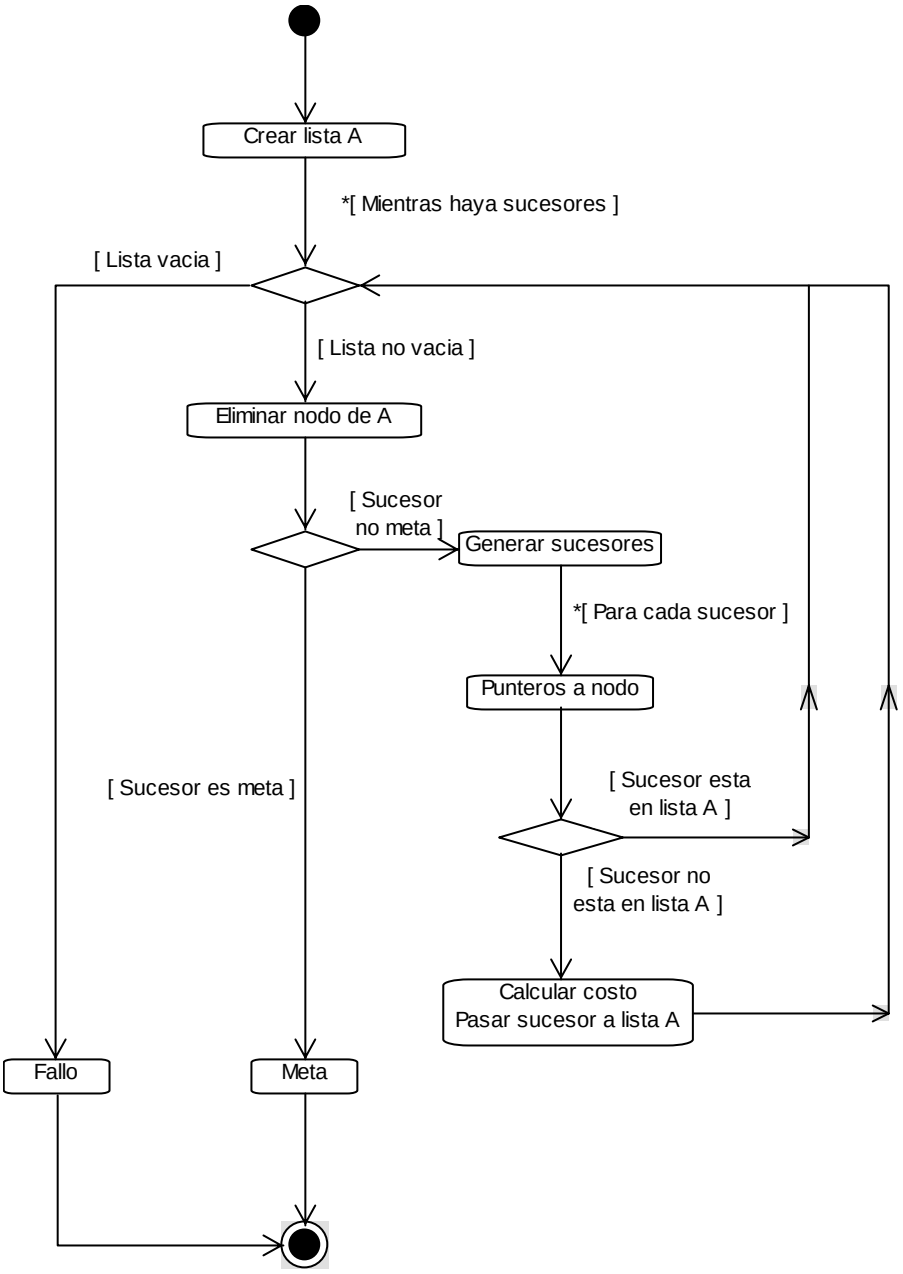

Figura 25. Diagrama de actividades: búsqueda primero el mejor.

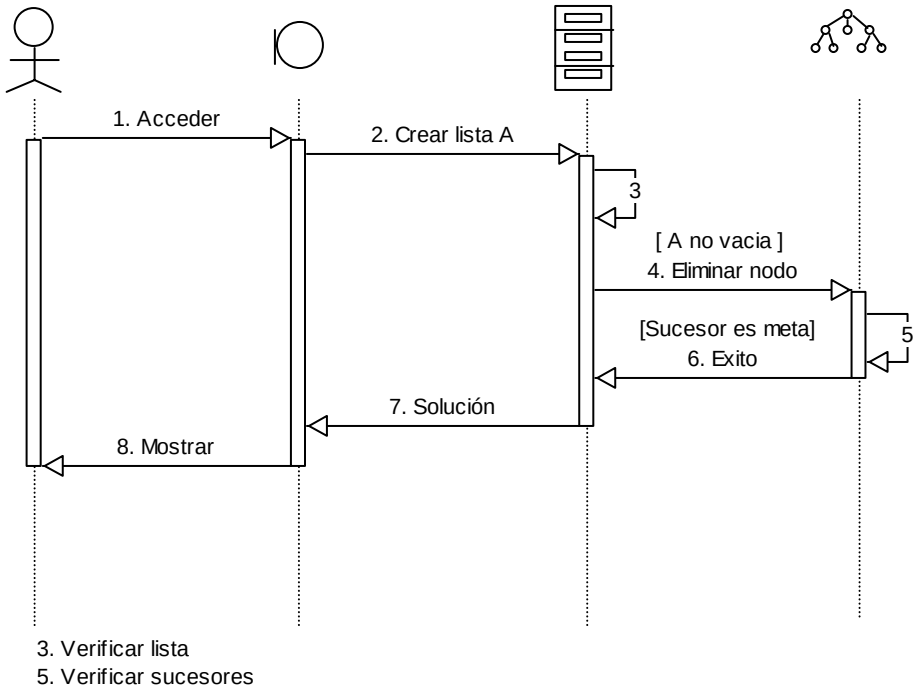

Figura 26. Diagrama de secuencias: búsqueda A*
cuando primer sucesor es meta.

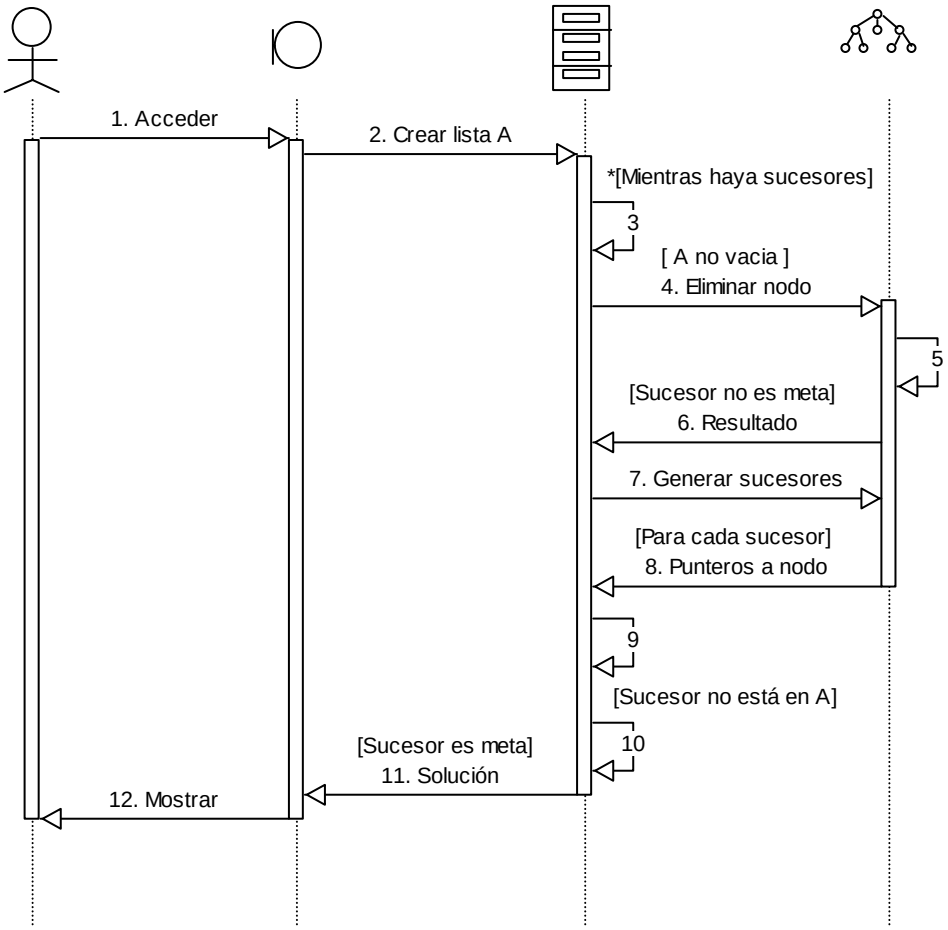

Figura 27. Diagrama de secuencias: búsqueda A*
cuando primer sucesor no es meta.

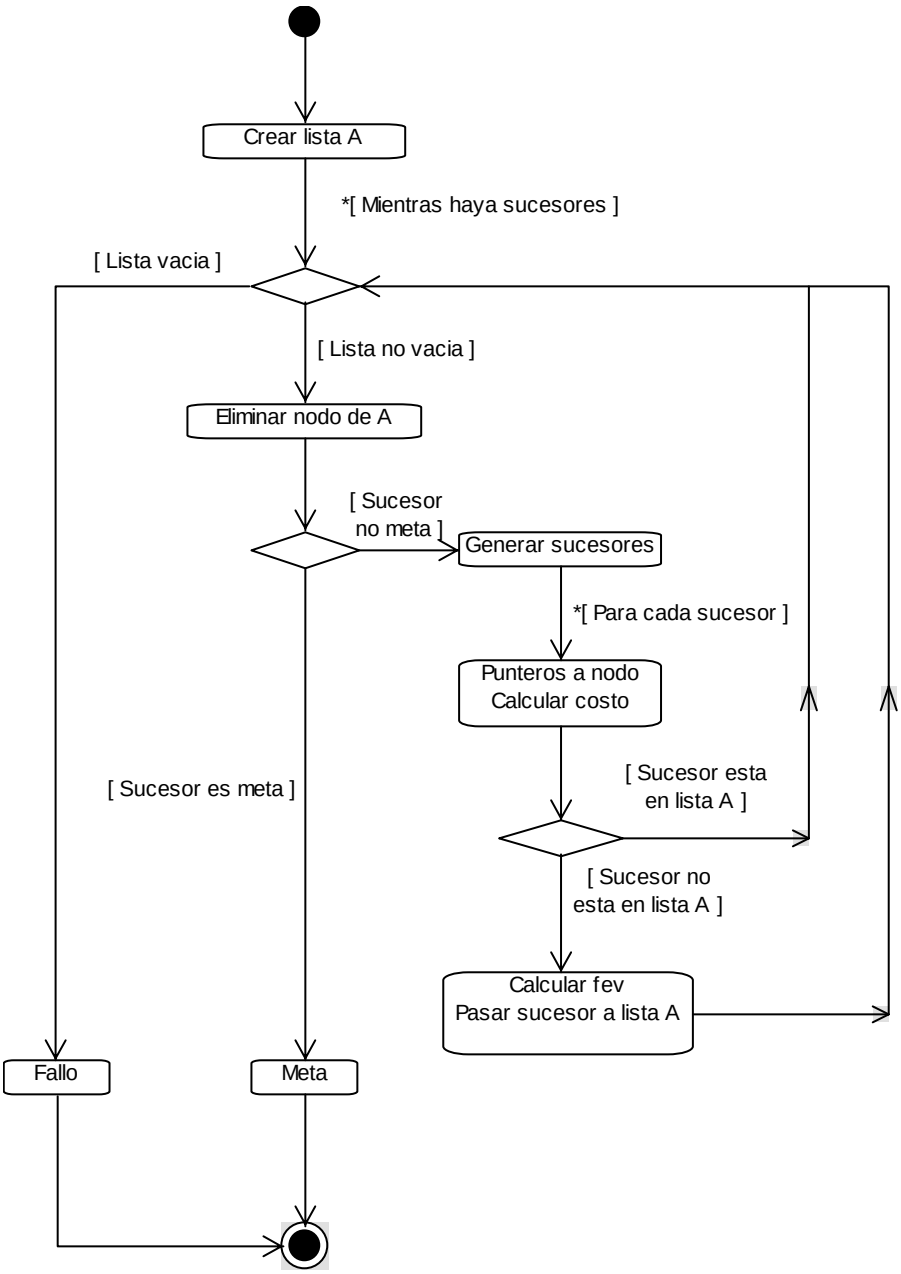

Figura 28. Diagrama de actividades: búsqueda A*.

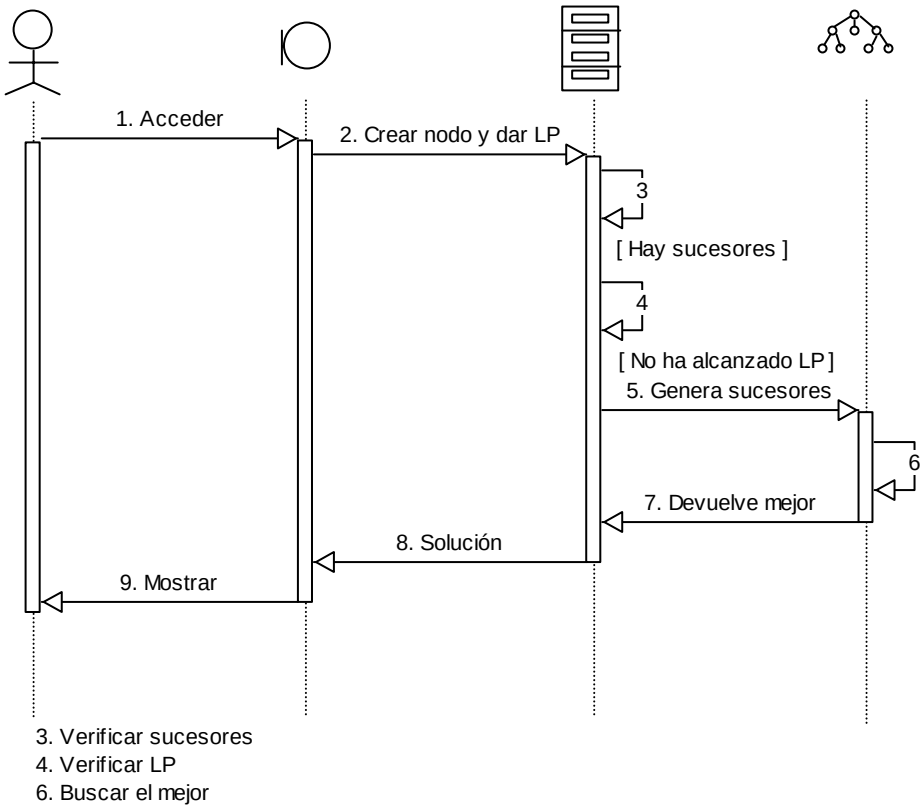

Figura 29. Diagrama de secuencias: búsqueda Mínimax cuando se generan sucesores.

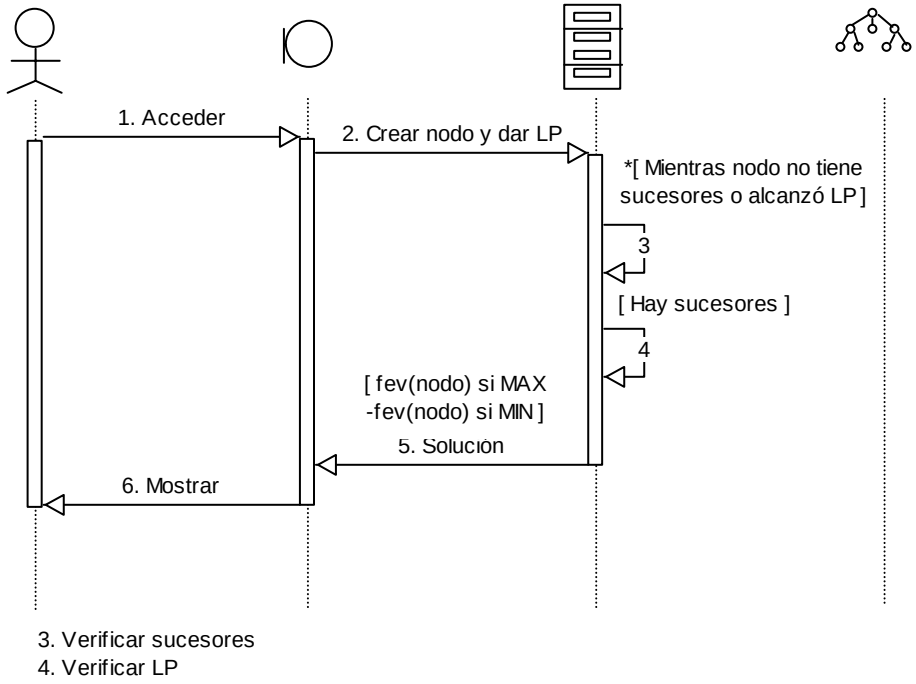

Figura 30. Diagrama de secuencias: búsqueda Mínimax
cuando m no tiene sucesores o ha alcanzado el limite de profundidad.

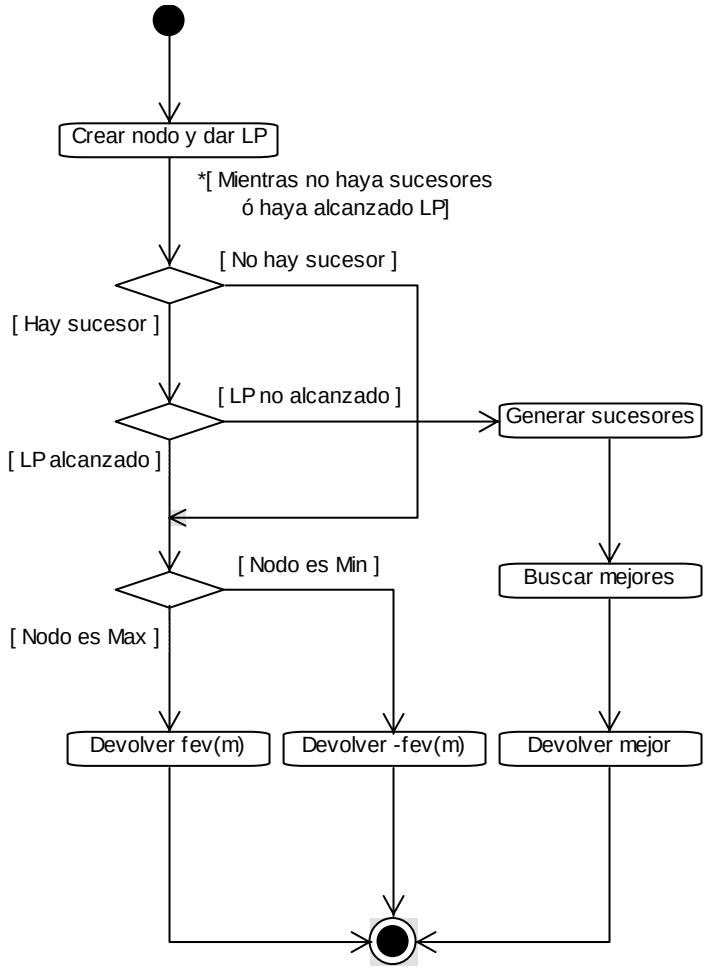

Figura 31. Diagrama de actividades: búsqueda Mínimax.

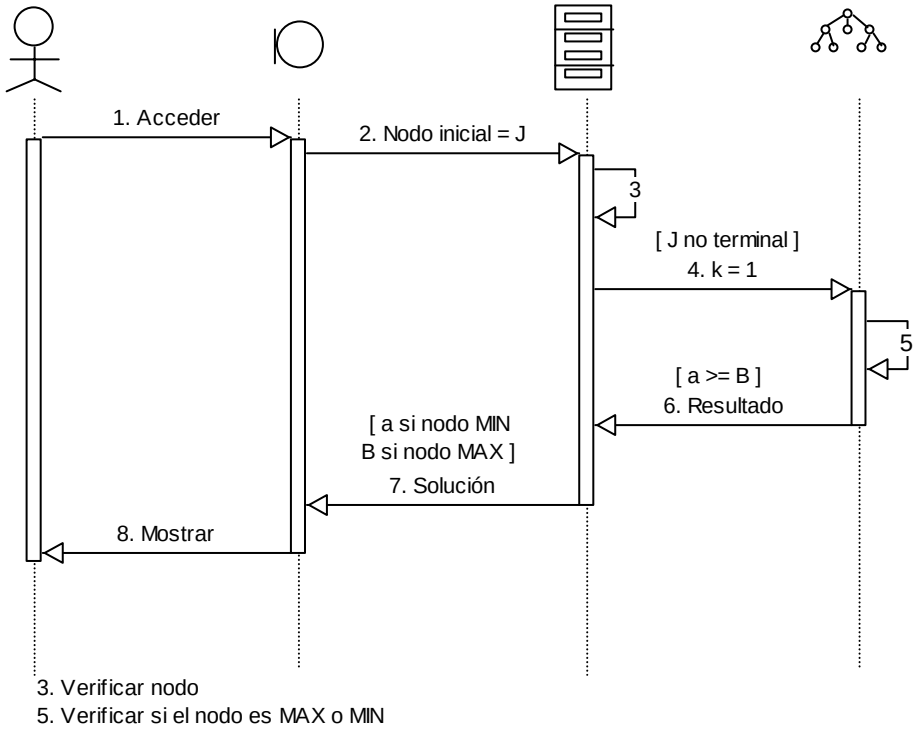

Figura 32. Diagrama de secuencias: búsqueda Poda Alfa – Beta cuando primer sucesor es meta.

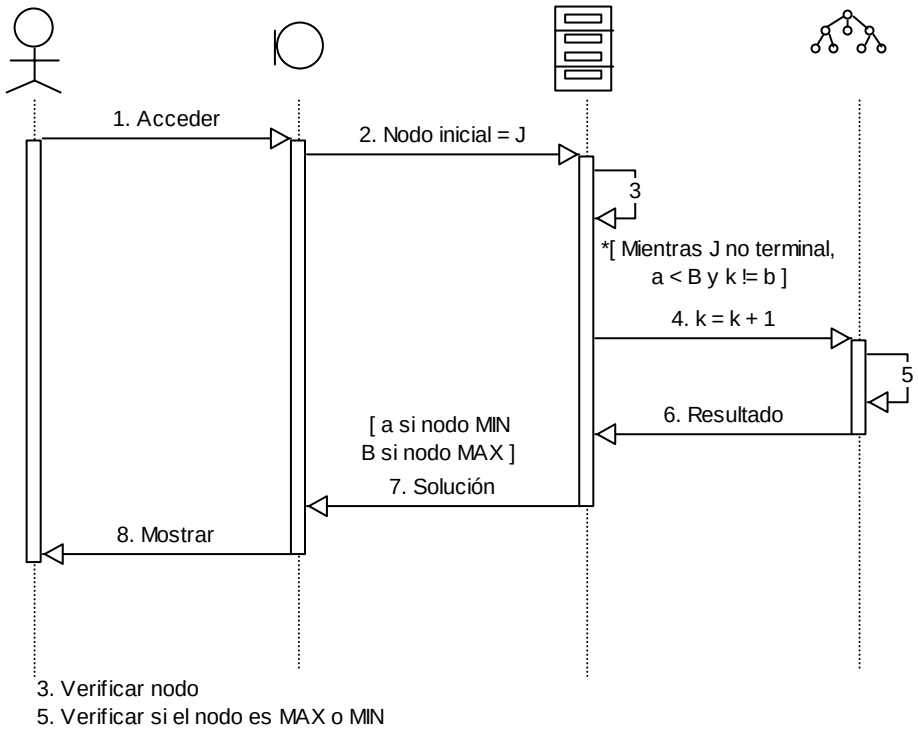

Figura 33. Diagrama de secuencias: búsqueda Poda Alfa – Beta cuando primer sucesor no es meta.

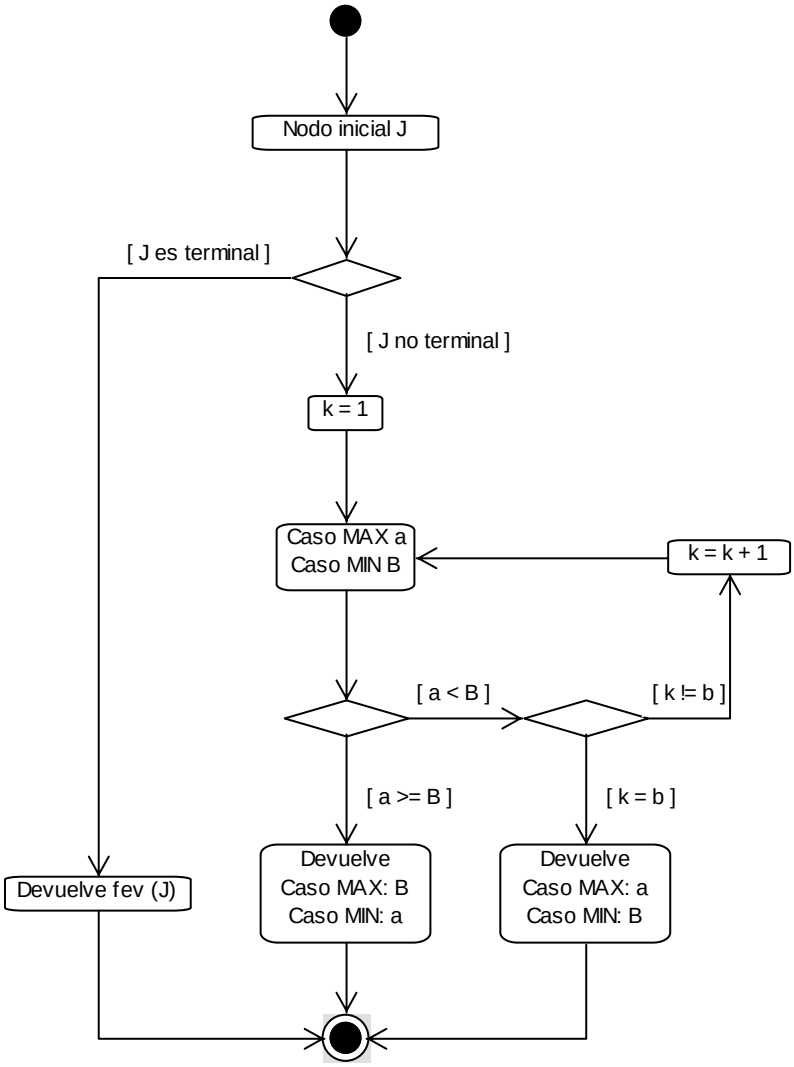

Figura 34. Diagrama de actividades: búsqueda Poda Alfa – Beta.

4.5.2. Construcción del Nuevo Prototipo

Para el desarrollo del nuevo prototipo se tuvo en cuenta la detección de errores al realizar las pruebas y la recopilación de cambios sugeridos por parte del cliente. Además, se definió el diseño gráfico que tendrá finalmente la herramienta, así como el logo que la identifica.

A este nuevo prototipo se le realizaron las pruebas correspondientes obteniendo cero errores y la aceptación por parte del cliente, convirtiéndose en el prototipo final.

FASE FINAL

4.6. PRODUCTO FINAL

4.6.1. Implantación del Prototipo Final

Se hizo el montaje de SABIA en el "Aula Virtu@l" como una herramienta específica del curso Inteligencia Artificial, para que quede a disposición de todos los usuarios que deseen consultarla.

5. CONCLUSIONES

El uso de applets para la implementación de simulaciones y juegos, le dio a la herramienta más dinamismo, permitiendo que los usuarios que la visiten, puedan afianzar el aprendizaje de los contenidos teóricos y fijarlos mediante su aplicación en casos concretos.

La creación de páginas HTML y applets, proporcionó una forma mediante la cual se puede distribuir software al cliente desde el servidor, en el momento en que lo necesite, y no antes, con lo cual siempre tendrá la ultima versión de éste.

La elección del Prototipado Evolutivo permitió aprovechar las ventajas que éste tiene sobre los demás métodos de desarrollo, manifestadas principalmente en el contacto con el cliente a lo largo del desarrollo y la constante realización de pruebas para la detección de errores, que permiten el mejoramiento gradual del prototipo para responder a todos los requerimientos del cliente.

El uso de software de libre distribución, para el desarrollo del proyecto posibilita una implementación de herramientas a bajo costo en cuanto a software, pero con calidad en cuanto a características y funciones.

El desarrollo de herramientas educativas con la característica de facilitar la actualización de contenidos, permite mantener al día la información que se ofrece a los estudiantes para apoyar sus procesos de aprendizaje.

6. RECOMENDACIONES

Las nuevas tecnologías intervienen en todas las áreas del quehacer cotidiano, entre ellas la educación, donde las Tecnologías de la Información y Comunicación (TIC) rompen esquemas o didácticas tradicionales generando nuevas posibilidades de comunicación, mediante la creación de un entorno fluido y multididáctico de comunicación entre profesores y alumnos, y entre los mismos alumnos. El "Aula Virtu@l" hace parte de estas tecnologías. Teniendo en cuenta lo anterior y tomando como referencia diversos proyectos que se han hecho para el "Aula Virtu@l", incluyendo a SABIA, se sugiere la creación de nuevas herramientas que apoyen el aprendizaje de los contenidos de clase, exigiendo nuevos roles en profesores y alumnos; ya que los primeros se convierten en orientadores, mientras que los segundos se convierten en protagonistas de su propio aprendizaje.

7. BIBLIOGRAFIA

PRESSMAN, Roger S. Ingeniería del software. Un enfoque práctico. Editorial Mc. Graw Hill. Cuarta edición. España, 1998.

GOMEZ Flórez, Luis Carlos. Proyectos informáticos. Notas de clase. UIS. 2001 – 2003.

FOWLER, Martín. UML gota a gota. Editorial Adison Wesley Longman de México, S.A. de C.V. Primera edición. México, 1999.

LARMAN, Craig. UML y patrones. Introducción al análisis y diseño orientado a objetos. Editorial Prentice Hall. Primera edición. México, 1999.

BOOCH, Grady; **RUMBAUGH**, James; **JACOBSON**, Ivar. El lenguaje unificado de modelado. Editorial Addison-Wesley Iberoamericana. Madrid, 1999.

CEBALLOS, Francisco Javier. Java 2. Curso de programación. Editorial Alfaomega. México, 2000.

BECERRA Santamaría, César A. Los 600 principales métodos del Java. Editorial Kimpres Ltda. Primera edición. Colombia, 1998.

FERNANDEZ Galán, Severino; **GONZALEZ** Boticario, Jesús; **MIRA** Mira, José. Problemas resueltos de inteligencia artificial y aplicada. Búsqueda y representación. Editorial Addison – Wesley. España, 1998.

MACROMEDIA, Inc. Macromedia dreamweaver: using dreamweaver. San Francisco. 2000.

CORREDOR Montagut, Marta Vitalia y otros. Aula Virtu@l: "una alternativa en educación superior". Ediciones UIS. 2004.

CORREDOR Montagut, Martha Vitalia. Principios de inteligencia artificial y sistemas expertos. Ediciones UIS. Bucaramanga, 2000.

RUSELL, Stuart. Inteligencia artificial un enfoque moderno. Editorial Prentice Hall Hispanoamericana.S.A. Primera edición. México, 1996.

MEZA Vega, Edwin. Tesis de grado: Ambiente virtual para apoyar aprendizajes colaborativos con aplicación específica en la hidráulica de canales abiertos. Universidad Industrial de Santander. 2002.

ANEXO A

DESCRIPCION DE SABIA

DESCRIPCION DE SABIA

Después de acceder al entorno "Aula Virtu@l", en el curso de Inteligencia Artificial (ver Figura 35), puede tener acceso a las herramientas que éste ofrece en el menú de opciones generales. Allí se despliega una nueva ventana, donde se encuentra un enlace a la herramienta SABIA.

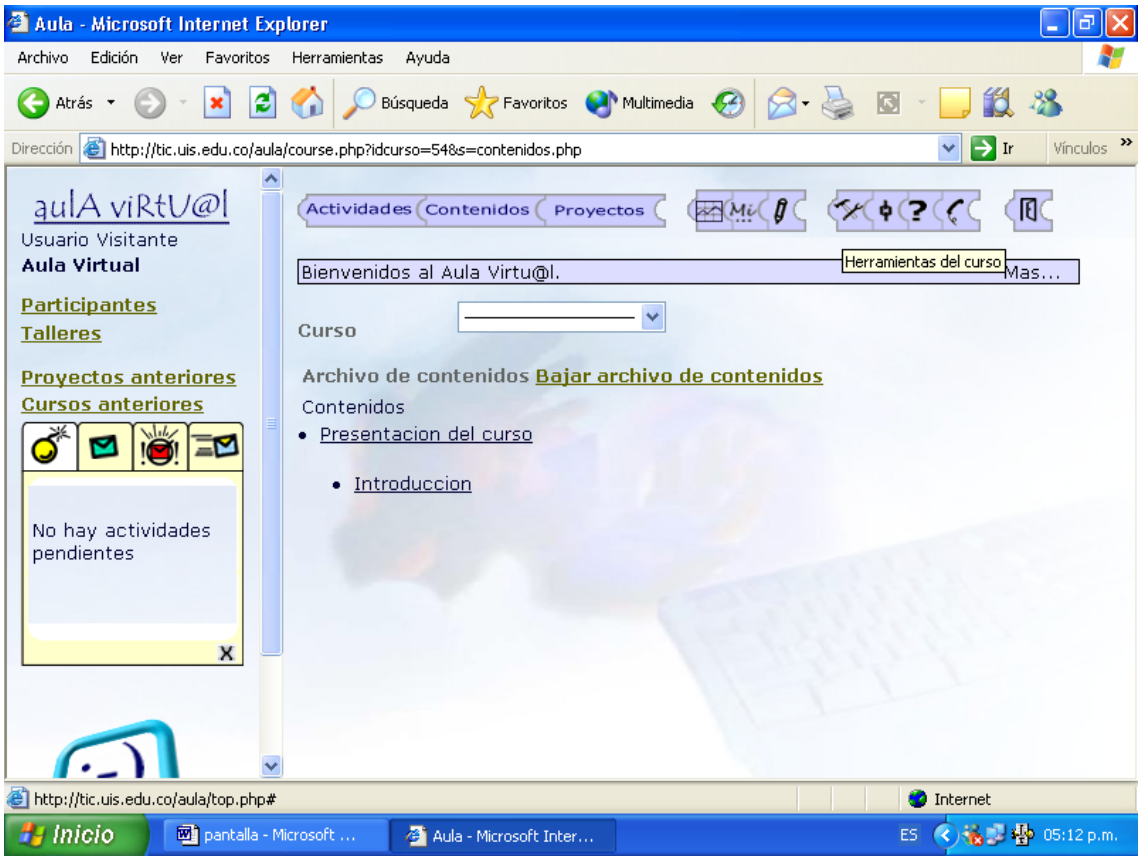

Figura 35. "Aula Virtu@l" y ventana de herramientas.

El enlace lo lleva a una página inicial (ver Figura 36) donde el usuario encuentra el respectivo logo y nombre de la herramienta, logotipos de las entidades interesadas y responsables de su desarrollo, así como un icono que invita al usuario a entrar y explorar la herramienta.

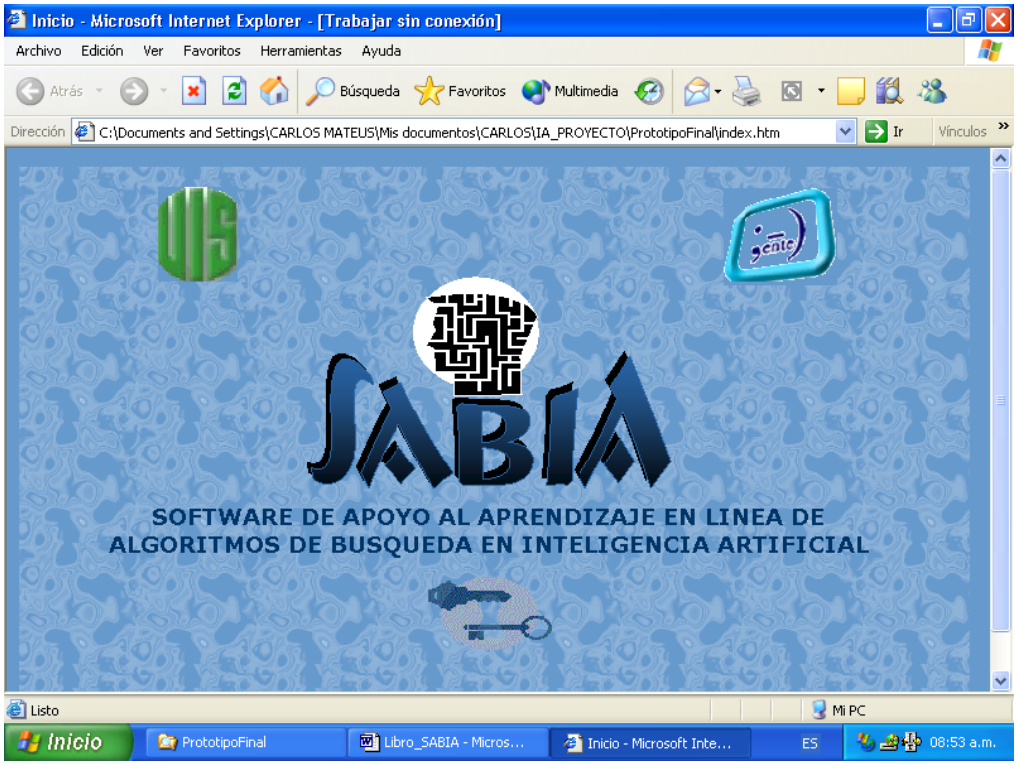

Figura 36. Página inicial de SABIA.

Comenzando con la exploración de la herramienta, se presenta una página (ver Figura 37) de introducción al tema de los algoritmos de búsqueda, de igual manera se observa un menú con las opciones generales.

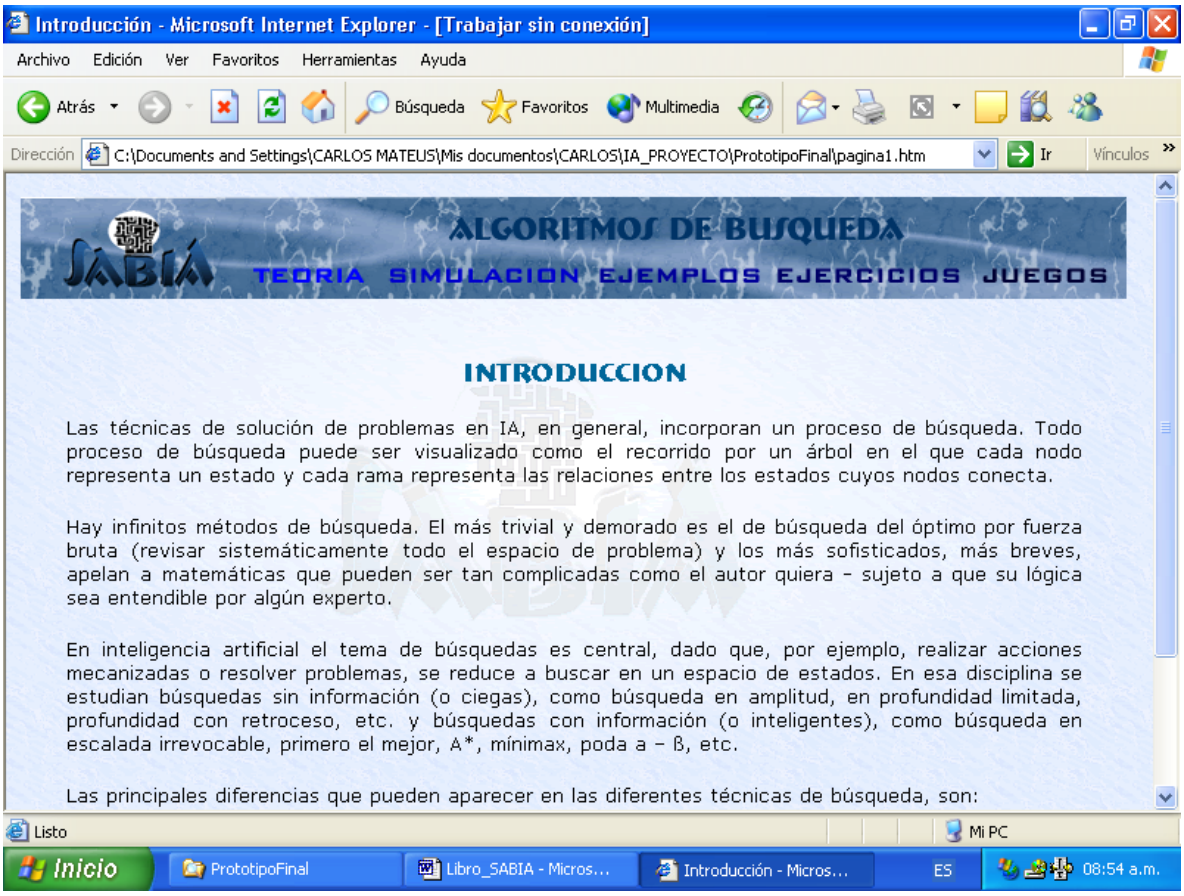

Figura 37. Página de introducción.

Dentro de las opciones TEORIA y SIMULACION (ver Figuras 38 y 39) aparece un nuevo menú con las opciones específicas, las cuales hacen referencia a cada uno de los algoritmos de búsqueda trabajados en la herramienta.

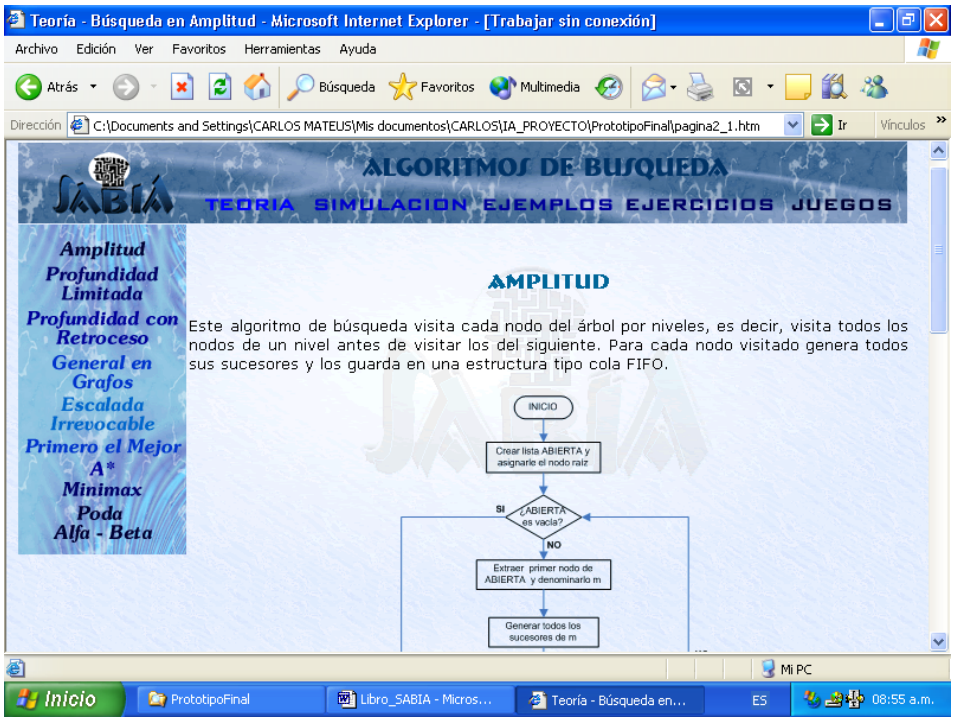

Figura 38. Página de teoría.

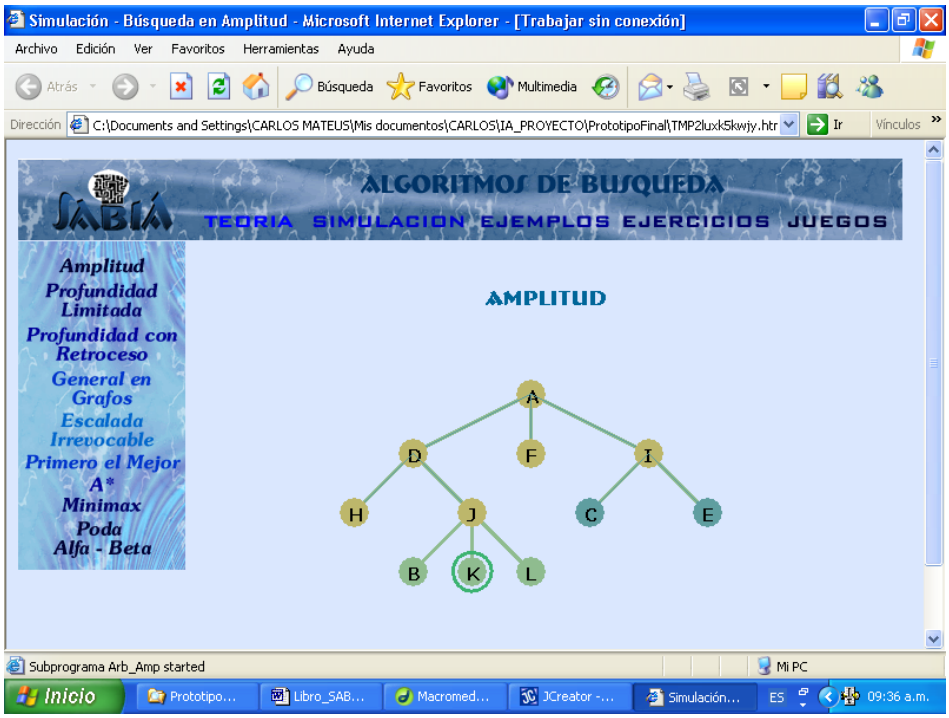

Figura 39. Página de simulaciones.

En las opciones EJEMPLOS, EJERCICIOS y JUEGOS (ver Figuras 40 - 42), no aparece el menú de opciones específicas, debido a que el contenido que presentan incluye todos los algoritmos de búsqueda.

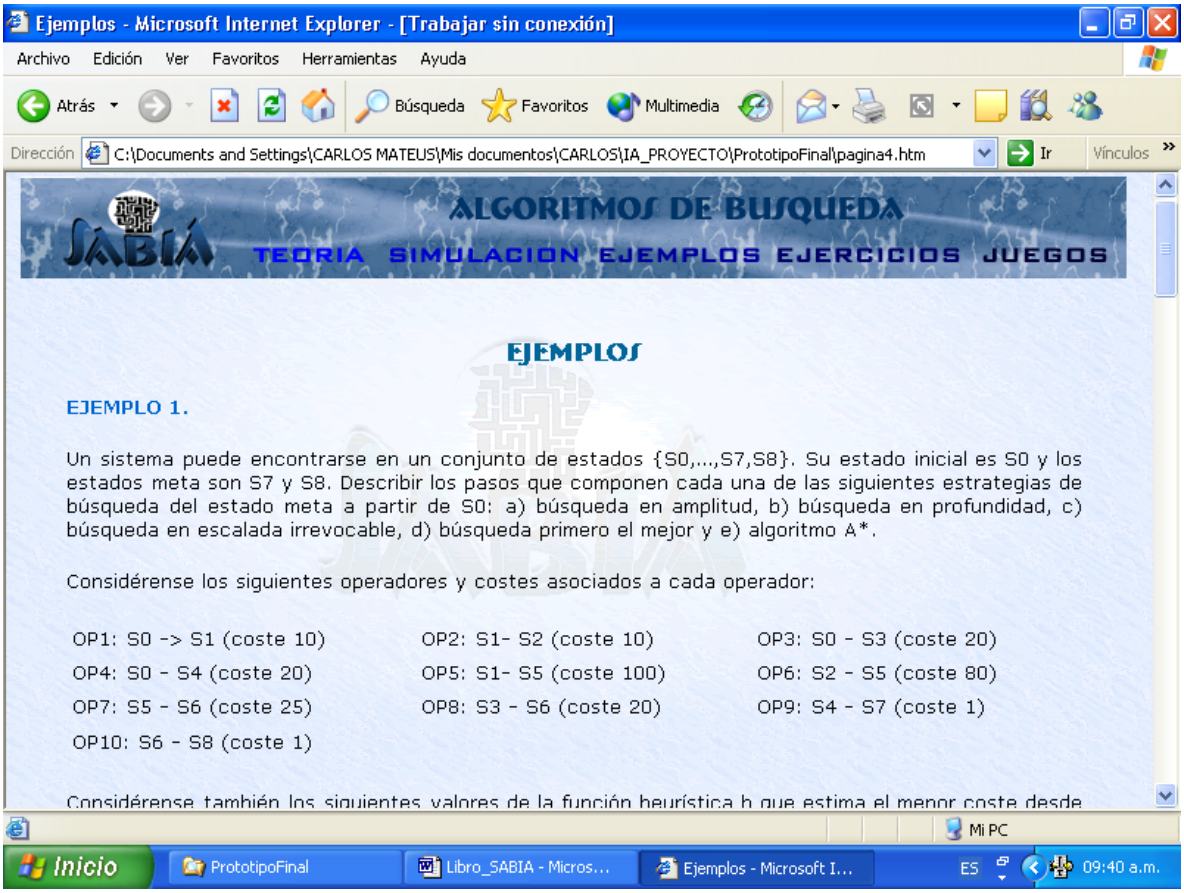

Figura 40. Página de ejemplos.

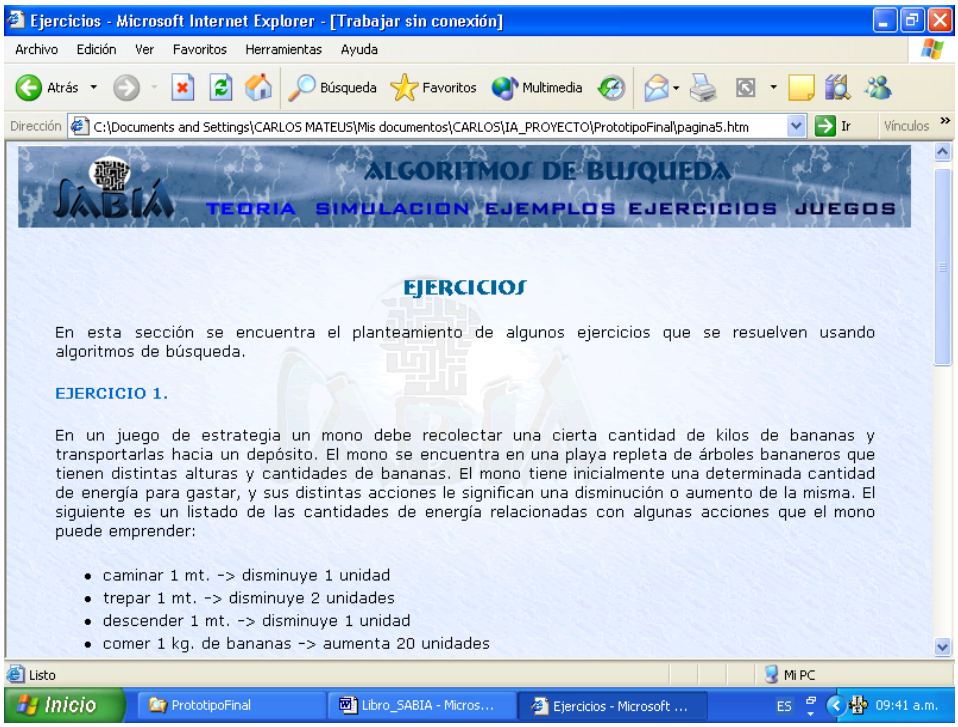

Figura 41. Página de ejercicios.

Figura 42. Página de juegos.