DESARROLLO DE UN ENTORNO INTERACTIVO PARA EL APRENDIZAJE DE LÓGICA DE PROGRAMACIÓN “SEUPROG”

ROBERT SNEYDER MORENO MOSQUERA

C.C: 12023348

CONVENIO UNAB - UOC

MAESTRIA EN SOFTWARE LIBRE

BUCARAMANGA

2012

DESARROLLO DE UN ENTORNO INTERACTIVO PARA EL APRENDIZAJE DE LÓGICA DE PROGRAMACIÓN "SEUPROG"

ROBERT SNEYDER MORENO MOSQUERA

C.C: 12023348

Anteproyecto de grado presentado como requisito parcial para optar al título de Magister en Software Libre

Directora: Mcc. Olga Lucia Monroy Vecino

CONVENIO UNAB - UOC

MAESTRIA EN SOFTWARE LIBRE

BUCARAMANGA

2012

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bucaramanga, 30/07/2012

DEDICATORIA

A mi familia por creer en mí y por brindarme su apoyo de manera incondicional a lo largo de los estudios de maestría y durante el tiempo que duró la realización de este trabajo.

AGRADECIMIENTOS

A Olga Lucia Monroy, por haberme ayudado durante todo el proceso.

A los estudiantes de Ingeniería teleinformática de la UTCH, por servir como usuarios de prueba de la aplicación y poder alcanzar los objetivos.

CONTENIDO

LISTA DE TABLAS

LISTA DE ILUSTRACIONES

RESUMEN

Este Trabajo llamado “Desarrollo de un entorno interactivo para el aprendizaje de lógica de programación “Seuprog” pretende brindar un conocimiento básico e importante a la hora de desarrollar un entorno interactivo para la enseñanza de lógica de Programación en nivel introductorio, utilizando herramientas de software libre. En la parte inicial del documento se tratará todo lo relacionado con un estudio de herramientas y entornos para la enseñanza de la programación aplicados en otras instituciones educativas. Luego se abordará el problema por medio de la Ingeniería de Software donde se muestra todo el análisis y el Diseño necesario para poder construir la aplicación para el diseño de soluciones algorítmicas a problemas propuestos. Finalmente se muestran los resultados de las pruebas realizadas con dos grupos pilotos de estudiantes de ingeniería de la UTCH y se dan todas las recomendaciones pertinentes y las conclusiones a las que se ha llegado.

1. OBJETIVO GENERAL

Desarrollar un entorno interactivo para la enseñanza de lógica de Programación en nivel introductorio, utilizando herramientas de software libre.

2. OBJETIVOS ESPECIFICOS

- Realizar un estudio de herramientas y entornos para la enseñanza de la programación, aplicados en otras instituciones educativas.

- Desarrollar un entorno para el diseño de soluciones algorítmicas a problemas propuestos.

- Desarrollar un traductor de los algoritmos diseñados a un lenguaje de programación.

- Realizar pruebas con un grupo piloto de estudiantes de ingeniería de la UTCH y documentar los resultados.

3. INTRODUCCIÓN

El presente trabajo muestra información muy concreta sobre lógica de programación, más específicamente en nivel introductorio que es el componente principal de esta tesis. Se realizan encuestas a estudiantes de Ingeniería de la UTCH de dos grupos pilotos. Conociendo los problemas que tienen con la asignatura se procede con el desarrollo de una aplicación que facilite el diseño de algoritmos, esta herramienta está compuesta por 3 áreas principales en su interfaz Comandos, Algoritmos y Resultados. Para esto se realizan el análisis y el diseño claro y concreto con términos simples que permiten entender de manera fácil de que se está hablando y a que se refiere cada operación para la creación de la misma. Luego se muestran los comparativos de los resultados obtenidos de las pruebas, se dan a conocer algunas recomendaciones para tener en cuenta a la hora de realizar este tipo de aplicaciones y por último unas conclusiones claras sobre el proceso y el resultado final.

4. JUSTIFICACION

Desde hace mucho tiempo el uso de lenguajes de programación de alto nivel como C, C++, C# o JAVA en una materia como lógica de programación ha presentado notorias dificultades para estudiantes inexpertos que deben aprender conceptos relativos al diseño de algoritmos y al mismo tiempo lidiar con cuestiones de implementación de las soluciones propuestas relativas a un lenguaje de programación como Sintaxis, Compilación, mensajes de errores en inglés, depuración etc. Por este motivo se propone diseñar algoritmos en pseudocódigo en español, con reglas sintácticas sencillas y básicas, que permitan concentrar al alumno en la lógica para la resolución de problemas mediante el diseño y la construcción de algoritmos y facilitar el aprendizaje y uso posterior de un lenguaje de alto nivel. Aunque la utilización del pseudocódigo en el Tablero a la hora de enseñar los conceptos básicos de programación produce un avance significativo, existen otros factores que afectan el desempeño de los estudiantes y el desarrollo de la asignatura. Partiendo del hecho de que un problema puede ser resuelto de muchas formas distintas y teniendo en cuenta el elevado número de estudiantes por salón de clase y la heterogeneidad de conocimientos de los mismos, se hace imposible en la práctica el seguimiento de todas las soluciones planteadas individualmente por cada alumno. De estas observaciones surge la motivación original de este Proyecto.

Muchos estudiantes que no tienen base alguna demuestran grandes dificultades para entender la lógica subyacente en las estructuras de datos y de control, lo cual llama la atención y junto con la necesidad de presentar un proyecto que facilite la enseñanza de la asignatura y al mismo tiempo sirva como propuesta de tesis para la maestría en software libre estimulan el desarrollo de este proyecto.

Al ser un software para uso exclusivamente didáctico, no se presentan grandes problemas de rendimiento por lo que se tendrán ciertas libertades en la implementación. Se debe tener en cuenta que el objetivo principal del intérprete no es sólo interpretar un buen

código, sino también señalar correctamente los errores de uno incorrecto. La idea es sugerir en el año 2012 a los otros compañeros responsables de la cátedra ofrecer el software a sus futuros alumnos.

5. ESTADO DEL ARTE

Partiendo desde la referencia que el uso de lenguajes de programación de alto nivel como C, C++, C# o JAVA en una materia como lógica de programación ha presentado notorias dificultades para la gran mayoría de estudiantes inexpertos que deben aprender conceptos relativos al diseño de algoritmos y lidiar con la implementación de las soluciones en un lenguaje de programación como Sintaxis, Compilación, mensajes de errores en inglés, depuración etc. Desde diferentes regiones del mundo y por diferentes Profesionales de la asignatura se han realizado proyectos e investigaciones que buscan diseñar algoritmos en pseudocódigo, con reglas sintácticas sencillas y básicas, que permitan concentrar a los alumnos en la lógica para la resolución de los problemas mediante el diseño y la construcción de algoritmos y facilitar el aprendizaje y uso posterior de un lenguaje de alto nivel.

Inicialmente la utilización del pseudocódigo en el Tablero a la hora de enseñar los conceptos básicos de programación ha producido un avance significativo, pero existen otros factores que afectan el desempeño de los estudiantes y el desarrollo de la asignatura.

Norma Moroni y Perla Señas del Instituto de Ciencias e Ingeniería de Computación y la Universidad Nacional del SUR en Argentina desarrollaron un proyecto en el año 1996 llamado ***"UN ENTORNO PARA EL APRENDIZAJE DE LA PROGRAMACIÓN"*** donde proponían un entorno de programación adecuado que permitiera a sus alumnos desarrollar los algoritmos trabajando directamente sobre la computadora evitando tener que recordar expresamente detalles del diseño estructural de los mismos. Sin embargo durante su desarrollo tuvieron algunos problemas como:

- Para un alto porcentaje de los alumnos, la elaboración de algoritmos "con lápiz y papel" resultaba una tarea pesada y poco atractiva. Muchos de ellos aducen que el desarrollo de esa tarea no cubre sus expectativas, ya que lo que deseaban era "trabajar con la computadora".
- El trabajo con el editor de algoritmos seria de tipo interactivo. Ofrecería al alumno las estructuras básicas de un lenguaje algorítmico que él completará según cada caso. Cada una de esas estructuras constaría de ***textos fijos*** y de ***textos reemplazables***. Los textos fijos quedarían en el algoritmo, y los reemplazables deberían ser sustituidos de acuerdo a su semántica.

  Por ejemplo la estructura de control condicional seria ofrecida por el editor de la siguiente manera:

  Si *condición* **Entonces**

  Acción

  Aquí **Si** y **Entonces** constituían los textos fijos, mientras que *condición* y *acción* eran los textos reemplazables que deberían ser sustituidos por una expresión booleana y una estructura de control, respectivamente. El editor ofrecería facilidades para la creación de los textos de reemplazo, controlaría su validez, y en caso de detectar entradas no válidas ofrecería mensajes orientadores y la posibilidad de realizar la corrección.
- Finalmente el programa tenía que traducirse al lenguaje PASCAL pidiéndole al usuario el nombre del archivo para guardarlo y luego poder ejecutarlo independientemente[1].

En el volumen 5 del 2006 de la revista Journal of Information Technology Education los ingenieros Samer Al-Imamy, Javanshir Alizadeh y Mohamed A. Nour de Universidades de Emiratos Árabes Unidos publicaron el proyecto "***On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process"***,

[1] MORONI, Norma. UN ENTORNO PARA EL APRENDIZAJE DE LA PROGRAMACIÓN. Bahía Blanca: 1996, 7 p.

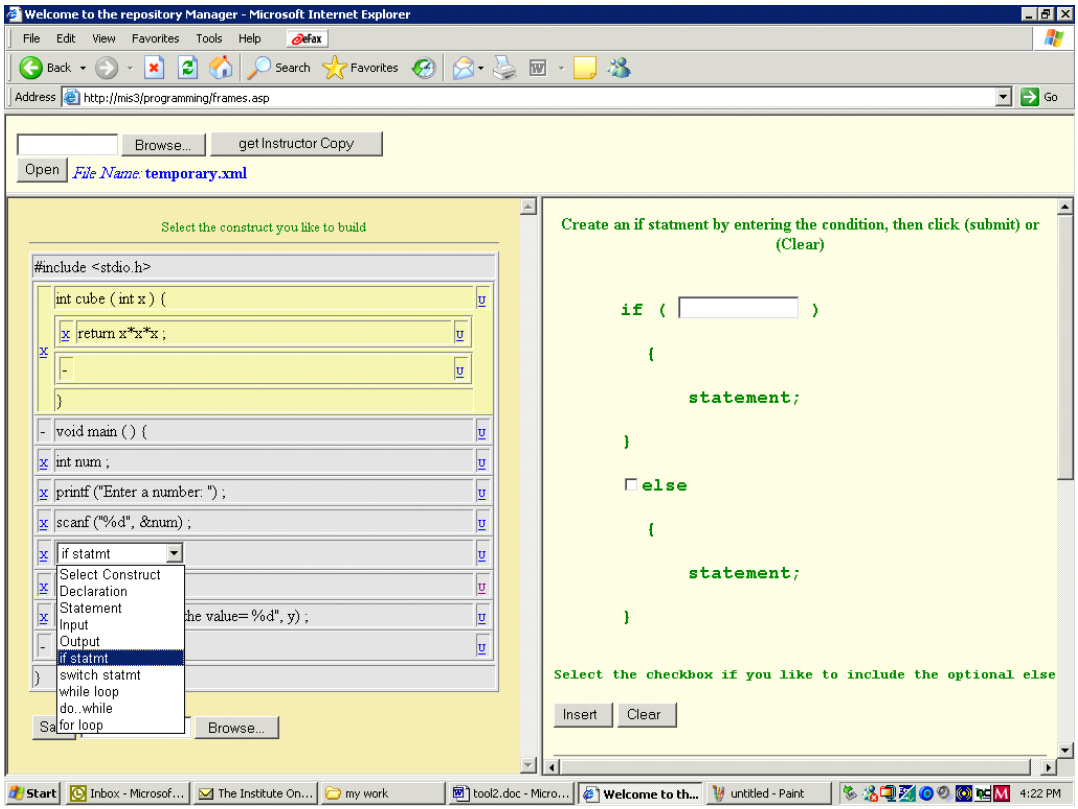
Sustentaron que uno de los principales problemas relacionados con la enseñanza de un curso introductorio a la programación era la excesiva cantidad de tiempo dedicado a la sintaxis del lenguaje, lo que dejaba poco tiempo para el desarrollo de habilidades en el diseño y la creatividad de la solución algorítmica. La solución planteada para este dilema fue proporcionar un ambiente de enseñanza que transformara y mejorara la enseñanza en el aula tradicional, con una herramienta de software para la enseñanza de un primer curso de lenguajes de programación. La herramienta fue utilizada en una de las cuatro secciones de un curso de programación en su institución con favorables resultados comparativos. Los resultados indicaron que la herramienta fue muy eficaz para mejorar el rendimiento de los estudiantes en el aprendizaje de los conceptos de programación y reducir el tiempo empleado en exceso de sintaxis. Además, la herramienta demostró ser útil en lo siguiente:

- La creación de plantillas para aplicar conceptos y construir automáticamente el código del programa, proporcionando flexibilidad para cambiar de un lenguaje de programación a otro.
- Ayudando a cerrar la brecha entre estudiantes de diversos orígenes.
- Aumentando el entusiasmo de los estudiantes y la confianza en la escritura de programas correctos.
- proporcionando una herramienta de auto-aprendizaje basado en un entorno web que facilita la enseñanza y el aprendizaje de los estudiantes.

Estos profesionales desarrollaron Plantillas de programas como lo muestra la figura 1, El usuario tenía la estructura principal del programa, la combinación de teclas “Control+X” para eliminar cualquier declaración (salvo la estructura principal) y “Control+U” para añadir una nueva declaración. Al pulsar “Control+U” se mostraba una lista de todos los estados posibles válidos en ese lugar del programa.

Los estados posibles de incluir en el cuerpo de la función principal eran if, for, while, etc. Sin embargo, la lista de selección en la parte superior del programa contiene la definición de librerías #include, la definición de variables globales #define, y así sucesivamente.

Ilustración 1: Proyecto "On the Development of a Programming Teaching Tool"



Fuente: web http://www.jite.org/documents/Vol5/v5p271-283Al-Imamy115.pdf

En el XIV Congreso Argentino de Ciencias de la Computación realizado en argentina en el año 2008 Jiménez Rey, E. Rodríguez, D. Britos, y P. García-Martínez del Centro de Ingeniería de Software e Ingeniería del Conocimiento ITBA Mostraron el proyecto llamado **"IDENTIFICACIÓN DE PROBLEMAS DE APRENDIZAJE DE PROGRAMACIÓN CON EXPLOTACIÓN DE INFORMACIÓN"** en el cual plasmaron un trabajo de investigación en progreso que se enfoco en herramientas de explotación de información para ayudar a los docentes a aplicar un proceso de descubrimiento del conocimiento en tres pasos para

diagnosticar las dificultades de aprendizaje de los alumnos (y sus causas) relacionadas a sus errores de programación [2].

Para el levantamiento de la información plantearon los siguientes interrogantes:

Paso 1. Metodología de desarrollo

- ***El estudiante aplica método de refinamientos sucesivos (SI/NO):*** *este componente buscaba diagnosticar si el estudiante había adquirido la habilidad analítica de descomponer un problema complejo en partes más simples.*
- ***El estudiante describe el significado de variables (SI/NO):*** *este componente buscaba detectar si el estudiante tenía la capacidad de explicar para qué definía los recursos que usaba en el desarrollo del programa.*
- ***El estudiante tiene estilo de escritura (SI/NO)****: este componente buscaba detectar si el estudiante tenía en cuenta el mantenimiento de un programa, es decir, que el programa podía ser leído y/o utilizado por otros programadores o incluso por sí mismo en un futuro cercano, por lo cual debía ser escrito en forma clara e inteligible.*
- ***El estudiante usa enunciados de documentación interna (SI/NO)****: este componente busca detectar si el estudiante ha adquirido la noción de programa autodocumentado, es decir, autoexplicativo.*

Paso 2. Funcionalidad del programa

- ***El estudiante descubre el algoritmo (SI/NO)****: este componente buscaba evaluar si el estudiante había desarrollado la habilidad de disponer las sentencias primitivas de programación en una secuencia lógica de acuerdo con el objetivo del problema a ser resuelto.*
- ***El estudiante generaliza la solución (SI/NO):*** *este componente buscaba detectar si el estudiante había incorporado el concepto de algoritmo como procedimiento de*

[2] JIMENEZ, Rey. IDENTIFICACIÓN DE PROBLEMAS DE APRENDIZAJE DE PROGRAMACIÓN CON EXPLOTACIÓN DE INFORMACIÓN. Buenos Aires: 2008. 8 p.

resolución de una clase de familia de problemas a la cual pertenece el problema propuesto.

- ***El estudiante utiliza delimitadores begin-end correctamente (SI/NO/NO SE PUDO EVALUAR):*** *este componente buscaba explorar si el estudiante había incorporado el concepto de abocamiento de sentencias primitivas o composición de sentencias para ser ejecutadas como sentencia única.*
- ***El estudiante descubre la naturaleza del problema (SI/NO):*** *este componente buscaba explorar si el estudiante había realizado un análisis apropiado del problema identificando el tipo o tipos de estructuras de control que permiten resolverlo y las condiciones que gobiernan la solución.*
- ***El estudiante comprende el objetivo del problema (SI/NO)****: este componente buscaba identificar si el estudiante había comprendido claramente cuál es el problema que debe resolver el programa.*
- ***El estudiante logra funcionamiento del programa (SI/NO/SI CON ALGUN ERROR)****: este componente busca indagar si el estudiante logra encontrar una solución productiva al problema obteniendo el resultado correcto.*
- ***El estudiante realiza prueba de escritorio (SI/NO)****: este componente buscaba identificar si el estudiante había completado las fases del proceso de creación de un programa evaluando si produce el resultado esperado de acuerdo con el enunciado del problema.*
- ***El estudiante asigna correctamente (SI/NO):*** *este componente buscaba detectar si el estudiante había comprendido genuinamente cómo el programador puede alterar el contenido de un espacio de memoria.*
- ***El estudiante comete errores de sintaxis (SI/NO)****: este componente buscaba controlar si el estudiante se había entrenado suficientemente en la escritura de programas en el entorno de desarrollo de Pascal.*

Paso 3. Calidad del diseño

- ***El estudiante obtiene una solución lógica (muy buena/buena/regular, mala):*** *este componente buscaba indagar si el estudiante había alcanzado la madurez necesaria para diseñar una solución de buena calidad.*
- ***El estudiante controla finalización ciclo repetitivo (sí/no/no siempre/no evaluado):*** *este componente buscaba evaluar si el estudiante había incorporado el concepto de algoritmo como un procedimiento que provee una solución en un tiempo finito verificando que una estructura iterativa efectivamente termine.*
- ***El estudiante usa conectores lógicos correctamente (sí/no/no evaluado):*** *este componente buscaba diagnosticar si el estudiante había detectado la relación entre el tipo de problema a resolver y las condiciones que controlan la repetición como reguladores de la generalización de la solución del problema.*
- ***El estudiante desarrolla un ciclo infinito (sí/no/no evaluado)****: este componente buscaba detectar si el estudiante controlaba el cuerpo de los ciclos iterativos para verificar el cambio de valor de las variables que conforman la expresión condicional asociada a la finalización del ciclo iterativo.*
- ***El estudiante comprende el objetivo del problema (SI/NO)****: este componente buscaba identificar si el estudiante había comprendido claramente cuál es el problema que debe resolver el programa.*
- ***El estudiante inicializa variables de condición antes del while (sí/no/no evaluado)****: este componente buscaba explorar si el estudiante había comprendido el funcionamiento de esta estructura iterativa como pretest, es decir, con control de condición al inicio.*
- ***El estudiante usa indistintamente while y repeat (sí/sólo Repeat/sólo While****): este componente buscaba detectar si el estudiante había adquirido la habilidad de usar correctamente cualquiera de las dos formas de estructuras iterativas gobernadas por control de condición al inicio y al final.*

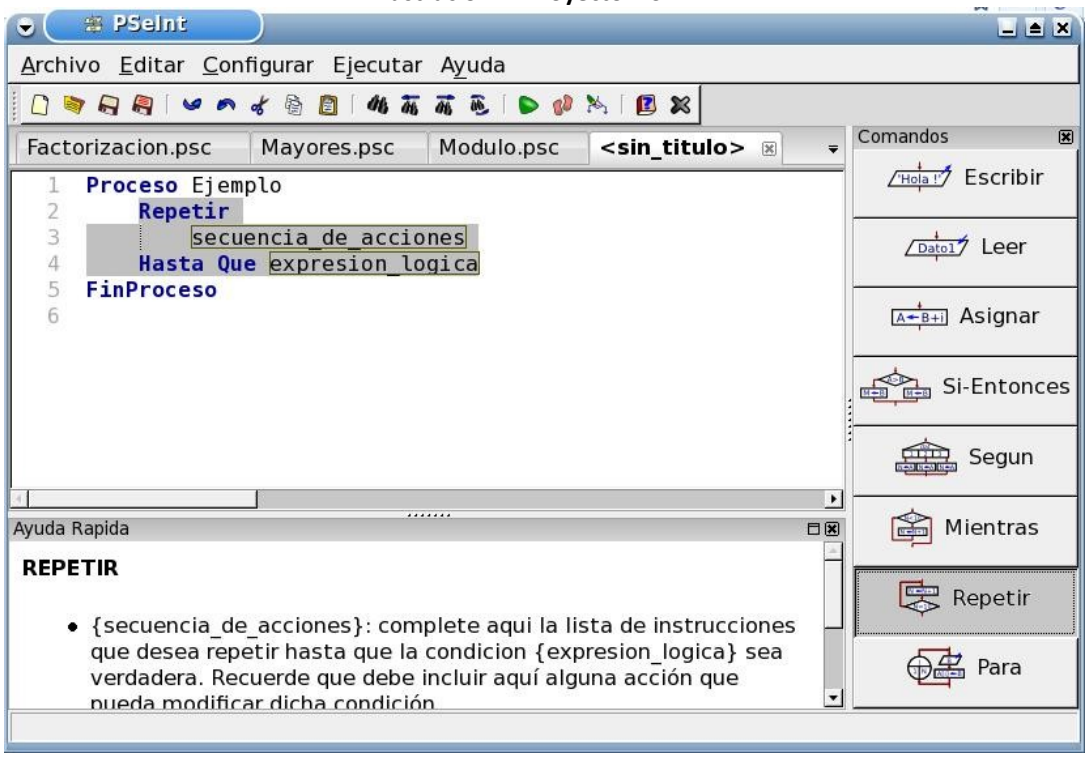
- *El estudiante usa símbolo de asignación en expresión condicional **(sí/no/no evaluado)**: este componente buscaba identificar si el estudiante había adquirido la capacidad de diferenciar el significado de un símbolo de asignación del significado de un operador de comparación.*
- ***El estudiante optimiza uso de recursos (SI/NO)**: este componente buscaba detectar si el estudiante había adquirido la habilidad de depurar el diseño del algoritmo logrando una buena solución al problema mediante el uso de recursos necesarios.*
- ***El estudiante evita repetición de código (SI/NO)**: este componente busca explorar si el estudiante ha incorporado la noción de algoritmo como una secuencia de sentencias ejecutables. Las respuestas.*
- ***El estudiante utiliza apropiadamente estructura case (SI/NO):** este componente busca detectar si el estudiante ha comprendido el funcionamiento de la estructura case como sentencia de selección múltiple y sus limitaciones de uso.*
- ***El estudiante anida estructuras selectivas (SI/NO):** este componente busca detectar si el estudiante ha logrado identificar las distintas situaciones de uso de las estructuras selectivas (con opción única, con opción doble y con opción múltiple).*

  De esta investigación tan importante se podrá tomar muchas de las variables utilizadas para realizar este proyecto ya que en su debido momento se pudo establecer donde se encuentran los principales problemas de aprendizaje de los estudiantes de lógica de programación durante el nivel introductorio.

En el año 2003 el estudiante Pablo Novara de la asignatura programación 1 del programa de Ingeniería en Informática de la Facultad de Ingeniería y Ciencias Hídricas en la Universidad Nacional del Litoral, comenzó a desarrollar utilizando *Borland C++ Builder* un software llamado **PseInt** como herramienta para aprender la lógica de programación, orientada a estudiantes sin experiencia en dicha área. Mediante la utilización de un

simple y limitado pseudo-lenguaje intuitivo y en español, permitió comenzar a comprender conceptos básicos y fundamentales de un algoritmo computacional. La última versión de este software Salió el 06 de Noviembre del 2011 y es un software que se mantiene en constante evolución.

Ilustración 2: Proyecto PSEINT



Fuente: el autor

Algunas gratas sorpresas, como los comentarios y felicitaciones recibidos de los usuarios, tanto alumnos como docentes, la mención del software en la revista española PC World o la inclusión en otras recopilaciones como cdlibre.org, uptodown.com, etc., las notas publicadas en diversos blogs y demás repercusiones muestran como este pequeño y modesto proyecto fue difundiéndose en diversos ámbitos y ganando lentamente popularidad como confirman las estadísticas de descargas o visitas al sitio web oficial. Esto genera una muy provechosa retroalimentación que lleva directo al crecimiento constante del paquete, el cual lejos aún de ser perfecto, presenta ya un funcionamiento

más acabado. El desarrollador del proyecto está convencido de que más allá de muchas coincidencias fortuitas todo este éxito se debe directa o indirectamente sin dudas a su carácter libre, por lo que seguirá promoviendo este modelo y tratando de mantener la filosofía, buscando una estrecha relación con los usuarios, liberando versiones a menudo, promoviendo la difusión de software y contenido libre y gratuito, y por sobre todo divirtiéndose y haciendo amigos en el camino[3].

En Julio de 2009 en la ciudad de Bogotá – Colombia, Jorge Alberto Villalobos Salcedo ingeniero de sistemas y computación de la universidad de Los Andes obtuvo el 10° Premio Colombiano en Informática Educativa al presentar el proyecto **"CUPI2 – UNA SOLUCIÓN INTEGRAL AL PROBLEMA DE ENSEÑAR Y APRENDER A PROGRAMAR"**

Para lograr una solución integral al problema planteado, Jorge Villalobos considero que se debían tener en cuenta los siguientes aspectos:

- Un modelo pedagógico adaptado al perfil de los estudiantes actuales. Era necesario volver a pensar a fondo la manera de enseñar a programar, utilizando para eso la estructura definida por ejes conceptuales.
- Un modelo de evaluación orientado a la verificación de las habilidades que se han debido generar en los alumnos, que tenga en cuenta los diferentes aspectos que hacen parte de la tarea de programar.
- Un conjunto de materiales y herramientas de soporte al proceso de aprendizaje.
- Un mecanismo de seguimiento a los estudiantes que midiera algunas variables que indicaran el éxito (o fracaso) del proceso de aprendizaje. Esto incluyó la definición de dichas variables, el diseño de los instrumentos que permitieran recoger la

[3] NOVARA, Pablo. PSEINT una invitación a entrar en el maravilloso mundo de la programación. Santa fe: 2003. Disponible en http://pseint.sourceforge.net/

información y la definición de los mecanismos que utilizaran lo anterior para garantizar un proceso de mejoramiento continuo.

- Un esquema de formación de profesores, que les diera una comprensión global de la problemática y que les permitiera participar de manera efectiva en el proceso de aprendizaje de los estudiantes.
- Un conjunto de herramientas de acompañamiento para los profesores, que facilitaran la planeación de las clases, la comunicación con los estudiantes, el registro de los resultados obtenidos, el reporte de las actividades y logros, el seguimiento, etc.
- Una estrategia de difusión de conocimiento para los programas de Ingeniería de Sistemas en Colombia. De acuerdo con esto, el proyecto Cupi2 busca aportar al fortalecimiento de la industria de software colombiana, a través de la generación y apropiación de conocimiento en comunidad[4].

El día 2 de Febrero de 2012 el Ingeniero Robert Sneyder Moreno Mosquera, actual candidato a Magister en Software Libre realizó una encuesta a 22 alumnos para verificar el estado en el cual finalizaron el curso de lógica de programación en ingeniería Civil de la Universidad Tecnológica del Chocó.

Teniendo en cuenta que el objetivo principal del intérprete no es sólo interpretar un buen código, sino también señalar correctamente los errores de un algoritmo incorrecto. La idea es sugerir en el año 2012 a los otros compañeros responsables de la cátedra ofrecer el software a sus futuros alumnos.

[4] VILLALOBOS, Humberto. PROYECTO CUPI2 – UNA SOLUCIÓN INTEGRAL AL PROBLEMA DE ENSEÑAR Y APRENDER A PROGRAMAR. Bogotá: 2009. Disponible en http://www.colombiaaprende.edu.co/html/mediateca/1607/articles-205832_recurso_1.pdf

6. MARCO TEORICO

Desde hace varias décadas los docentes han intentado descubrir conceptos mal comprendidos en programación de los estudiantes, en esta ocasión se realizará un estudio para detectar dónde se encuentran las principales dificultades que tienen los aprendices en el proceso para adquirir una óptima lógica de programación de algoritmos analizando los resultados de la construcción de una base de datos sobre las características de cada estudiante, su estilo y los conceptos menos comprendidos de programación. Se estudiará la metodología de desarrollo de los algoritmos escritos por cada estudiante, el funcionamiento del algoritmo y la calidad del diseño. Se detectarán reglas que establezcan relaciones entre conceptos no percibidos de programación y sus potenciales causas. Las reglas obtenidas serán utilizadas para revisar la estructura propuesta para el curso y las hipótesis de los docentes. Estos resultados indicarán que aplicando estos pasos, el docente encontrará posibles causas de los conceptos no comprendidos por los estudiantes. Luego se intentará descubrir la ponderación que cada causa tiene sobre cada concepto mal comprendido permitiendo establecer un rango de importancia de las causas de conceptos no comprendidos para proponer mejores estrategias en el proceso enseñanza-aprendizaje.

Para iniciar a verificar donde se encuentran los grandes problemas de aprendizaje de lógica de programación se abordan inicialmente los conceptos teóricos de la materia, es decir, para que un estudiante diseñe un algoritmo o un programa debe tener claras sus definiciones:

- “Un algoritmo es la solución a cualquier problema de cómputo, involucra la ejecución de una serie de acciones en orden específico. Un procedimiento para resolver un problema en términos de Las acciones a ejecutarse y el orden en el cual estas acciones deben ejecutarse”.

- “un programa es una secuencia de instrucciones mediante las cuales se ejecutan diferentes acciones de acuerdo con los datos que se estén procesando”.
- “Es un algoritmo desarrollado para ser utilizado por la computadora” o “Expresión de un algoritmo en un lenguaje preciso que puede llegar a entender una máquina de cómputo”.

Cuando Los estudiantes tienen los conceptos de algoritmo y programa totalmente claros se procede a realizar el análisis del entendimiento de las 7 fases principales para la creación de un programa en su respectivo orden: definición del Problema, Análisis del Problema, Diseño del Algoritmo, Codificación, Prueba y Depuración, Documentación y el Mantenimiento.

6.1 CONCEPTOS BÁSICOS DE PROGRAMACIÓN

Se pueden utilizar muchos lenguajes para programar una computadora. El más básico es el lenguaje de máquina, una colección de instrucciones muy detallada que controla la circuitería interna de la máquina. Este es el dialecto natural de la máquina. Muy pocos programas se escriben actualmente en lenguaje de máquina por dos razones importantes: primero, porque el lenguaje de maquina es muy incomodo para trabajar y segundo porque la mayoría de las máquinas se pide programar en diversos tipos de lenguajes, que son lenguajes de alto nivel, cuyas instrucciones son más compatibles con los lenguajes y la forma de pensar de los humanos como lo es el lenguaje c que además es de propósito general. Debido a que los programas diseñados en este lenguaje se pueden ejecutar en cualquier máquina, casí sin modificaciones. Por tanto el uso del lenguaje de alto nivel ofrece tres ventajas importantes, sencillez, uniformidad y portabilidad.

6.1.1 LENGUAJE DE PROGRAMACION

Sistema de símbolos y reglas que permite la construcción de programas con los que la computadora puede operar así como resolver problemas de manera eficaz.
Estos contienen un conjunto de instrucciones que nos permiten realizar operaciones de entrada - salida, cálculo, manipulación de textos, lógica - comparación y almacenamiento - recuperación.
Los lenguajes de programación se clasifican en:

- **Lenguaje Máquina:** Son aquellos cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la CPU pueda comprender y ejecutar el programa. Las instrucciones en lenguaje máquina se expresan en términos de la unidad de memoria más pequeña el bit (dígito binario 0 ó 1).

- **Lenguaje de Bajo Nivel (Ensamblador):** En este lenguaje las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos para las operaciones y direcciones simbólicas.

- **Lenguaje de Alto Nivel:** Los lenguajes de programación de alto nivel (BASÍC, pascal, cobol, fortran, etc.) son aquellos en los que las instrucciones o sentencias a la computadora son escritas con palabras similares a los lenguajes humanos (en general en inglés), lo que facilita la escritura y comprensión del programa.

6.1.2 ALGORITMO

La palabra algoritmo se deriva de la traducción al latín de la palabra árabe alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

La solución a cualquier problema de cómputo involucra la ejecución de una serie de acciones en orden específico. Un procedimiento para resolver un problema en términos de: a) Las acciones a ejecutarse y b) el orden en el cual estas acciones deben ejecutarse se llama algoritmo.

6.1.3 PROGRAMA

- Secuencia de instrucciones mediante las cuales se ejecutan diferentes acciones de acuerdo con los datos que se estén procesando.
- Es un algoritmo desarrollado para ser utilizado por la computadora
- Expresión de un algoritmo en un lenguaje preciso que puede llegar a entender una máquina de cómputo.

6.1.4 FASES PARA LA CREACIÓN DE UN PROGRAMA

1. Definición del Problema

Esta fase está dada por el enunciado del problema, el cual requiere una definición clara y precisa. Es importante que se conozca lo que se desea que realice la computadora; mientras esto no se conozca del todo no tiene mucho caso continuar con la siguiente etapa.

2. Análisis del Problema

Una vez que se ha comprendido lo que se desea de la computadora, es necesario definir:

- Los datos de entrada.
- Cuál es la información que se desea producir (salida)

- Los métodos y fórmulas que se necesitan para procesar los datos.

Una recomendación muy practica es ponerse en el lugar de la computadora y analizar que es lo que se necesita que ordenen y en que secuencia para producir los resultados esperados.

3. Diseño del Algoritmo

Las características de un buen algoritmo son:

- Debe tener un punto particular de inicio.
- Debe ser definido, no debe permitir dobles interpretaciones.
- Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
- Debe ser finito en tamaño y tiempo de ejecución.

4. Codificación

La codificación es la operación de escribir la solución del problema (de acuerdo a la lógica del diagrama de flujo o pseudocódigo), en una serie de instrucciones detalladas en un código reconocible por la computadora conocido como código fuente escrito en un lenguaje de programación o lenguaje de alto nivel.

5. Prueba y Depuración

Los errores humanos dentro de la programación de computadoras son muchos y aumentan considerablemente con la complejidad del problema. El proceso de identificar y eliminar errores, para dar paso a una solución sin errores se le llama depuración.

La prueba consiste en la captura de datos hasta que el programa no presente errores (los más comunes son los sintácticos y lógicos).

6. Documentación

Es la guía o comunicación escrita es sus variadas formas, ya sea en enunciados, procedimientos, dibujos o diagramas.

A menudo un programa escrito por una persona, es usado por otra. Por ello la documentación sirve para ayudar a comprender o usar un programa o para facilitar futuras modificaciones (mantenimiento).

La documentación se divide en las siguientes partes:

- **Documentación Interna:** Son los comentarios o mensajes que se añaden al código fuente para hacer más claro el entendimiento de un proceso.
- **Documentación Externa:** Se define en un documento escrito los siguientes puntos:

  Descripción del Problema

  Nombre del Autor

  Algoritmo (diagrama de flujo o pseudocódigo)

  Diccionario de Datos

  Código Fuente (programa)
- **Manual del Usuario:** Describe paso a paso la manera cómo funciona el programa, con el fin de que el usuario obtenga el resultado deseado.

7. Mantenimiento

Se lleva a cabo después de terminado el programa, cuando se detecta que es necesario hacer algún cambio, ajuste o complementación al programa para que siga trabajando de manera correcta. Se debe tener en cuenta que para realizar este trabajo se requiere que el programa este correctamente documentado.

6.1.5 LOS DATOS Y OPERACIONES BÁSICAS

1. IDENTIFICADOR. Un identificador es una serie de caracteres formados por letras, dígitos y el carácter subrayado (_) que no inicie con dígito, así mismo es el nombre que damos a todo lo que manipulamos dentro de un programa, Por ejemplo variables, constantes, funciones, tipos definidos por el usuario etc.

2. TIPOS DE DATOS. Todos los datos tienen un tipo asociado con ellos. Un dato puede ser un simple carácter, tal como b, un valor entero tal como 35. El tipo de dato determina la naturaleza del conjunto de valores que puede tomar una variable.

3. VARIABLES. Una variable es un identificador que puede tomar diferentes valores dependiendo del tipo que esta se declare.

Una variable es un identificador que puede cambiar de valor durante la ejecución de un programa.

Una variable es una posición de memoria donde se puede almacenar un valor para uso de un programa.

4. INICIALIZACIÓN DE VARIABLES. Inicializar una variable es darle un valor después que se ha declarado pero antes de que se ejecuten las sentencias en las que se emplea.

5. CONSTANTES. Constantes son los valores que no pueden ser modificados. En C, pueden ser de cualquier tipo de datos. Además de los ejemplificados anteriormente, Podemos

crear constantes de caracteres con barra invertida. Estos corresponden a los caracteres que son imposibles introducir desde el teclado.

6. OPERADORES. Un operador es un símbolo que indica al compilador que realice manipulaciones lógicas o matemáticas específicas.

Los operadores del mismo nivel de precedencia son evaluados por el compilador de izquierda a derecha. Por supuesto, se puede utilizar paréntesis para ordenar la evaluación.
También, conviene utilizar paréntesis para hacer más claro el orden en que se producen las evaluaciones, tanto para la persona que lo elabora o para los que después tengan que seguir el programa.

Operadores Lógicos:
Estos operadores se utilizan para establecer relaciones entre valores lógicos y pueden ser resultado de una expresión relacional.

- And Y
- Or O
- Not Negación

Operadores de Asignación.
Los operadores de asignación se utilizan para formar expresiones de asignación, en las que se asigna el valor de una expresión a un identificador.
Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.

Una expresión consta de operadores y operandos. Según sea el tipo de datos que manipulan, se clasifican las expresiones en:

- Aritméticas
- Relacionales
- Lógicas

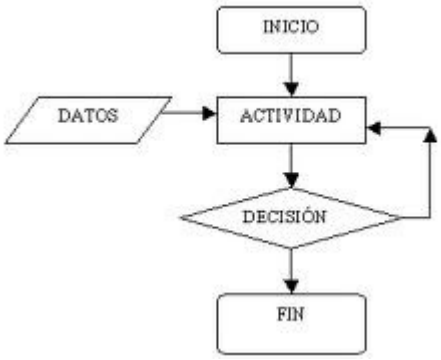
7. PALABRAS RESERVADAS. Son palabras que tienen un significado especial para el lenguaje y no se pueden utilizar como identificadores.

8. COMENTARIOS. Los comentarios pueden aparecer en cualquier parte del programa, mientras estén situados entre los delimitadores /* comentario */. Los comentarios son útiles para identificar los elementos principales de un programa o para explicar la lógica subyacente de estos.

6.1.6 DIAGRAMAS DE FLUJO

Un diagrama de flujo es la representación gráfica de un algoritmo. También se puede decir que es la representación detallada en forma gráfica de cómo deben realizarse los pasos en la computadora para producir resultados.

Ilustración 3: Ejemplo de Un Diagrama de Flujo



Fuente: web http://www.monografias.com/trabajos38/programacion/programacion2.shtml

6.1.7 PSEUDOCÓDIGO

Mezcla de lenguaje de programación y español (o inglés o cualquier otro idioma) que se emplea, dentro de la programación estructurada, para realizar el diseño de un programa. En esencial, el pseudocódigo se puede definir como un lenguaje de especificaciones de algoritmos.

Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado. El pseudocódigo utiliza palabras que indican el proceso a realizar.

El pseudocódigo se concibió para superar las dos principales desventajas del Diagrama de Flujo: el diagrama de flujo es lento de crear y difícil de modificar sin un nuevo redibujo. Por otra parte el pseudocódigo es más fácil de utilizar ya que es similar al lenguaje natural.

6.1.8 PROGRAMACIÓN ESTRUCTURADA

Método disciplinado de escribir programas que sean claros, que se demuestren que son correctos y fáciles de modificar

Un programa se compone de:

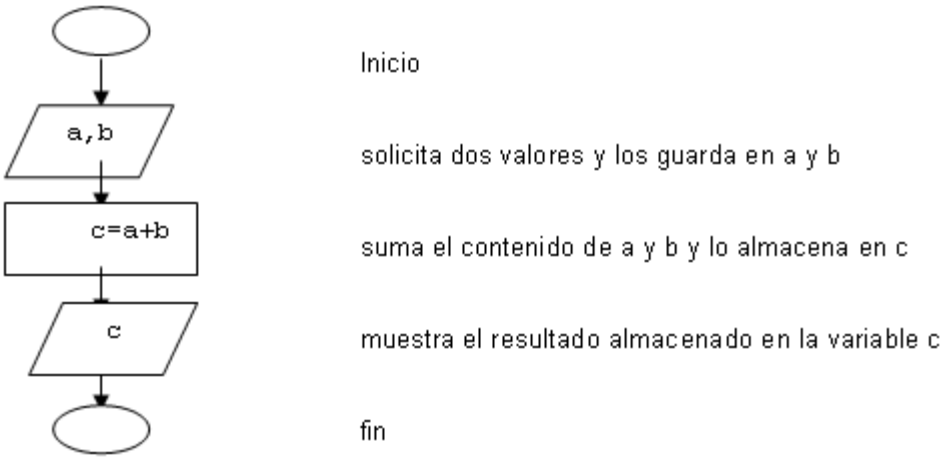
a. **Estructuras de datos.** Los hechos reales, representación en forma de datos, manera en que se organizan los datos.
b. **Operaciones primitivas elementales.** Son acciones que se ejecutan sobre los datos para transformarlos en información.

c. **Estructuras de control.** Son los métodos que existen para dirigir el flujo de acciones que la computadora deberá ejecutar sobre los datos manejados por el programa.

Estructura de control secuencial

La computadora ejecutará automáticamente enunciados uno después del otro, en el orden en el cual se han escrito de inicio a fin.

Ilustración 4: Ejemplo de Estructura de Control Secuencial



Fuente: web http://www.monografias.com/trabajos38/programacion/programacion2.shtml

Estructura de control selectiva

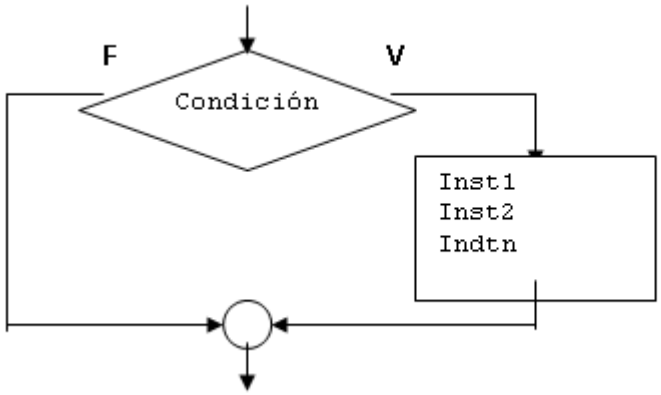
Existen tres tipos de estructuras de control selectivas, estas se basan en una condición o en una opción para decidir la parte del programa por el que pasará.

a. Simple
b. Doble o compuesta
c. Múltiple

Selectiva simple.

Evalúa una condición, si esta es verdadera ejecuta la acción o acciones especificadas, si es falsa no realiza ninguna acción.

Ilustración 5: Ejemplo Estructura Selectiva Simple




```
Pseudocódigo

Si condición
   Inicio
      Inst1
      Inst2
      Instn
   fin
```

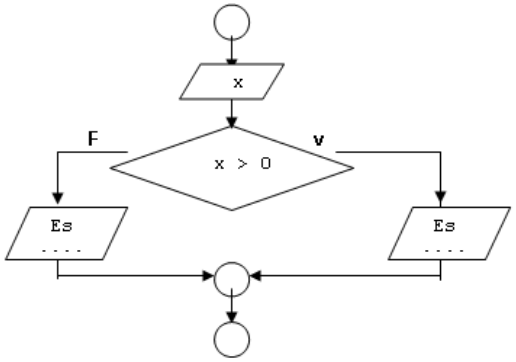
Fuente: web http://www.monografias.com/trabajos38/programacion/programacion2.shtml

Nota: Si existe sola una instrucción o sentencia dentro de la condición no es necesario marcarlos con inicio y fin, en caso contrario si, como se muestra en el diagrama anterior.

Selectiva doble o compuesta.

Evalúa una condición, si esta es verdadera ejecuta la acción o acciones especificadas, si es falsa ejecuta otra acción o acciones.

Ilustración 6: Ejemplo Estructura Selectiva Doble Compuesta




```
Inicio
      Entero x

      Leer  x


      Si x >0


      Imprimir "Es positivo"
      Sino
      Imprime "Es negativo"


fin
```

Fuente: web http://www.monografias.com/trabajos38/programacion/programacion2.shtml

Nota: Si existe sola una instrucción o sentencia dentro de la condición no es necesario marcarlos con inicio y fin como en este caso que la condición fue falsa, en caso contrario si, en este ejemplo cuando la condición fue verdadera.

Ejemplo: Imprimir si un número es positivo o negativo

7. ANALISIS DEL SISTEMA

Es un conjunto o disposición de procedimientos o programas relacionados de manera que juntos forman una sola unidad.

Un conjunto de hechos, principios y reglas clasificadas y dispuestas de manera ordenada mostrando un plan lógico en la unión de las partes.

Un método, plan o procedimiento de clasificación para hacer algo.

También es un conjunto o arreglo de elementos para realizar un objetivo predefinido en el procesamiento de la Información.

Esto se lleva a cabo teniendo en cuenta ciertos principios:

- Debe presentarse y entenderse el dominio de la información de un problema.
- Definir las funciones que debe realizar el Software.
- Representar el comportamiento del software a consecuencias de acontecimientos externos.
- Dividir en forma jerárquica los modelos que representan la información, funciones y comportamiento.

El proceso debe partir desde la información esencial hasta el detalle de la Implementación.

7.1 DEFINICIÓN DE TÉRMINOS

Orientación a Objetos

Es una forma de estudiar el universo como un conjunto de objetos interrelacionados. Al estudiar un problema específico se delimitan los objetos relevantes al problema (Enfoque).

Objeto

Componente del Universo que presenta tres cualidades principales.

- **Identidad**

  Aquella cualidad que permite a un objeto, diferenciarse de entre todos los demás.

- **Características**

  Son los datos que describen al objeto.

- **Comportamiento**

  Son las acciones que puede realizar el objeto.

Clase

Es un concepto que representa las características y comportamientos comunes a un conjunto de objetos, que son los miembros de la clase, las clases presentan tres cualidades principales.

- **Nombre**

  Permite que una clase se diferencie de las otras.

- **Atributos**

  Son variables que permiten almacenar las características comunes de los objetos.

- **Operaciones**

  Son las acciones que pueden realizar los objetos que pertenecen a la clase.

7.2 REQUERIMIENTOS

Se requiere un sistema que permita:

- A la Aplicación, Interpretar el código escrito por el usuario.
- A la Aplicación, Compilar el código escrito por el usuario.
- A la Aplicación, Ejecutar el código escrito por el usuario.
- Al usuario, tener acceso a la interfaz principal de la aplicación.
- Al usuario, crear un nuevo programa en seudocódigo.
- Al usuario, Recibir errores de compilación del programa generado.
- Al usuario, obtener la ejecución del programa generado.
- Al usuario, obtener el código fuente resultante en C, JAVA y VISUAL BASÍC de la conversión del seudocódigo.

7.3. CASOS DE USO

Los casos de uso se emplean para capturar el comportamiento deseado del sistema en desarrollo, sin tener que especificar cómo se implementa ese Comportamiento.

Un caso de uso especifica unas secuencias de acciones incluyendo variantes que el sistema pueda llevar a cabo y que producen un resultado observable de valor para un actor concreto.

Los casos de uso se utilizan para contener un conjunto de responsabilidades, luego las responsabilidades son asignadas o distribuidas entre las clases.

7.3.1 EVENTOS DEL NEGOCIO

Cada evento del negocio se puede tomar como un caso de uso, y puede ser la forma en que un actor externo interactúa con el sistema.

7.3.1.1. Crear Nuevo Programa

El Usuario desea crear un nuevo programa en seudocódigo.

7.3.1.2. Guardar Programa

El Usuario desea guardar el programa realizado en seudocódigo.

7.3.1.3. Abrir Programa

El Usuario desea abrir un programa existente escrito en seudocódigo.

7.3.1.4. Cerrar Programa

El Usuario desea Cerrar el programa de la pestaña activa.

7.3.1.5. Ejecutar Programa

El Usuario desea Compilar y Ejecutar el programa en seudocódigo.

7.3.1.6. Exportar Programa

El Usuario desea conocer y exportar el código del programa escrito en otro lenguaje de programación.

7.3.1.7 Programar

El Usuario desea editar el código de un programa en seudocódigo.

7.3.1.8. Declarar Variables

El Usuario desea insertar el código para declarar variables.

7.3.1.9. Insertar ESCRIBIR

El Usuario desea insertar el código para la instrucción de salida a pantalla ESCRIBIR.

7.3.1.10. Insertar LEER

El Usuario desea insertar el código para la instrucción de Entrada de teclado LEER.

7.3.1.11. Asígnar a variable

El Usuario desea insertar el código para realizar la asígnación de valores a una variable.

7.3.1.12. Insertar SI

El Usuario desea insertar el código para definir la estructura selectiva SI – SINO – FINSI.

7.3.1.13. Insertar CASOS

El Usuario desea insertar el código para definir la estructura selectiva CASOS.

7.3.1.14. Insertar MIENTRAS

El Usuario desea insertar el código para definir la estructura repetitiva MIENTRAS.

7.3.1.15. Insertar HAGA – MIENTRAS

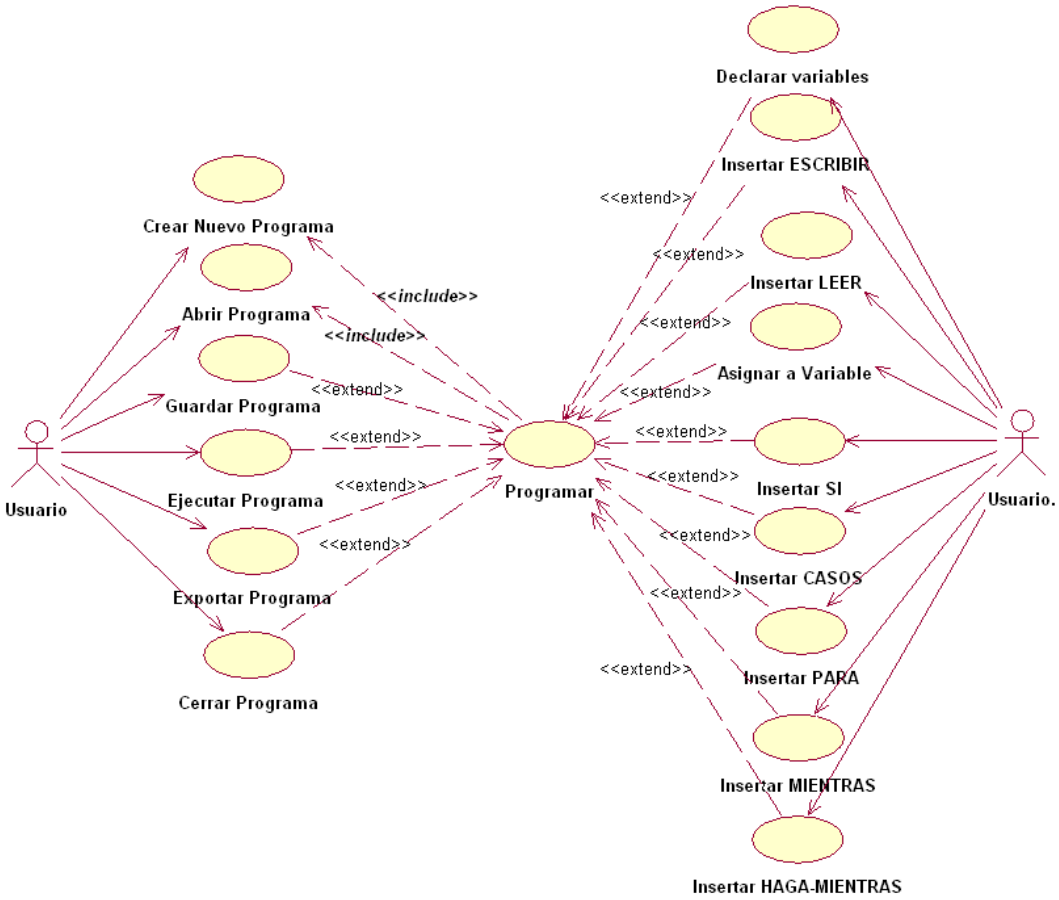
El Usuario desea insertar el código para definir la estructura repetitiva HAGA – MIENTRAS.

7.3.1.16. Insertar PARA

El Usuario desea insertar el código para definir la estructura repetitiva PARA.

7.3.2 DIAGRAMA DE CASOS DE USO

Ilustración 7: Diagrama general de casos de uso



Fuente: El Autor

7.3.3 ESPECIFICACIÓN DE CASOS DE USO

Caso de uso

Colocar el nombre del caso de uso.

Actores

Se coloca la lista de actores que intervienen, se debe identificar al iniciador del estado de uso.

Propósito

Es la intención del caso de uso, objetivo general.

Resumen

Una narración resumida del caso de uso.

Tipo

Si es primario, secundario u opcional.

- **Primario**

Lo que normalmente sucede

- **Secundario**

Formas alterna de cómo pueden suceder los eventos diferentes del caso primero.

- **Opcional**

Un caso de uso que podría no suceder.

Esencial o real.

- **Esencial**

Es una descripción tan general que sirve para todas las formas de implementación del caso de uso.

- **Real**

Describe una determinada implementación del caso de uso.

Luego sigue la sección **Curso normal de los eventos.**

La sección se divide en 2 columnas; la de la izquierda son Acciones de los actores, la de la derecha son las Acciones del sistema.

Las acciones se enumeran consecutivamente.

El resumen inicial se divide en pasos o en eventos.

Luego sigue la sección: **Cursos alternativos.**

7.3.3.1. Crear Nuevo Programa

Especificación Expandida

Tabla 1: Especificación caso de uso Crear Nuevo Programa

Caso de Uso	Crear Nuevo Programa
Actores	Usuario
Propósito	Crear un nuevo Programa en Seudocódigo
Resumen	Un Usuario desea crear un nuevo programa en Seudocódigo.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. El Usuario ejecuta la aplicación.	
2. El Usuario selecciona la opción crear	3. El sistema abre una nueva pestaña con las palabras reservadas INICIO – FIN para

Nuevo Programa	comenzar la edición del código del programa.
4. El Usuario queda interactuando con la interfaz para programar.	

Fuente: El Autor

7.3.3.2. Guardar Programa

Especificación Expandida

Tabla 2: Especificación caso de uso Guardar Programa

Caso de Uso	Guardar Programa
Actores	Usuario
Propósito	Guardar el código del programa en Seudocódigo creado en la herramienta SeuProg.
Resumen	Un Usuario desea Guardar los cambios hechos al código del programa en seudocódigo.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. El Usuario ejecuta la aplicación.	

2. El Usuario abre y edita un programa Nuevo o un programa Existente.	
3. El Usuario selecciona la opción Guardar Programa.	4. El sistema muestra el cuadro de dialogo para ingresar el nombre del programa y seleccionar el directorio donde se desea guardar el programa.
5. El Usuario selecciona el directorio, asígna el nombre del archivo y selecciona la opción Guardar Programa.	6. El sistema guarda el archivo con el nombre y directorio seleccionado por el usuario.

Cursos Alternativos

Línea 6: El Nombre del archivo ya existe en el directorio seleccionado, se muestra el error y se regresa a la línea 4.

Fuente: El Autor

7.3.3.3. Abrir Programa

Especificación Expandida

Tabla 3: Especificación caso de uso Abrir Programa

Caso de Uso	Abrir Programa
Actores	Usuario
Propósito	Abrir un programa existente y que el código del programa se encuentre en Seudocódigo creado en la herramienta SeuProg.
Resumen	Un Usuario desea Abrir un programa existente para editarlo y/o ejecutarlo en SeuProg.

Tipo	Secundario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. El Usuario ejecuta la aplicación.	
2. El Usuario selecciona la opción Abrir Programa.	3. El sistema muestra el cuadro de dialogo para seleccionar el directorio y Archivo que se desea abrir.
4. El Usuario busca el archivo, lo selecciona y escoge la opción abrir.	5. El sistema abre una nueva pestaña con el contenido del archivo para comenzar la edición del código del programa.

Fuente: El Autor

7.3.3.4. Cerrar Programa

Especificación Expandida

Tabla 4: Especificación caso de uso Cerrar Programa

Caso de Uso	Cerrar Programa
Actores	Usuario
Propósito	Cerrar un programa abierto.

Resumen	Un Usuario desea Cerrar un programa abierto.
Tipo	Secundario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. El Usuario ejecuta la aplicación.	
2. El Usuario selecciona la opción Abrir programa o Crear Nuevo Programa *(Ver casos de usos Crear Nuevo Programa y Abrir Programa)*	
4. El usuario selecciona la opción Cerrar Programa.	5. el sistema muestra un cuadro de dialogo para confirmar si desea guardar el archivo a cerrar.
7. El Usuario selecciona el directorio, asígna el nombre del archivo y selecciona la opción Guardar Programa.	6. Si el Usuario Selecciona la Opción SI, El sistema muestra un cuadro de dialogo para seleccionar el directorio y Nombre del archivo a guardar.

Fuente: El Autor

7.3.3.5. Ejecutar Programa

Especificación Expandida

Tabla 5: Especificación caso de uso Ejecutar Programa

Caso de Uso	Ejecutar Programa
Actores	Usuario
Propósito	Ejecutar un programa abierto.
Resumen	Un Usuario desea Ejecutar un programa que tiene abierto. El usuario organiza el código del programa y selecciona la opción EJECUTAR.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. El usuario construye el código del programa *(ver caso de uso Programar)*.	
2. El usuario selecciona la opción EJECUTAR.	3. El sistema compila el código del programa verificando errores.
	4. Si el programa No tiene errores realiza la traducción del programa al lenguaje de Programación.
6. El Usuario puede interactuar con el programa creado.	5. El Sistema ejecuta el programa generado por el Usuario

Cursos Alternativos

Línea 4: El programa tiene Errores, se muestra el error y se regresa a la línea 1.

Fuente: El Autor

7.3.3.6. Exportar Programa

Especificación Expandida

Tabla 6: Especificación caso de uso Exportar Programa

Caso de Uso	Exportar Programa
Actores	Usuario
Propósito	Traducir y exportar a lenguaje de programación el código creado por el usuario
Resumen	El Usuario y/o Aplicación desean traducir y exportar a lenguaje C o JAVA el código creado.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. El usuario selecciona la opción Exportar Programa.	2. El sistema muestra el cuadro de dialogo para digitar el nombre del archivo y seleccionar el lenguaje al que desea exportar el programa.
3. El usuario selecciona el lenguaje de	4. El Sistema realiza la traducción línea a

programación y el nombre del archivo y da clic en el botón exportar.	línea de las instrucciones del programa en seudocódigo al lenguaje de programación seleccionado por el usuario.
	5. Si el programa no tiene errores El sistema muestra al usuario que el archivo fue exportado correctamente.
6. el usuario puede revisar el archivo exportado.	

Cursos Alternativos

Línea 5. El programa tiene errores, el sistema muestra el mensaje de error al usuario, el usuario debe editar el código. Regresa a la línea 1.

Fuente: El Autor

7.3.3.7. Programar

Especificación Expandida

Tabla 7: Especificación caso de uso Programar

Caso de Uso	Programar
Actores	Usuario
Propósito	Crear un programa en seudocódigo utilizando las herramientas de la aplicación Seuprog.
Resumen	El Usuario desea editar el código de un programa en seudocódigo.

Tipo	Primario y Esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. El usuario abre un programa nuevo *(Ver caso de Uso Crear Programa)* o abrir un programa *(Ver caso de uso Abrir Programa)*	2. El sistema prepara la interfaz del área de programación y los comandos que pueden ser utilizados.
	3. El sistema Ubica el cursor en el Cuadro de texto perteneciente al área del programa
9. El usuario edita del programa en seudocódigo Digitando el texto o Haciendo clic en los botones que ofrece la interfaz: • Declarar Variables *(Ver caso de Uso Declarar Variables)* • *Insertar Escribir (Ver Caso de Uso Insertar ESCRIBIR)* • Insertar LEER (Ver Caso de Uso Insertar LEER) • Asígnar a Variable (Ver Caso de Uso Asígnar a Variable) • Insertar SI (Ver Caso de Uso Insertar SI)	

• Insertar Casos (Ver Caso de Uso Insertar Casos) • Insertar PARA (Ver Caso de Uso Insertar PARA) • Insertar MIENTRAS (Ver Caso de Uso Insertar MIENTRAS) • Insertar HAGA-MIENTRAS (Ver Caso de Uso Insertar HAGA-MIENTRAS)	
5. El usuario ha creado su propio programa en seudocódigo.	

Fuente: El Autor

7.3.3.8. Declarar Variables

Especificación Expandida

Tabla 8: Especificación caso de uso Declarar Variables

Caso de Uso	Declarar Variables
Actores	Usuario
Propósito	El usuario desea insertar el código para declarar variables.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para declarar variables.
Tipo	Primario y esencial
Referencias	

Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón Declarar Variables.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **Var <tipo_variable>** <nombre_variables>**;**
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.9. Insertar ESCRIBIR

Especificación Expandida

Tabla 9: Especificación caso de uso Insertar ESCRIBIR

Caso de Uso	Insertar ESCRIBIR
Actores	Usuario
Propósito	El usuario desea insertar el código para la instrucción Escribir.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para la instrucción Escribir.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón Insertar ESCRIBIR.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **Escribir** <"Texto", variables> **;**
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.10. Insertar LEER

Especificación Expandida

Tabla 10: Especificación caso de uso Insertar LEER

Caso de Uso	Insertar LEER
Actores	Usuario
Propósito	El usuario desea insertar el código para la Instrucción LEER.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para la instrucción LEER.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón insertar LEER.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código:

	Leer <Nombre_Variables(s)>;
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.11. Asígnar a variable

Especificación Expandida

Tabla 11: Especificación caso de uso Asígnar a variable

Caso de Uso	Asígnar a variable
Actores	Usuario
Propósito	El usuario desea insertar el código para asígnación de valores a una variable.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para la asígnación de valores a una variable.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	**Respuesta del Sistema**
1. Un usuario hace clic en el botón Asígnar a variable.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **<Nom_vararіable> =** <Expresion>;
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.12. Insertar SI

Especificación Expandida

Tabla 12: Especificación caso de uso insertar SI

Caso de Uso	Insertar SI
Actores	Usuario
Propósito	El usuario desea insertar el código para la estructura Selectiva SI.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para la estructura selectiva SI.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón insertar **SI**.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **SI (**<Condicion(es)>**)** <Acciones_Si_Verdadero> **SINO** <Acciones_Si_Falso> **FINSI**
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.13. Insertar CASOS

Especificación Expandida

Tabla 13: Especificación caso de uso Insertar CASOS

Caso de Uso	Insertar CASOS
Actores	Usuario
Propósito	El usuario desea insertar el código para utilizar la estructura selectiva CASOS.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para utilizar la estructura selectiva CASOS.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón Insertar CASOS.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **Según** <variable_numerica> **Hacer** **uméri_1:** secuencia_de_acciones_1 **uméri_2:** secuencia_de_acciones_2 **uméri_3:** secuencia_de_acciones_3 **De Otro Modo:** secuencia_de_acciones_dom **FinSegun**
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.14. Insertar MIENTRAS

Especificación Expandida

Tabla 14: Especificación caso de uso Insertar MIENTRAS

Caso de Uso	Insertar MIENTRAS
Actores	Usuario
Propósito	El usuario desea insertar el código para utilizar la estructura repetitiva MIENTRAS.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para la estructura repetitiva MIENTRAS.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón Insertar MIENTRAS.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **MIENTRAS (**<Condiciones>**)** <Acciones_Si_Verdadero> **FINMIENTRAS**
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.15. Insertar HAGA-MIENTRAS

Especificación Expandida

Tabla 15: Especificación caso de uso Insertar HAGA-MIENTRAS

Caso de Uso	Insertar HAGA-MIENTRAS
Actores	Usuario
Propósito	El usuario desea insertar el código para la estructura repetitiva Haga – Mientras.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para la estructura repetitiva Haga – Mientras.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón Insertar HAGA-MIENTRAS.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **HAGA** <Acciones_Si_Verdadero> **MIENTRAS (**<Condiciones>**)**
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.3.3.16. Insertar PARA

Especificación Expandida

Tabla 16: Especificación caso de uso Insertar PARA

Caso de Uso	Insertar PARA
Actores	Usuario
Propósito	El usuario desea insertar el código para la estructura Repetitiva PARA.
Resumen	Un usuario está creando un programa en seudocódigo y desea insertar el código para la estructura Repetitiva PARA.
Tipo	Primario y esencial
Referencias Cruzadas	

Curso Normal de los Eventos

Acción de los Actores	Respuesta del Sistema
1. Un usuario hace clic en el botón Insertar PARA.	2. el sistema inserta en la posición actual del cursor dentro del código del programa las siguientes líneas de código: **Para <**variable_numerica>=<valor_inicial> **Hasta** <valor_final> **Con Paso** <paso> **Hacer** <secuencia_de_acciones> **FinPara**
3. El usuario edita el código insertado por la aplicación dependiendo de sus requerimientos.	

Fuente: El Autor

7.4. MODELO DINÁMICO

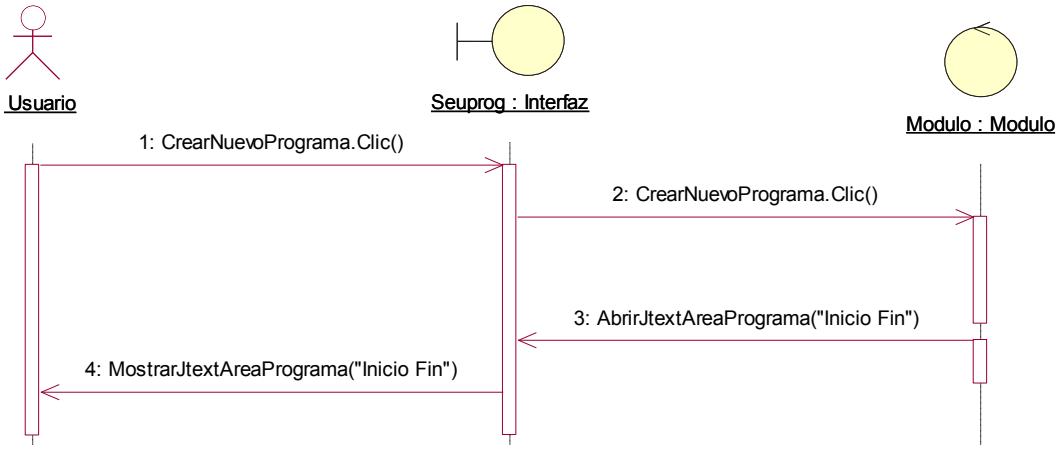
7.4.1. DIAGRAMAS DE SECUENCIA

Su objetivo es mostrar la secuencia de eventos que sucede en un determinado escenario de un caso de uso ordenados en el tiempo. El tiempo corre de arriba hacia abajo. El diagrama muestra un conjunto de objetos que intervienen entre sí para realizar un caso de uso.

7.4.1.1. Diagrama de Secuencia Crear Nuevo Programa

Curso Normal de los Eventos

Ilustración 8: Diagrama de secuencia Crear Nuevo Programa (curso normal)

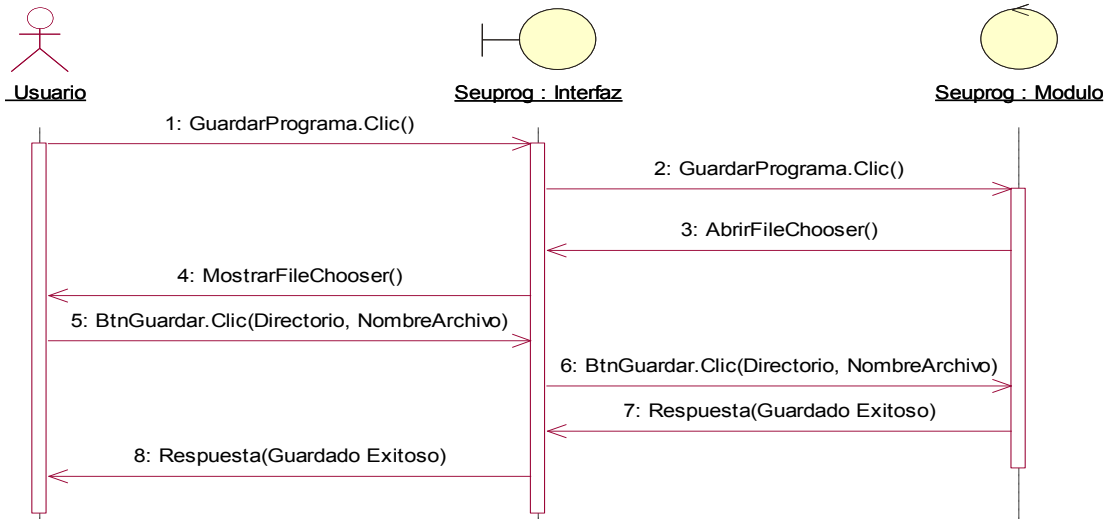


Fuente: El Autor

7.4.1.2. Diagrama de Secuencia Guardar Programa

Curso Normal de los Eventos

Ilustración 9: Diagrama de secuencia Guardar Programa (curso normal)

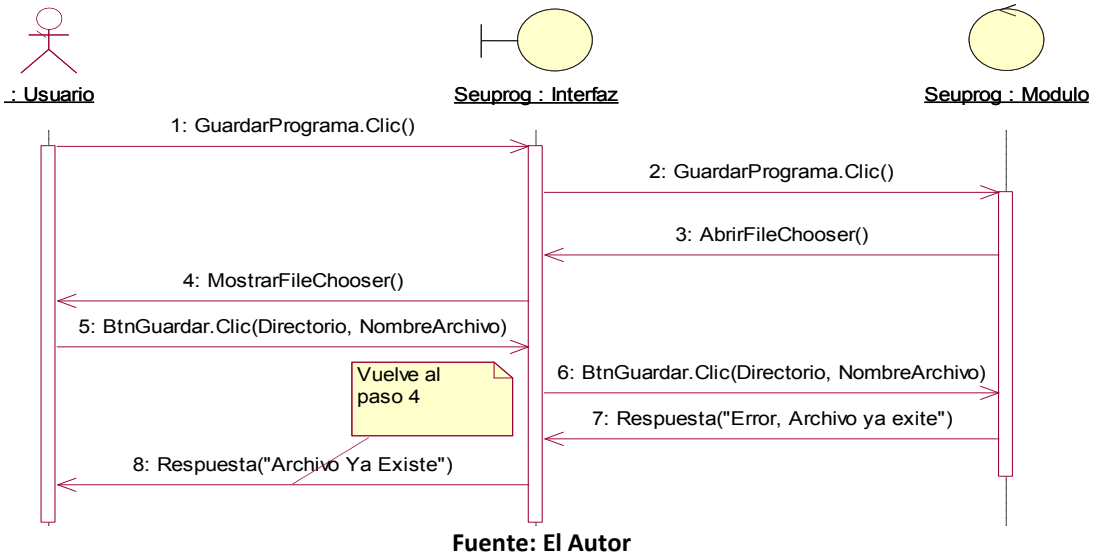


Fuente: El Autor

Cursos Alternativos de los Eventos

Curso Alternativo 1

Ilustración 10: Diagrama de secuencia Guardar Programa (curso alternativo 1)

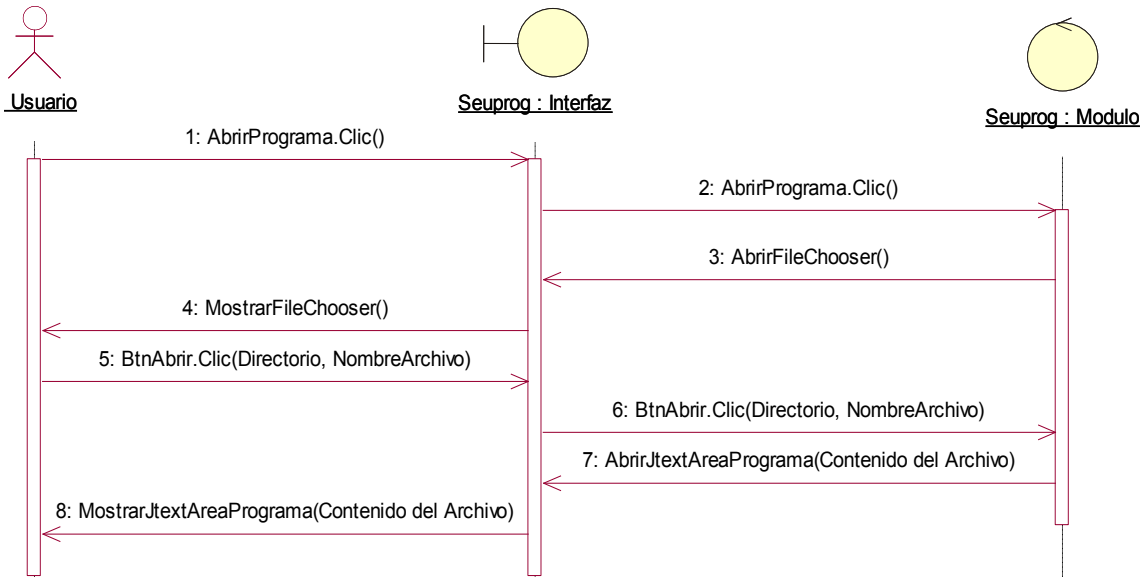


Fuente: El Autor

7.4.1.3. Diagrama de Secuencia Abrir Programa

Curso Normal de los Eventos

Ilustración 11: Diagrama de secuencia Abrir Programa (curso normal)

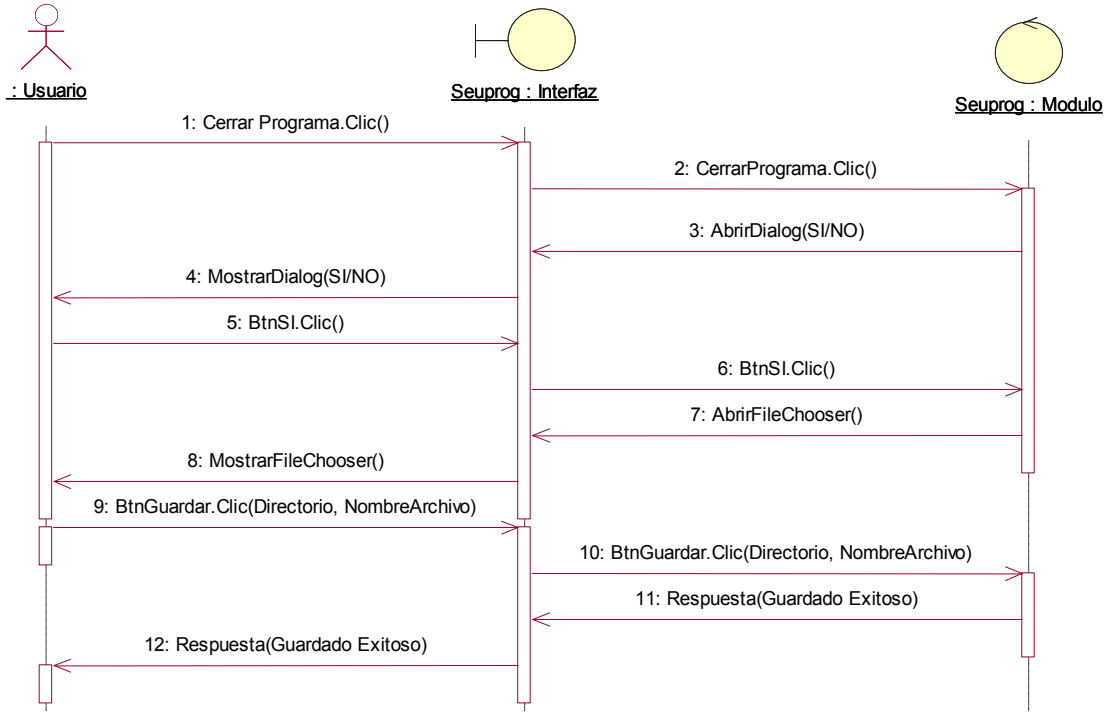


Fuente: El Autor

7.4.1.4. Diagrama de Secuencia Cerrar Programa

Curso Normal de los Eventos

Ilustración 12: Diagrama de secuencia Cerrar Programa (curso normal)

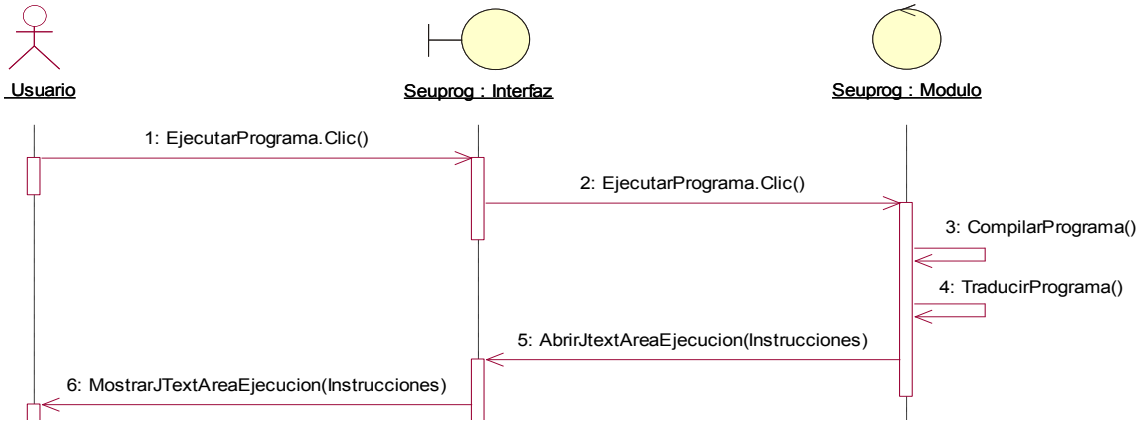


Fuente: El Autor

7.4.1.5. Diagrama de Secuencia Ejecutar Programa

Curso Normal de los Eventos

Ilustración 13: Diagrama de secuencia Ejecutar Programa (curso normal)

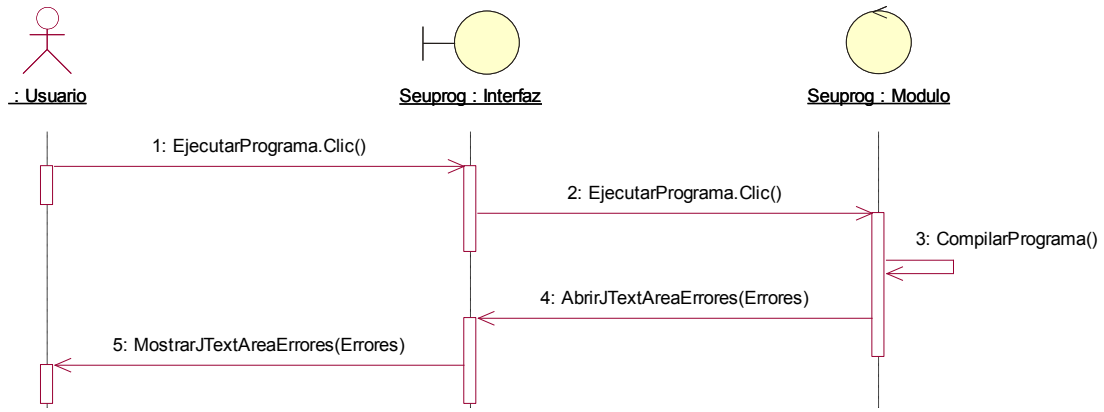


Fuente: El Autor

Cursos Alternativos de los Eventos

Curso Alternativo 1

Ilustración 14: Diagrama de secuencia Ejecutar Programa (curso alternativo 1)

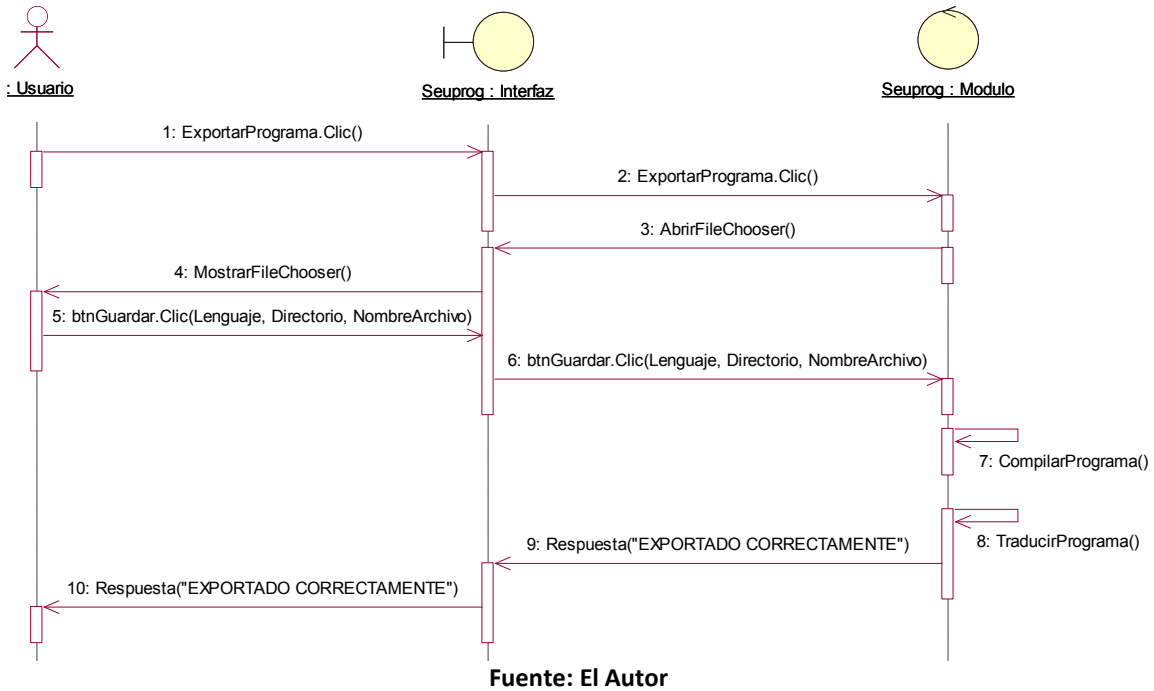


Fuente: El Autor

7.4.1.6. Diagrama de Secuencia Exportar Programa

Curso Normal de los Eventos

Ilustración 15: Diagrama de secuencia Exportar Programa (curso normal)

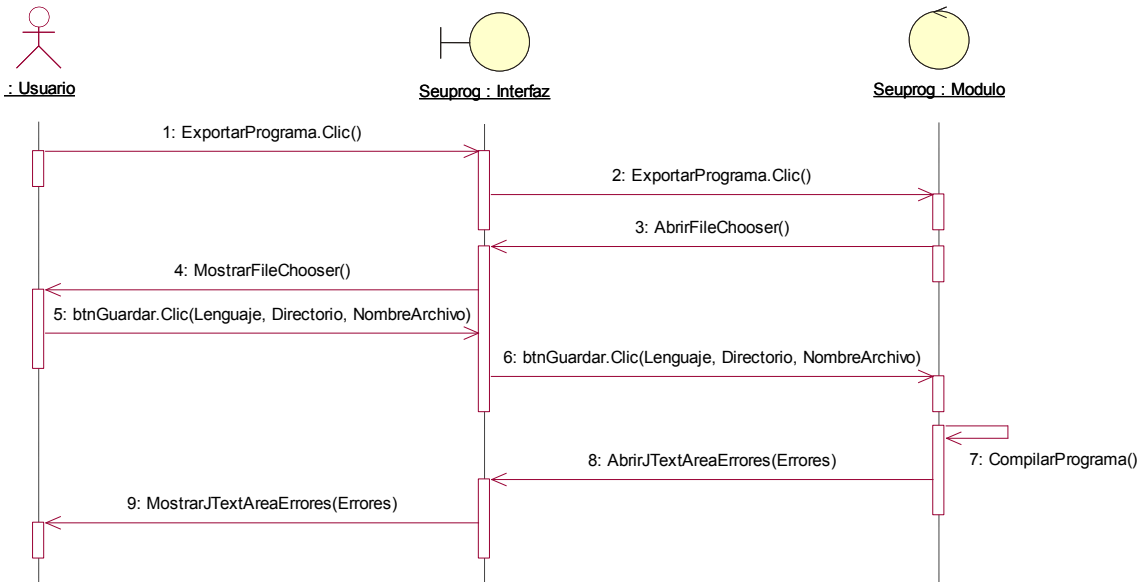


Fuente: El Autor

Cursos Alternativos de los Eventos

Curso Alternativo 1

Ilustración 16: Diagrama de secuencia Exportar Programa (curso alternativo 1)

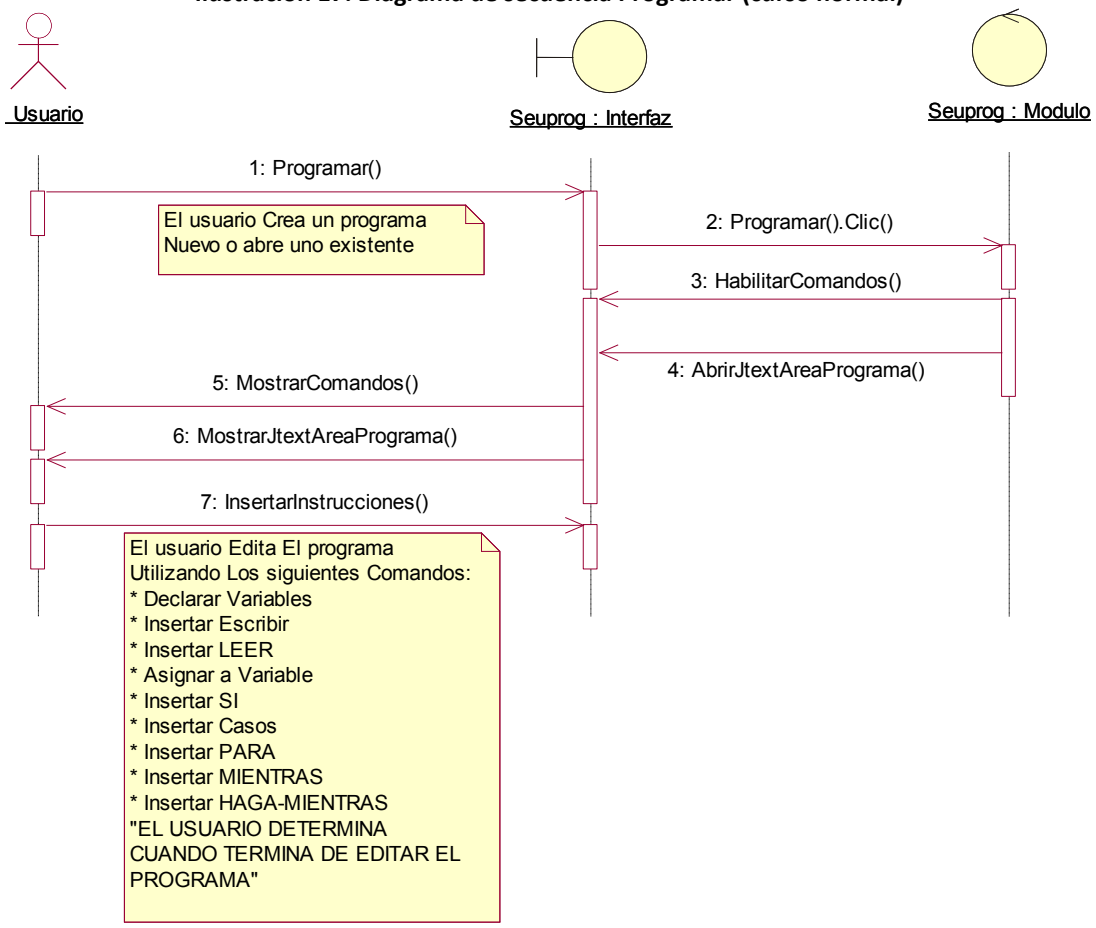


Fuente: El Autor

7.4.1.7. Diagrama de Secuencia Programar

Curso Normal de los Eventos

Ilustración 17: Diagrama de secuencia Programar (curso normal)

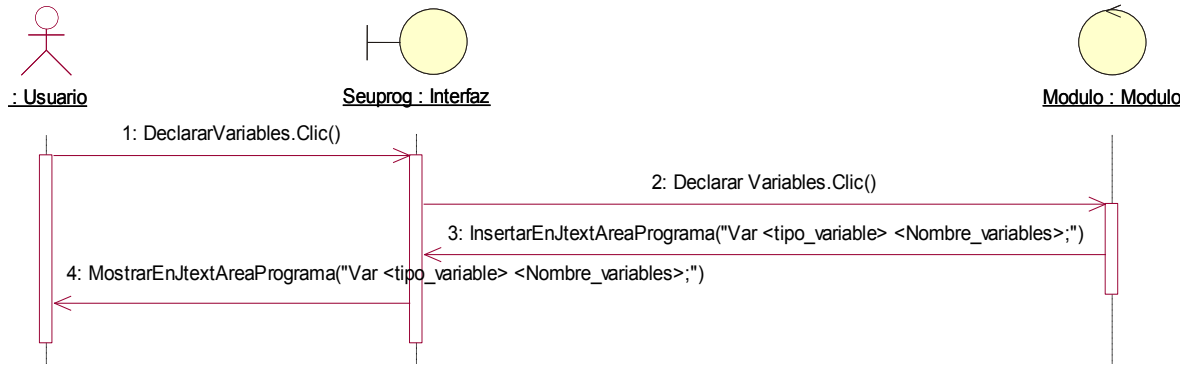


Fuente: El Autor

7.4.1.8. Diagrama de Secuencia Declarar Variables

Curso Normal de los Eventos

Ilustración 18: Diagrama de secuencia Declarar Variables (curso normal)

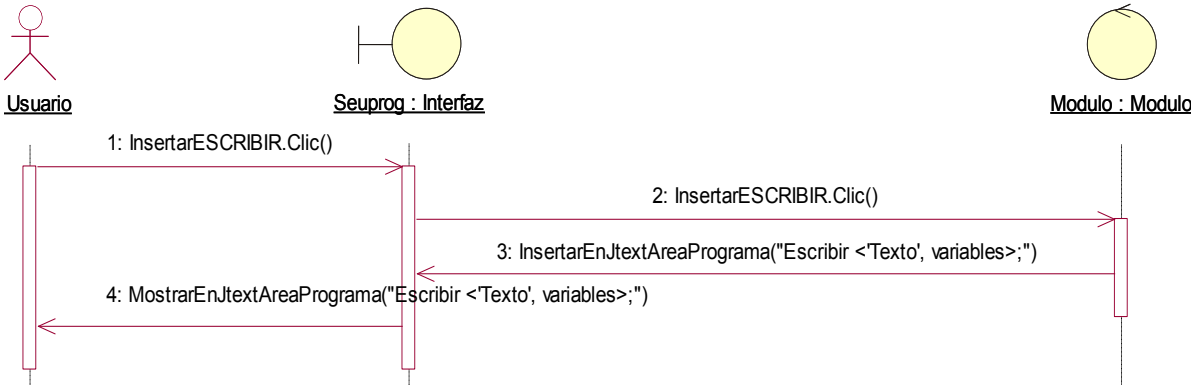


Fuente: El Autor

7.4.1.9. Diagrama de Secuencia Insertar ESCRIBIR

Curso Normal de los Eventos

Ilustración 19: Diagrama de secuencia Insertar ESCRIBIR (curso normal)

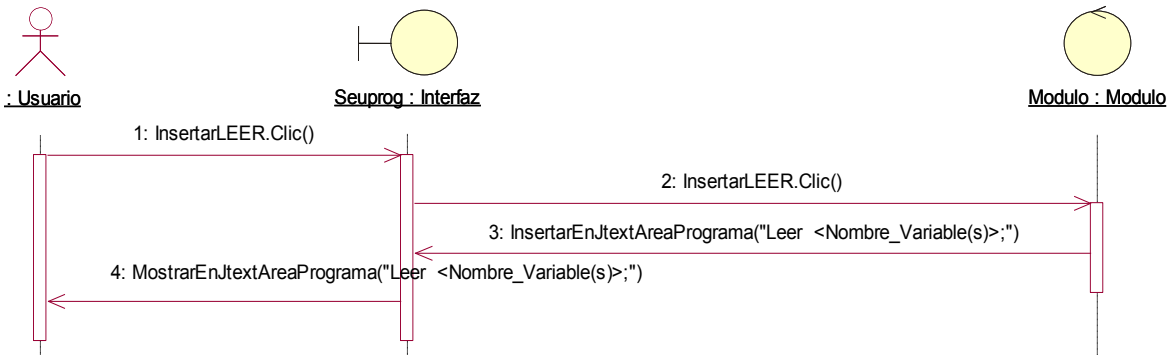


Fuente: El Autor

7.4.1.10. Diagrama de Secuencia Insertar LEER

Curso Normal de los Eventos

Ilustración 20: Diagrama de secuencia Insertar LEER (curso normal)

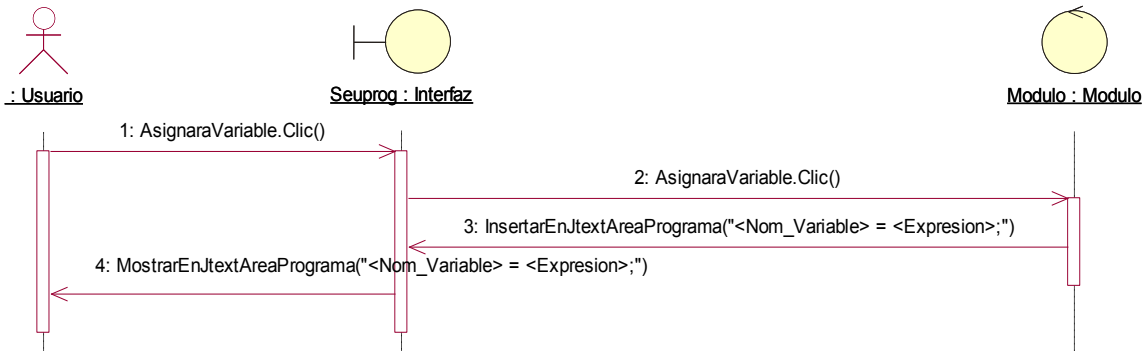


Fuente: El Autor

7.4.1.11. Diagrama de Secuencia Asígnar a Variable

Curso Normal de los Eventos

Ilustración 21: Diagrama de secuencia Asígnar a Variable (curso normal)

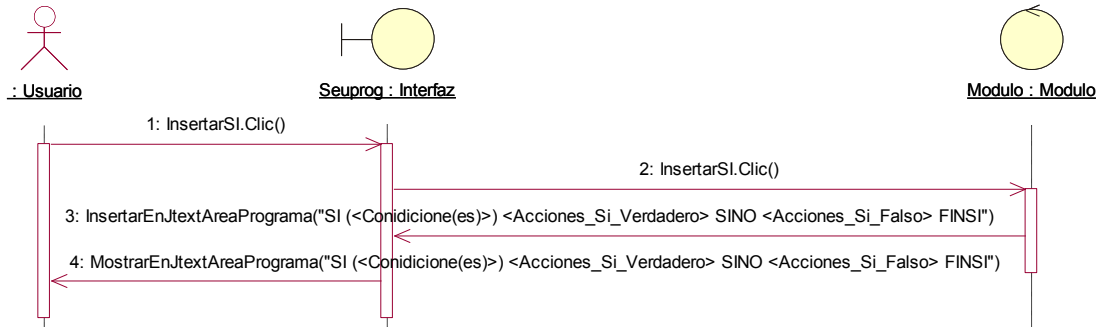


Fuente: El Autor

7.4.1.12. Diagrama de Secuencia Insertar SI

Curso Normal de los Eventos

Ilustración 22: Diagrama de secuencia Insertar SI (curso normal)

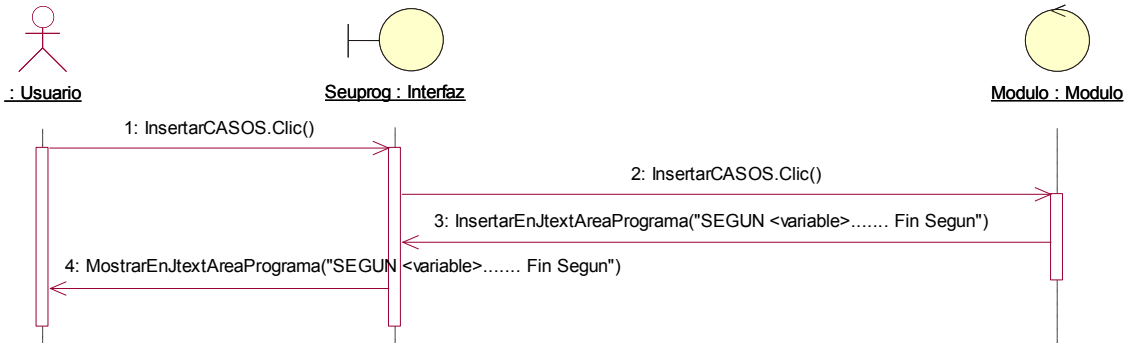


Fuente: El Autor

7.4.1.13. Diagrama de Secuencia Insertar CASOS

Curso Normal de los Eventos

Ilustración 23: Diagrama de secuencia Insertar Casos (curso normal)

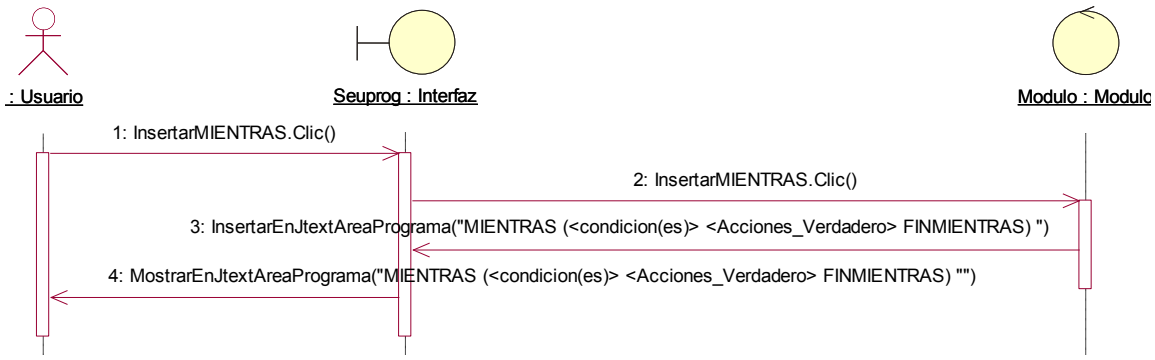


Fuente: El Autor

7.4.1.14. Diagrama de Secuencia Insertar MIENTRAS

Curso Normal de los Eventos

Ilustración 24: Diagrama de secuencia Insertar MIENTRAS (curso normal)

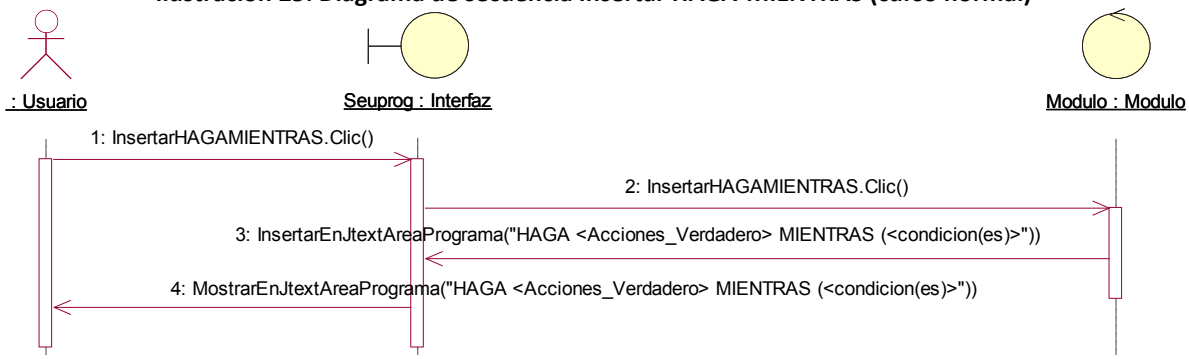


Fuente: El Autor

7.4.1.15. Diagrama de Secuencia Insertar HAGA-MIENTRAS

Curso Normal de los Eventos

Ilustración 25: Diagrama de secuencia Insertar HAGA-MIENTRAS (curso normal)

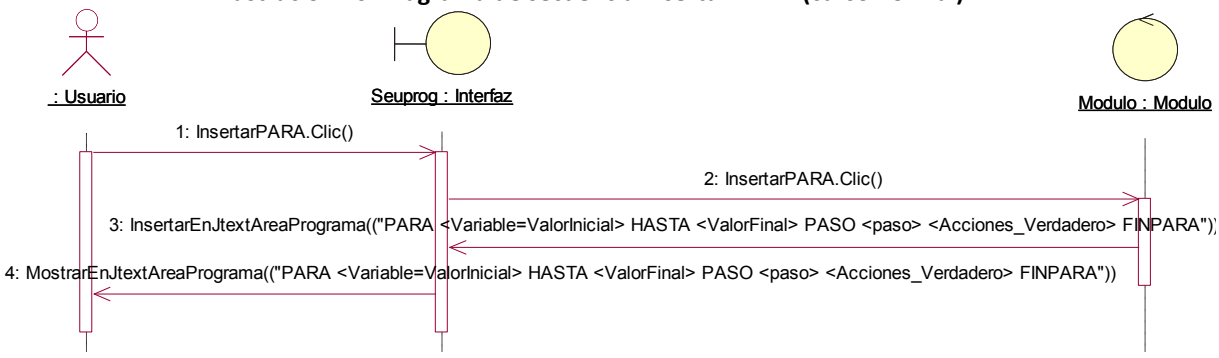


Fuente: El Autor

7.4.1.16. Diagrama de Secuencia Insertar PARA

Curso Normal de los Eventos

Ilustración 26: Diagrama de secuencia Insertar PARA (curso normal)



Fuente: El Autor

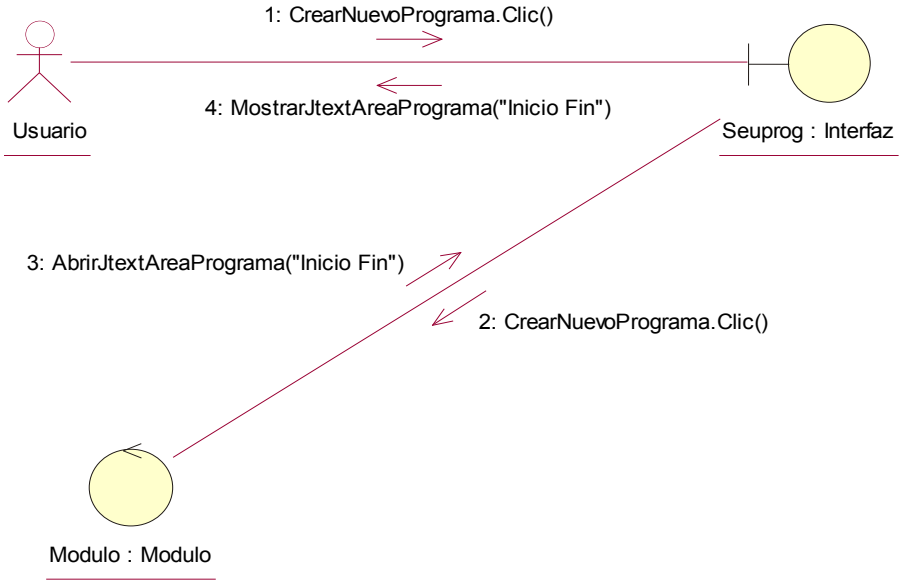
7.4.2. DIAGRAMAS DE COLABORACIÓN

Su objetivo es mostrar la estructura de clases donde ocurren los eventos.

7.4.2.1. Diagrama de Colaboración Crear Nuevo Programa

Curso Normal de los Eventos

Ilustración 27: Diagrama de colaboración Crear Nuevo Programa (curso normal)

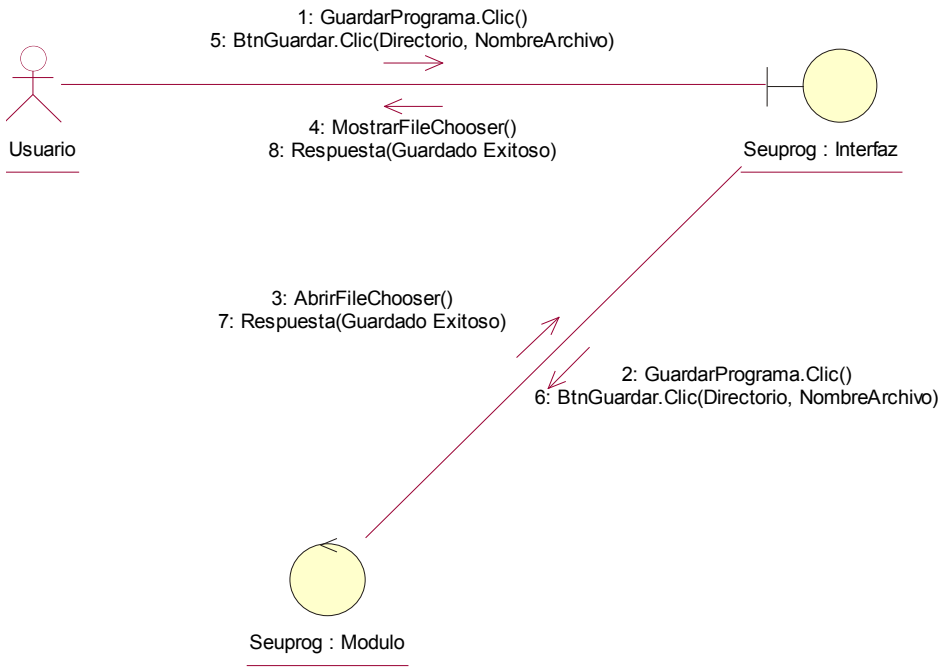


Fuente: El Autor

7.4.2.2. Diagrama de Colaboración Guardar Programa

Curso Normal de los Eventos

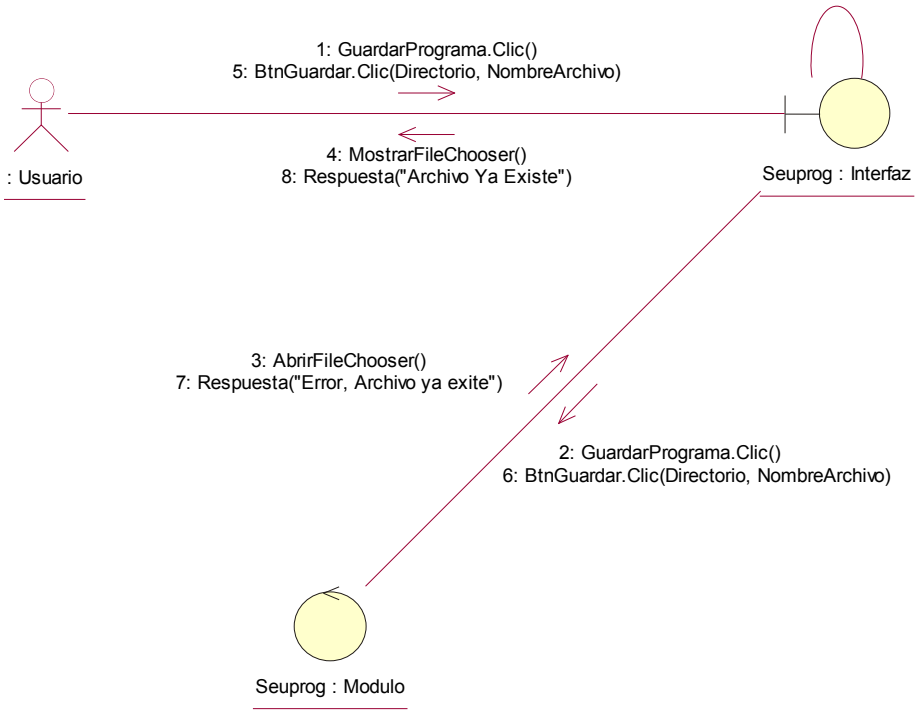
Ilustración 28: Diagrama de colaboración Guardar Programa (curso normal)



Fuente: El Autor

Curso Alternativo

Ilustración 29: Diagrama de colaboración Guardar Programa (curso alternativo)

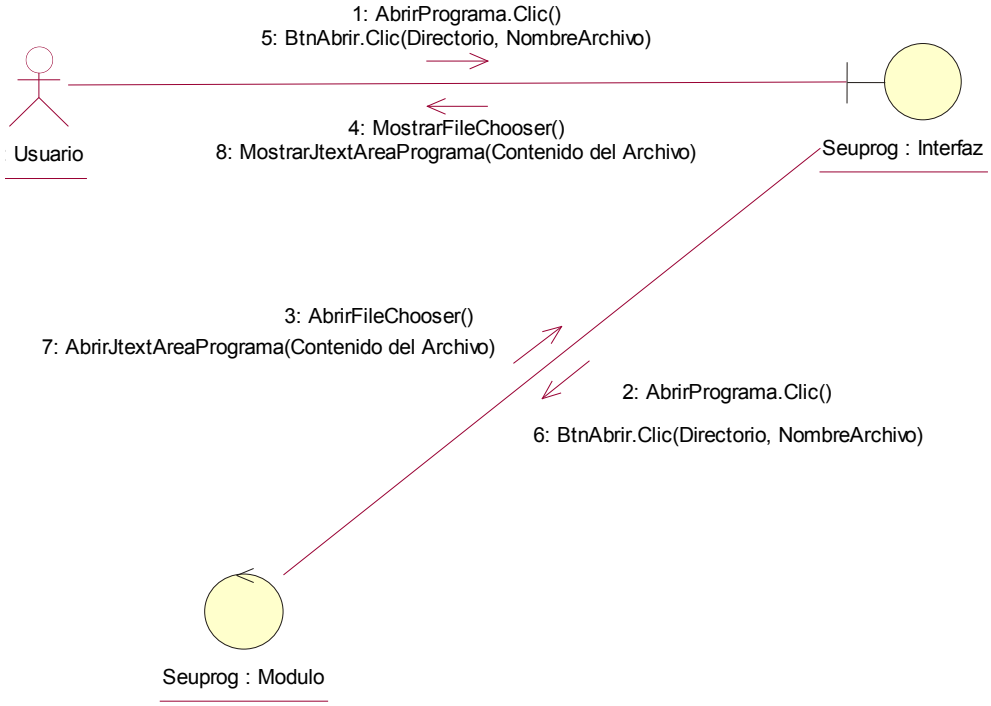


Fuente: El Autor

7.4.2.3. Diagrama de Colaboración Abrir Programa

Curso Normal de los Eventos

Ilustración 30: Diagrama de colaboración Abrir Programa (curso normal)

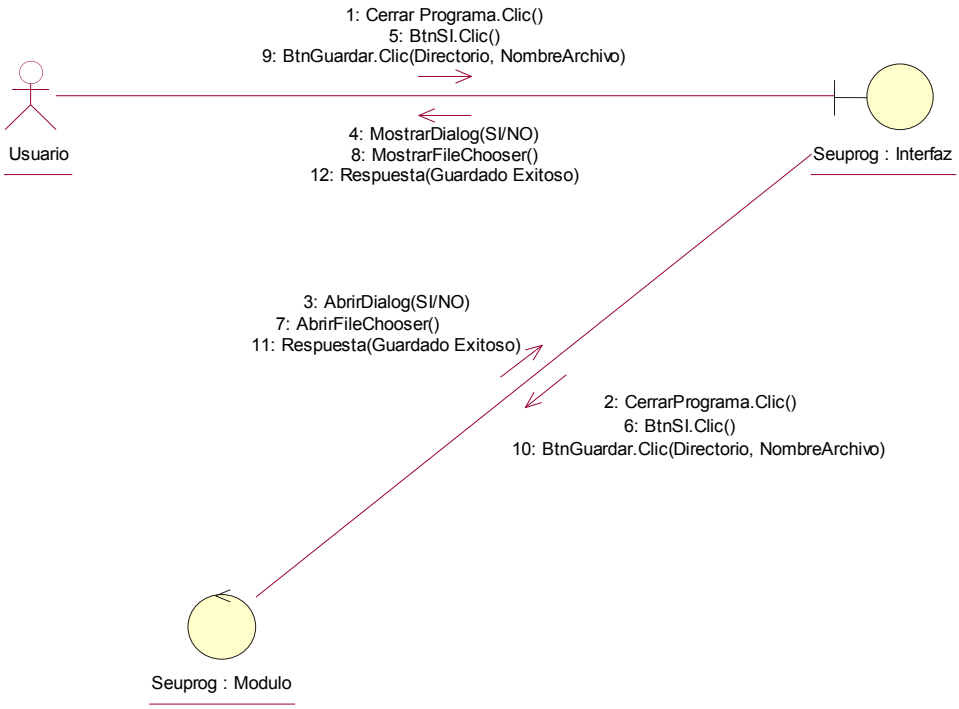


Fuente: El Autor

7.4.2.4. Diagrama de Colaboración Cerrar Programa

Curso Normal de los Eventos

Ilustración 31: Diagrama de colaboración Cerrar Programa (curso normal)

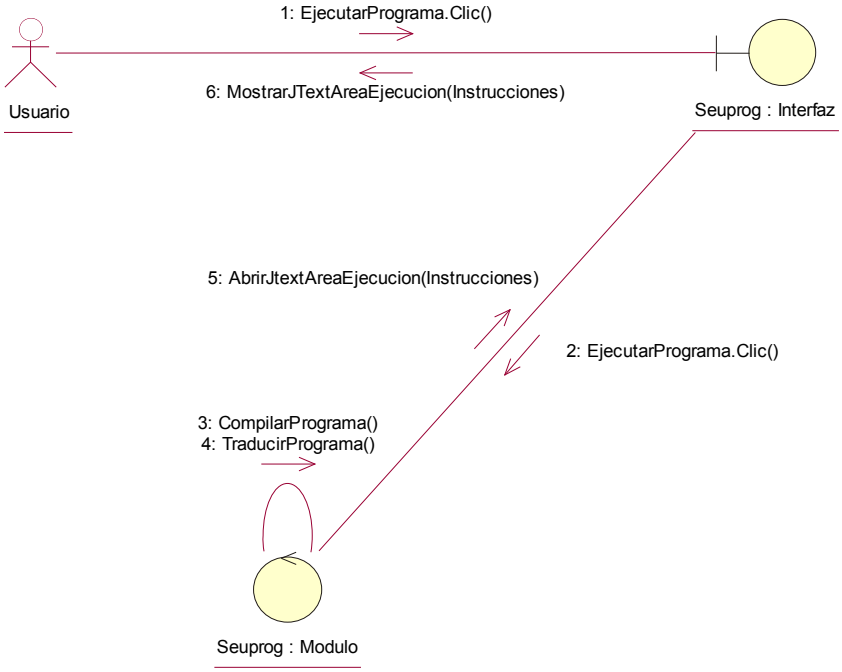


Fuente: El Autor

7.4.2.5. Diagramas de Colaboración Ejecutar Programa

Curso Normal de los eventos

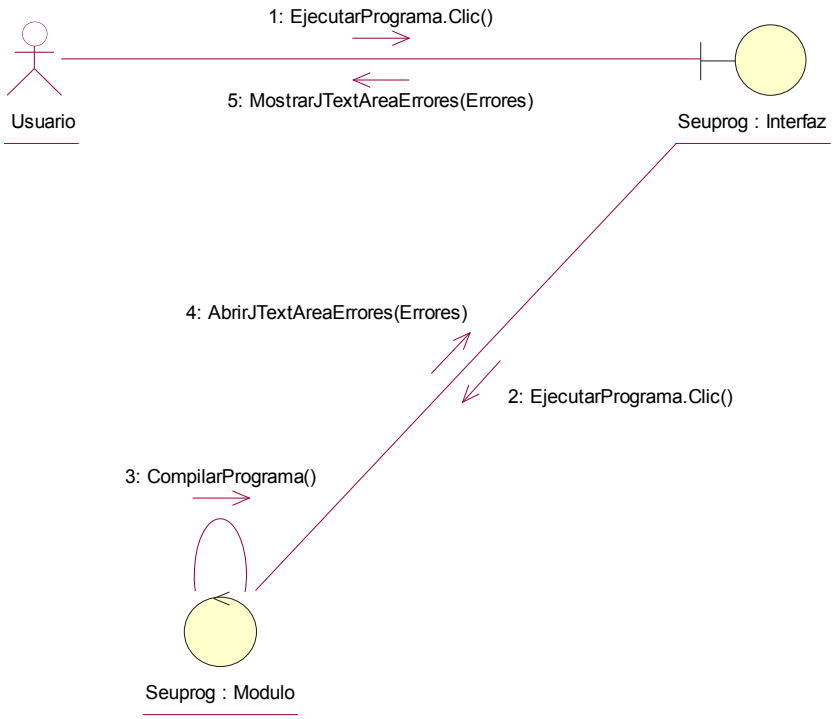
Ilustración 32: Diagrama de colaboración Ejecutar Programa (curso normal)



Fuente: El Autor

Curso Alternativo

Ilustración 33: Diagrama de colaboración Ejecutar Programa (curso alternativo)

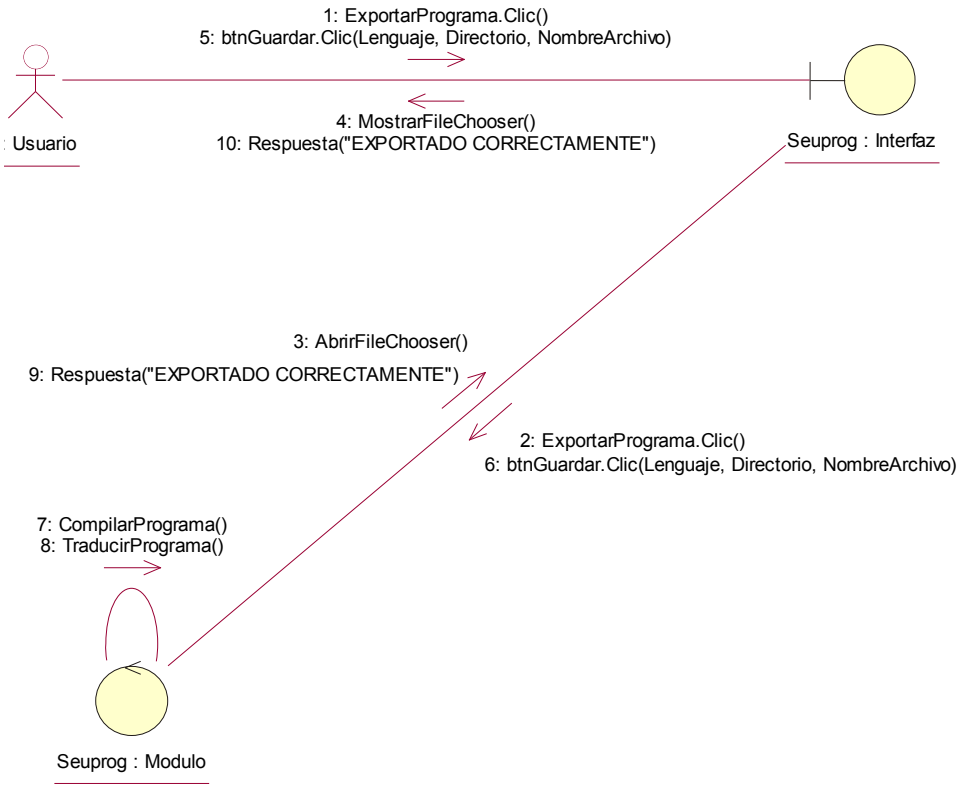


Fuente: El Autor

7.4.2.6. Diagrama de Colaboración Exportar Programa

Curso Normal de los Eventos

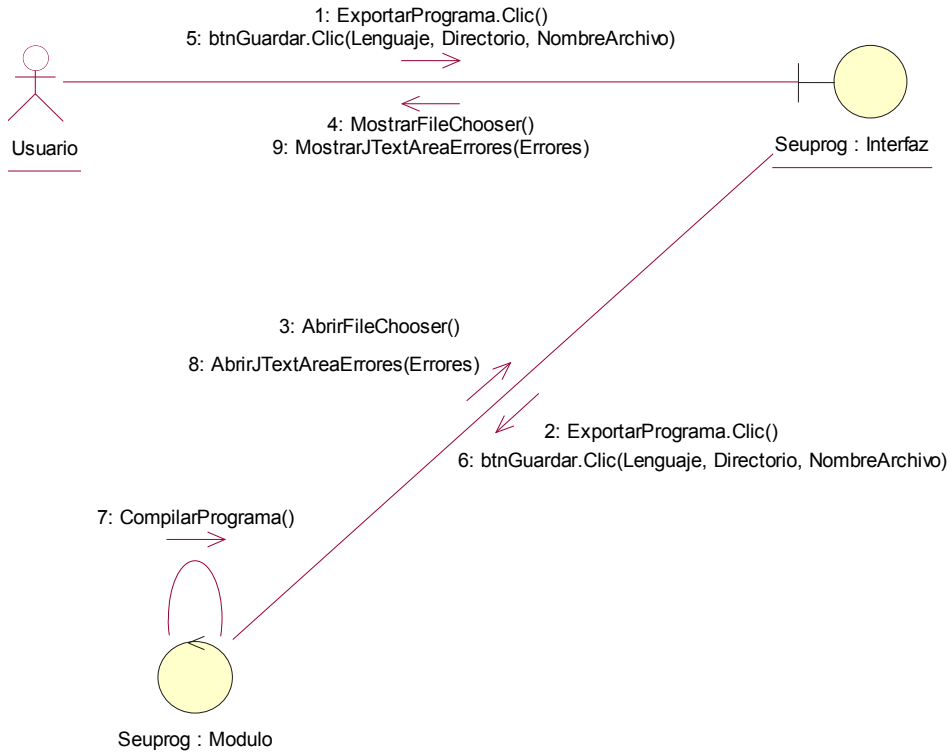
Ilustración 34: Diagrama de colaboración Exportar Programa (curso normal)



Fuente: El Autor

Curso Alternativo

Ilustración 35: Diagrama de colaboración Exportar Programa (curso alternativo)

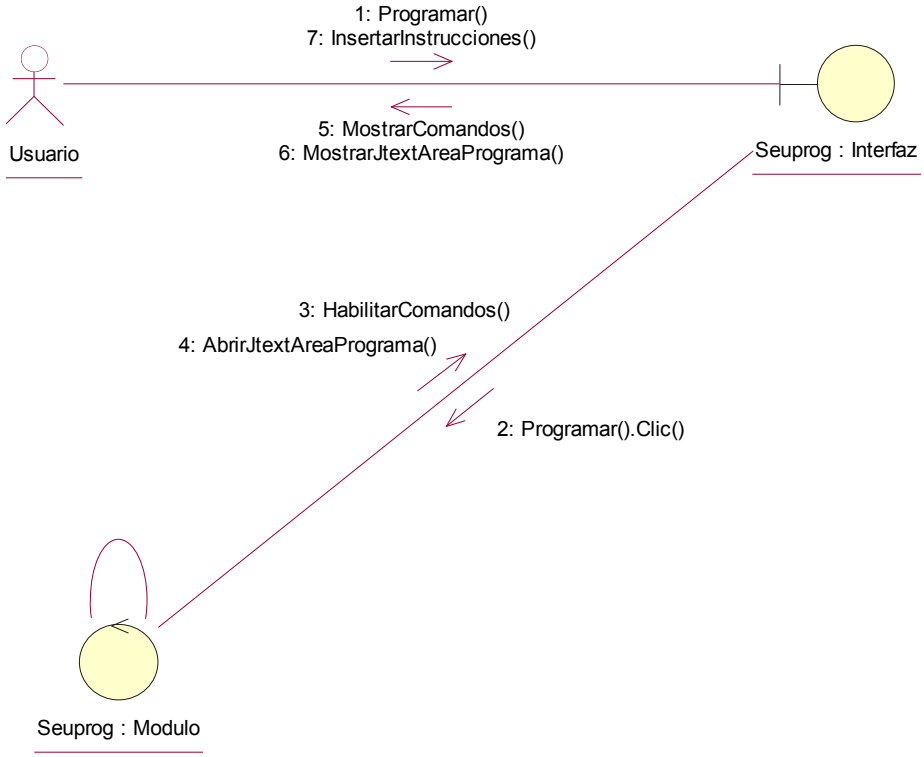


Fuente: El Autor

7.4.2.7. Diagrama de Colaboración Programar

Curso Normal de los Eventos

Ilustración 36: Diagrama de colaboración Programar (curso normal)

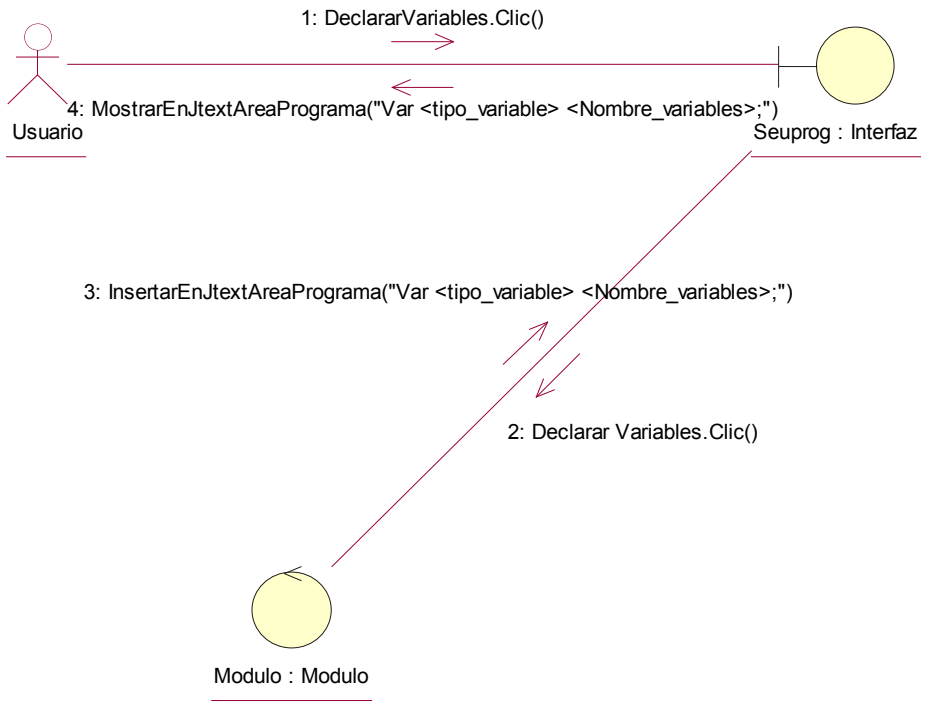


Fuente: El Autor

7.4.2.8. Diagrama de Declarar Variables

Curso Normal de los Eventos

Ilustración 37: Diagrama de colaboración Declarar Variables (curso alternativo)

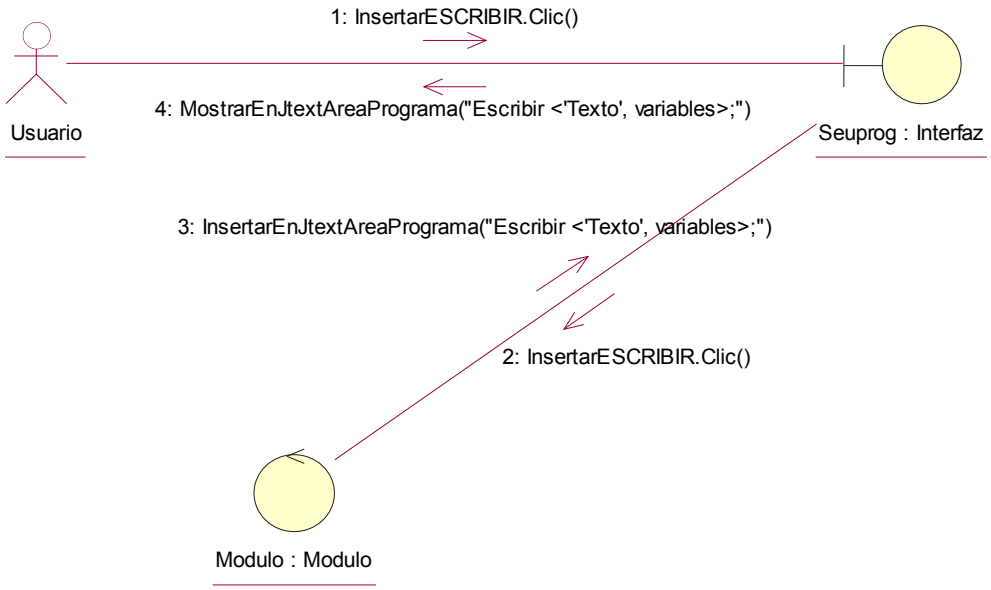


Fuente: El Autor

7.4.2.9. Diagrama de Colaboración Insertar ESCRIBIR

Curso Normal de los Eventos

Ilustración 38: Diagrama de colaboración Insertar ESCRIBIR (curso normal)

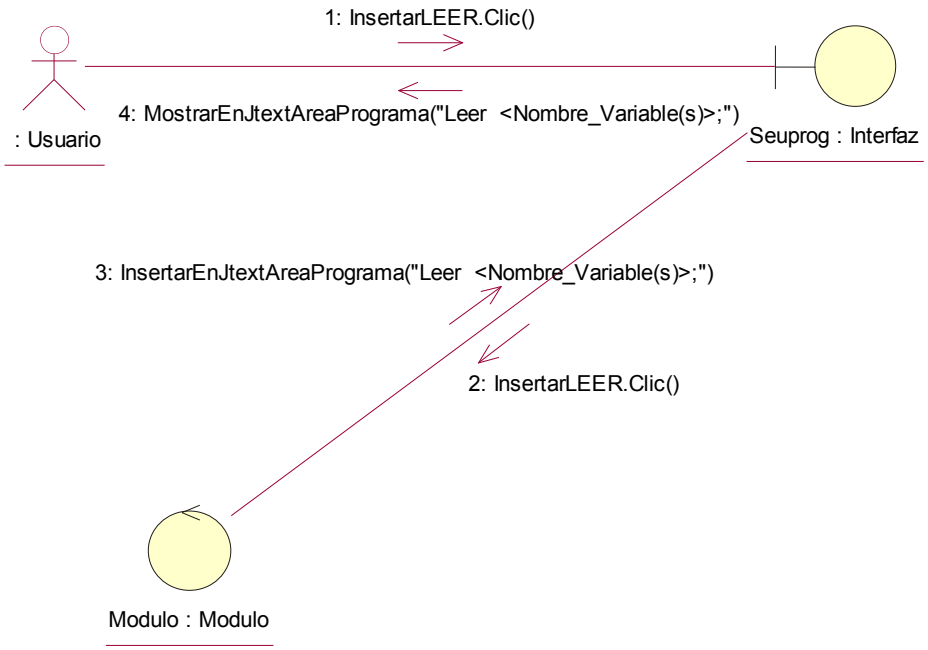


Fuente: El Autor

7.4.2.10. Diagrama de Colaboración Insertar LEER

Curso Normal de los Eventos

Ilustración 39: Diagrama de colaboración Insertar LEER (curso normal)

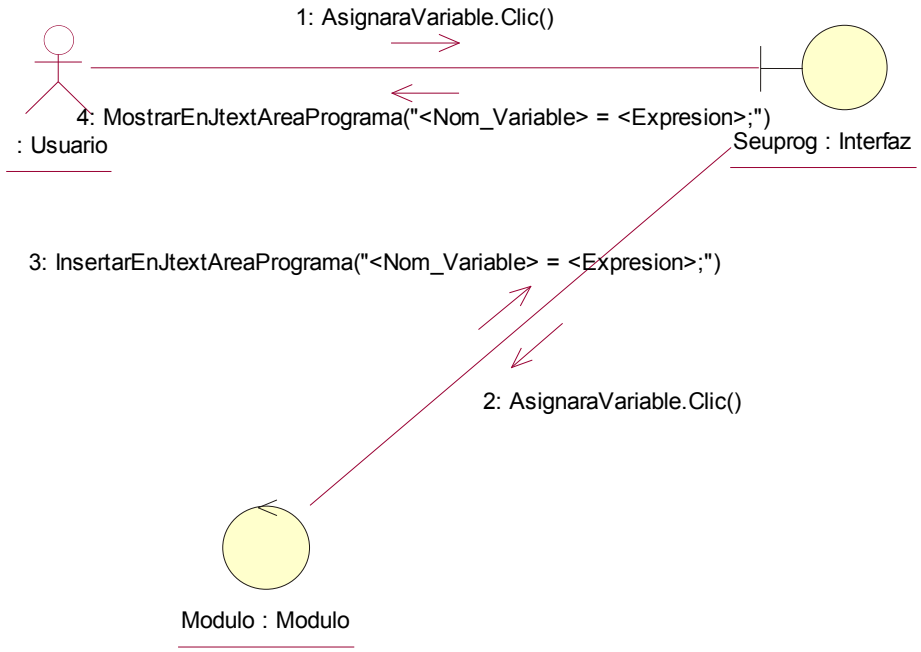


Fuente: El Autor

7.4.2.11. Diagramas de Colaboración Asígnar a Variable

Curso Normal de los Eventos

Ilustración 40: Diagrama de colaboración Asígnar a Variable (curso normal)

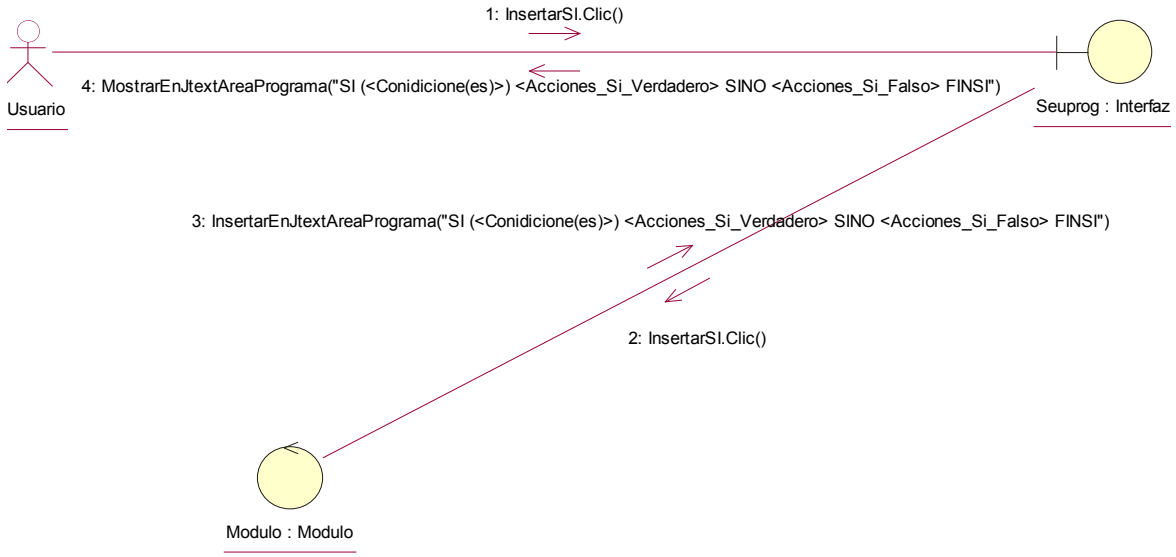


Fuente: El Autor

7.4.2.12. Diagramas de Colaboración Insertar SI

Curso Normal de los Eventos

Ilustración 41: Diagrama de colaboración Insertar SI (curso normal)

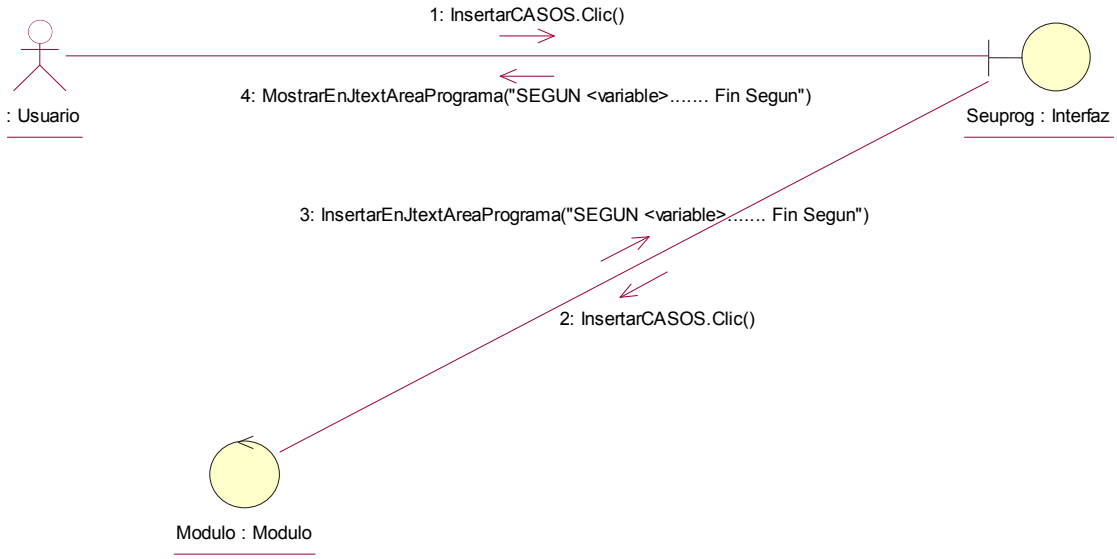


Fuente: El Autor

7.4.2.13. Diagramas de Colaboración Insertar CASOS

Curso Normal de los Eventos

Ilustración 42: Diagrama de colaboración Insertar CASOS (curso normal)

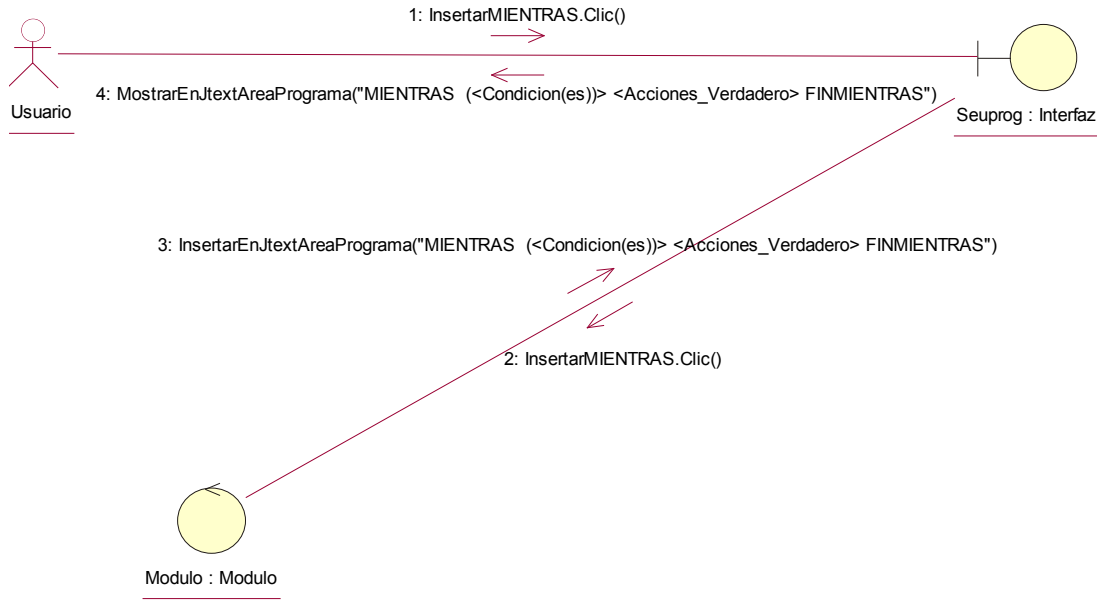


Fuente: El Autor

7.4.2.14. Diagramas de Colaboración Insertar MIENTRAS

Curso Normal de los Eventos

Ilustración 43: Diagrama de colaboración Insertar MIENTRAS (curso normal)

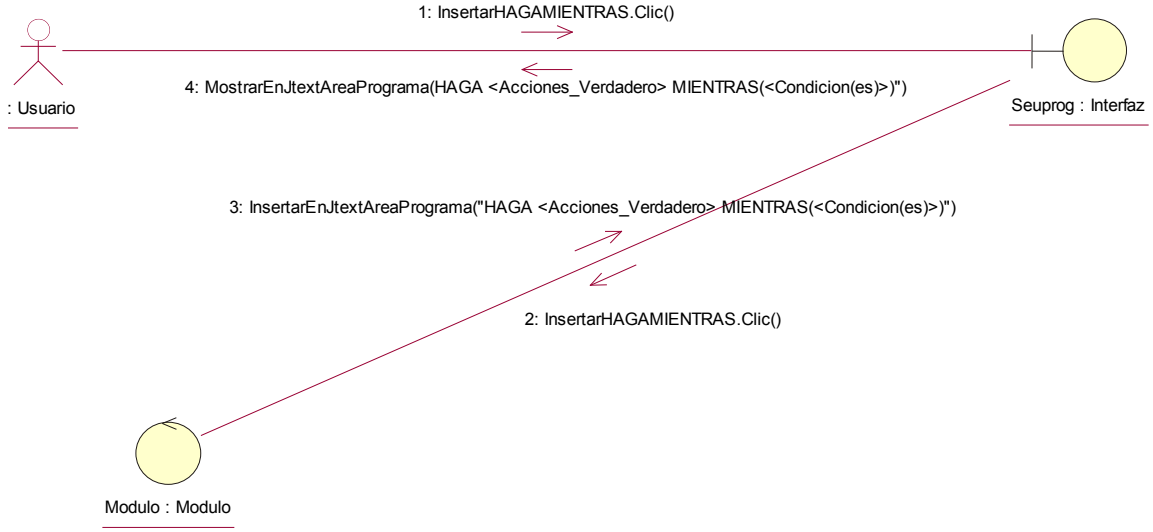


Fuente: El Autor

7.4.2.15. Diagramas de Colaboración Insertar HAGA-MIENTRAS

Curso Normal de los Eventos

Ilustración 44: Diagrama de colaboración Insertar HAGA-MIENTRAS (curso normal)

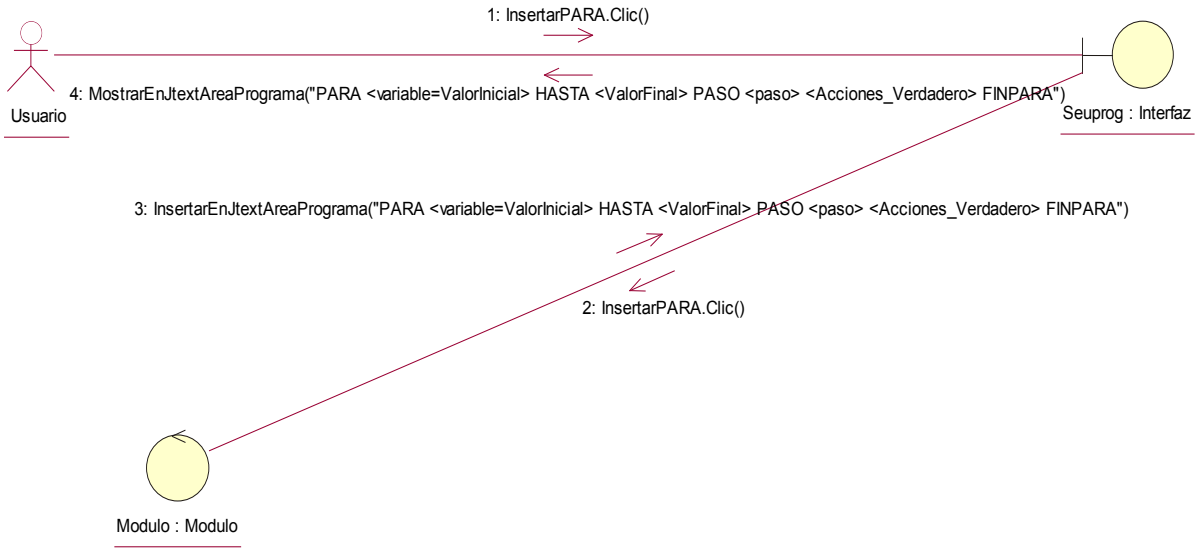


Fuente: El Autor

7.4.2.16. Diagramas de Colaboración Insertar PARA

Curso Normal de los Eventos

Ilustración 45: Diagrama de colaboración Insertar PARA (curso normal)



Fuente: El Autor

7.4.3. DIAGRAMAS DE ACTIVIDADES

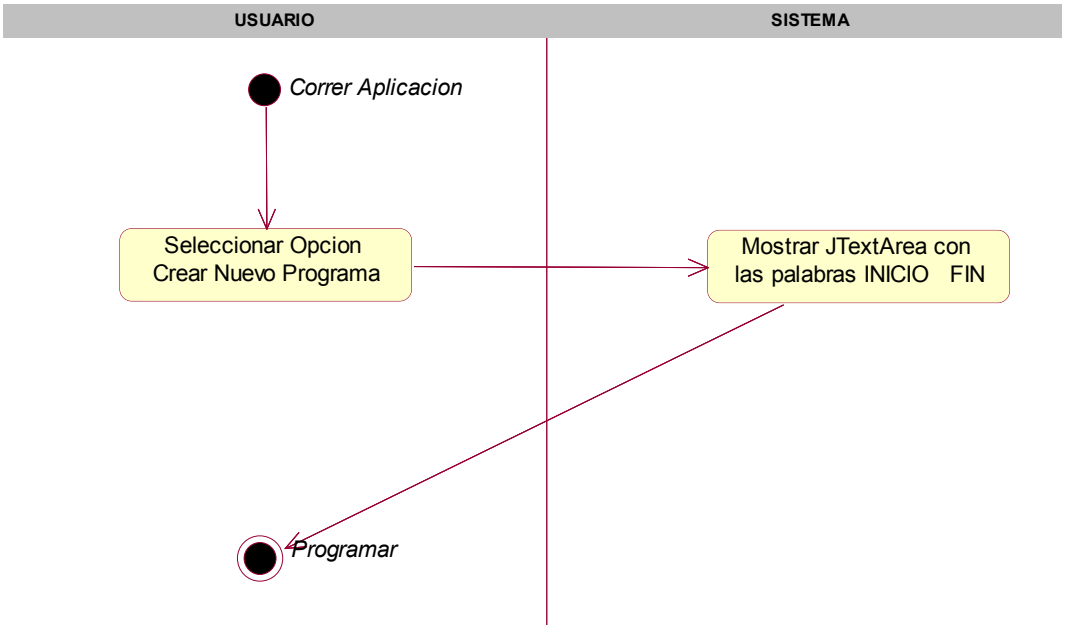
Se utilizan para construir el modelo funcional. Es un caso especial del diagrama de estados, que muestra estados de acción y estados de actividad.

La diferencia entre acciones y actividades es su duración, las actividades son largas y las acciones son instantáneas.

En este diagrama no se tiene en cuenta la diferencia entre acciones y actividades.

7.4.3.1. Diagrama de Actividades Crear Nuevo Programa

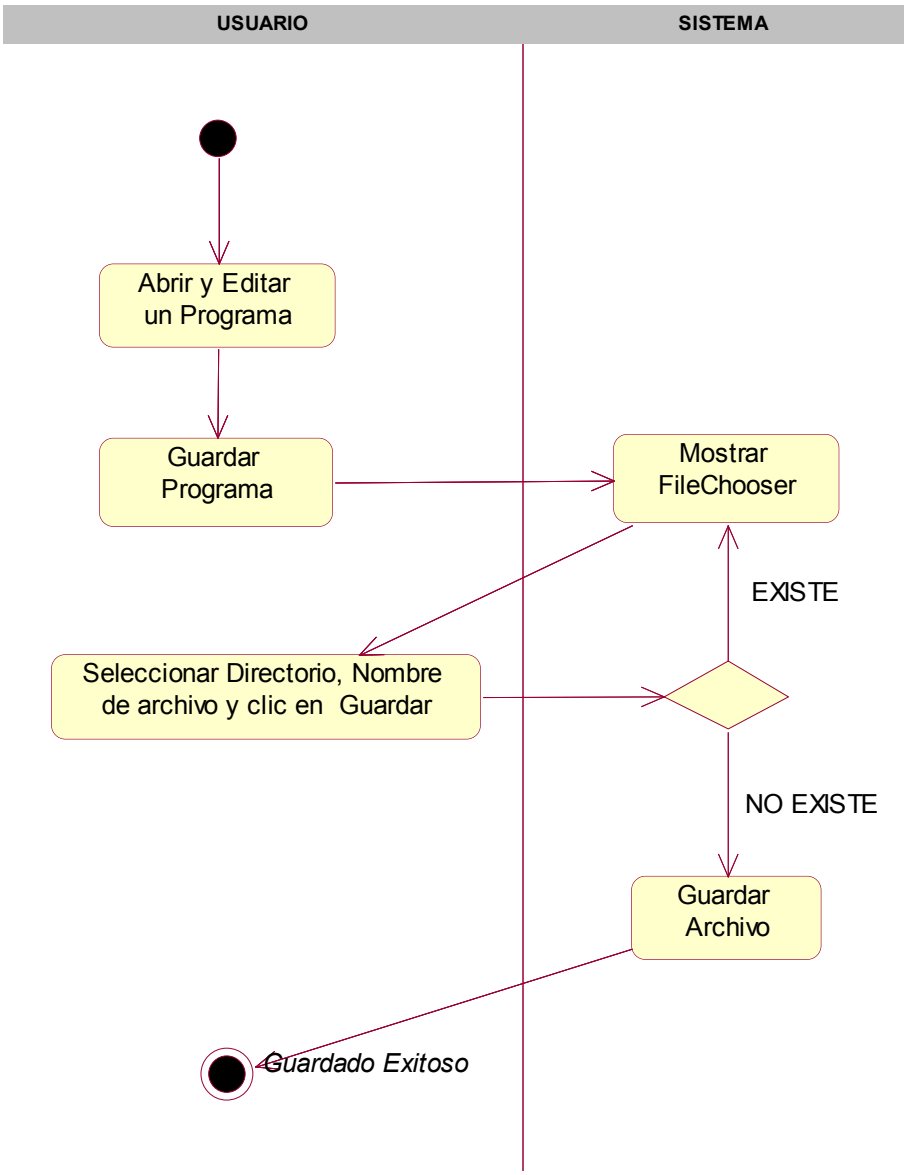
Ilustración 46: Diagrama de actividades Crear Nuevo Programa



Fuente: El Autor

7.4.3.2. Diagrama de Actividades Guardar Programa

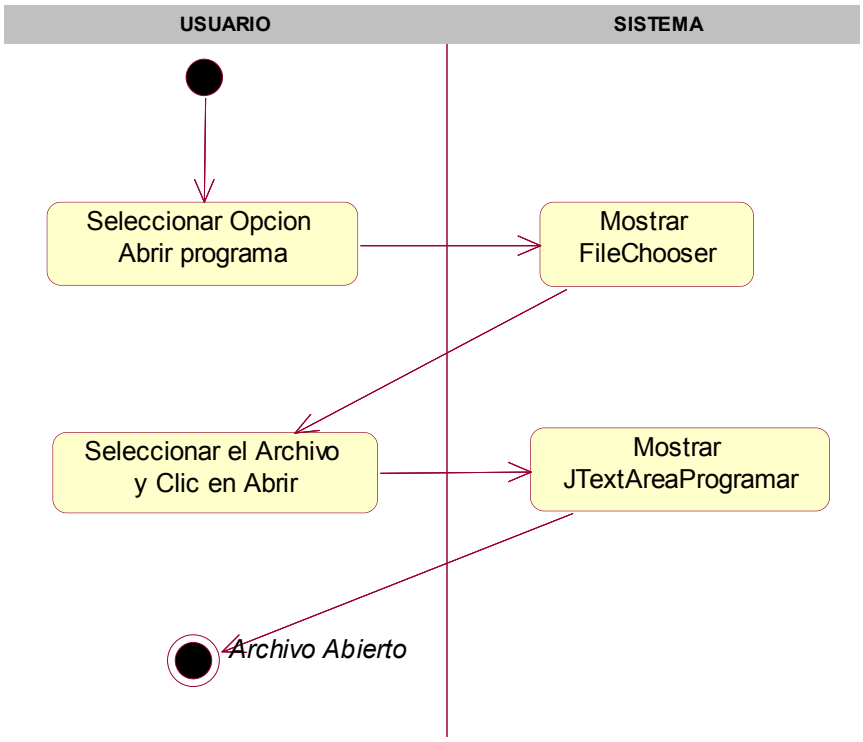
Ilustración 47: Diagrama de actividades Guardar Programa



Fuente: El Autor

7.4.3.3. Diagrama de Actividades Abrir Programa

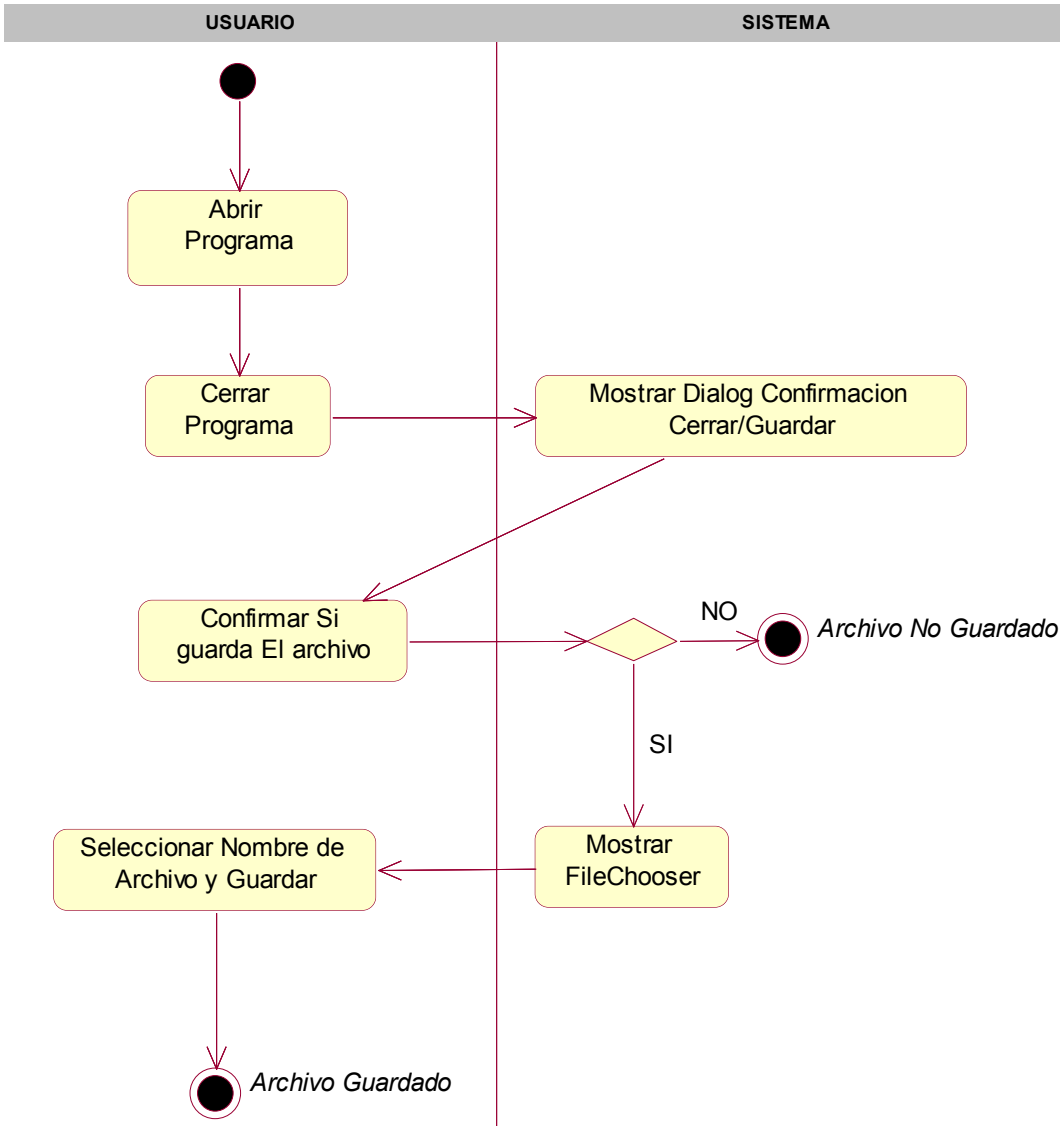
Ilustración 48: Diagrama de actividades Abrir Programa



Fuente: El Autor

7.4.3.4. Diagrama de Actividades Cerrar Programa

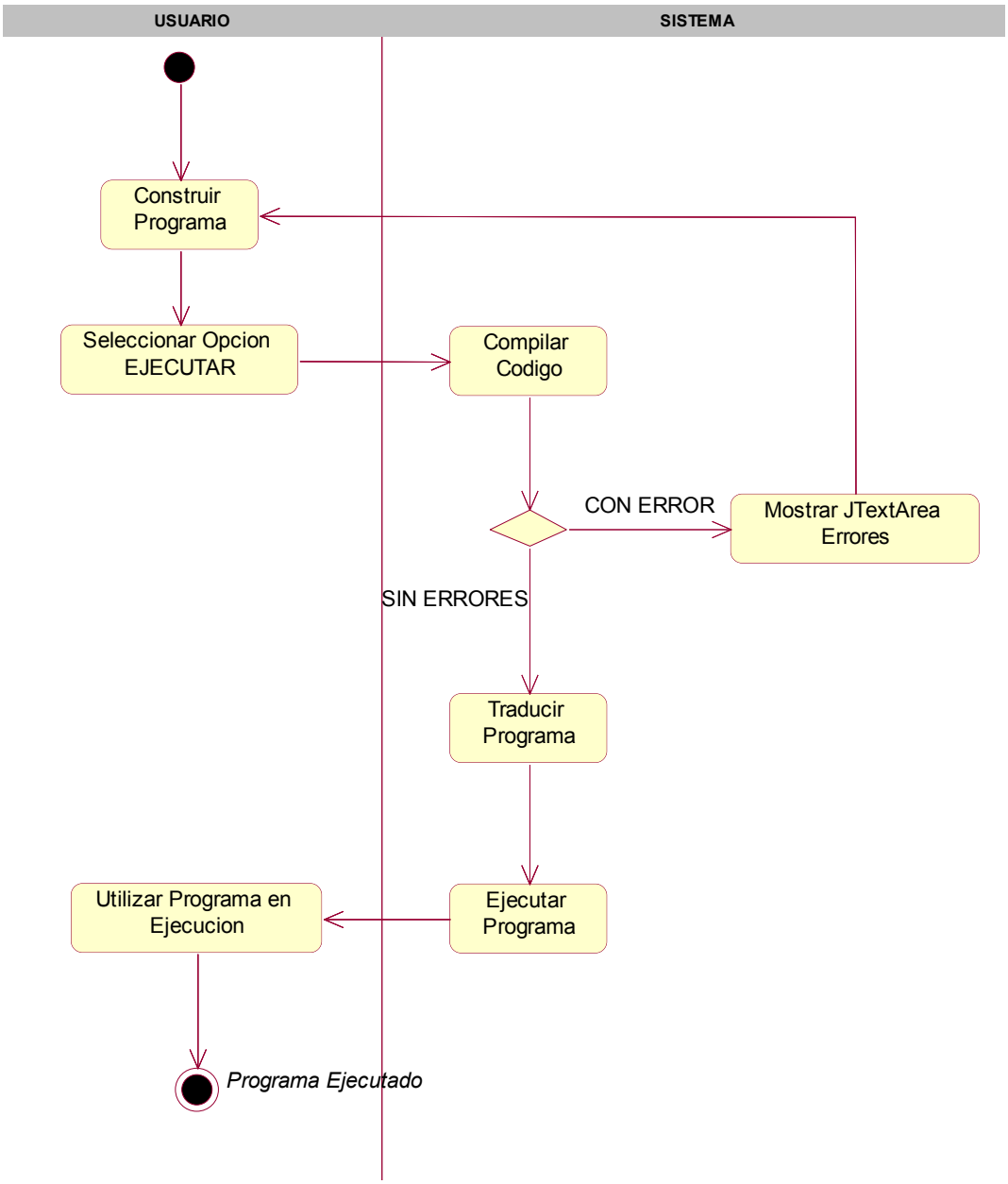
Ilustración 49: Diagrama de actividades Cerrar Programa



Fuente: El Autor

7.4.3.5. Diagrama de Actividades Ejecutar Programa

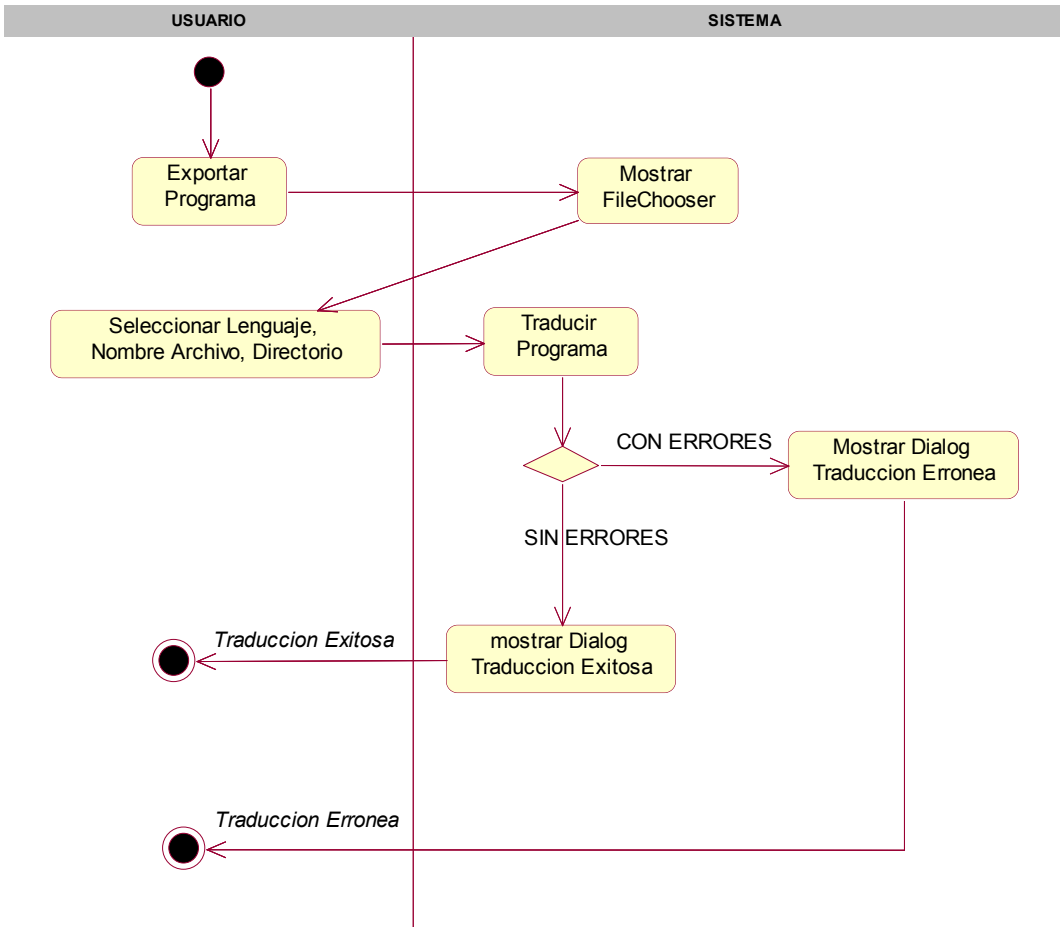
Ilustración 50: Diagrama de actividades Ejecutar Programa



Fuente: El Autor

7.4.3.6. Diagrama de Actividades Exportar Programa

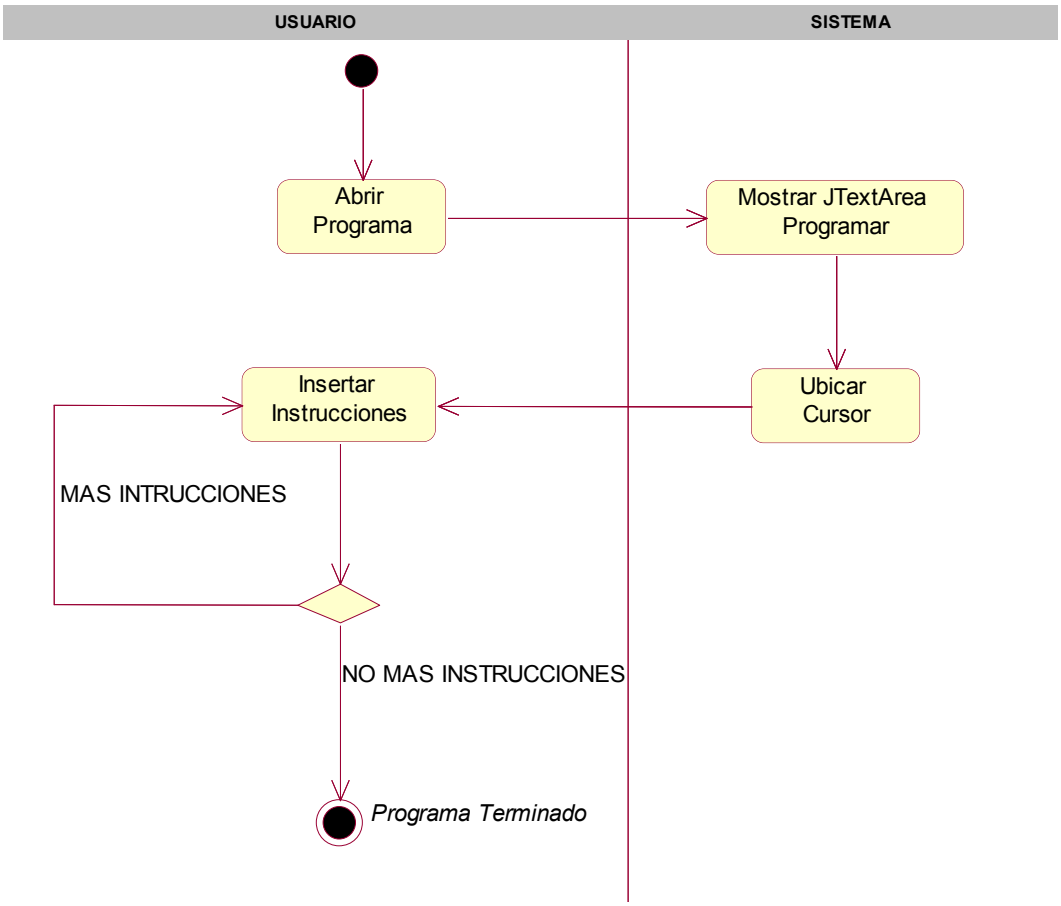
Ilustración 51: Diagrama de actividades exportar Programa



Fuente: El Autor

7.4.3.7. Diagrama de Actividades Programar

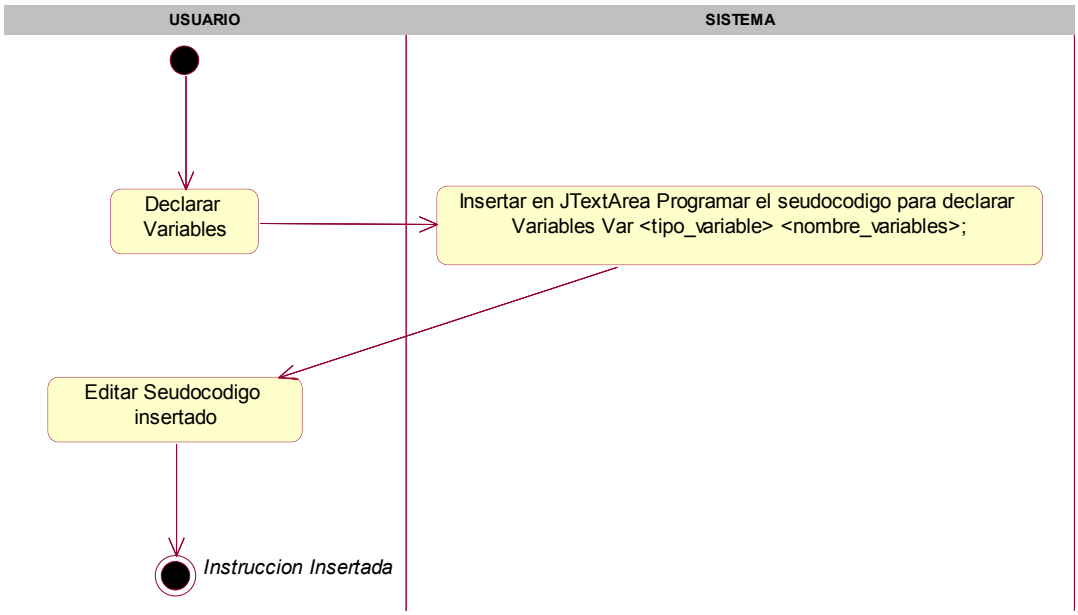
Ilustración 52: Diagrama de actividades Programar



Fuente: El Autor

7.4.3.8. Diagrama de Actividades Declarar Variables

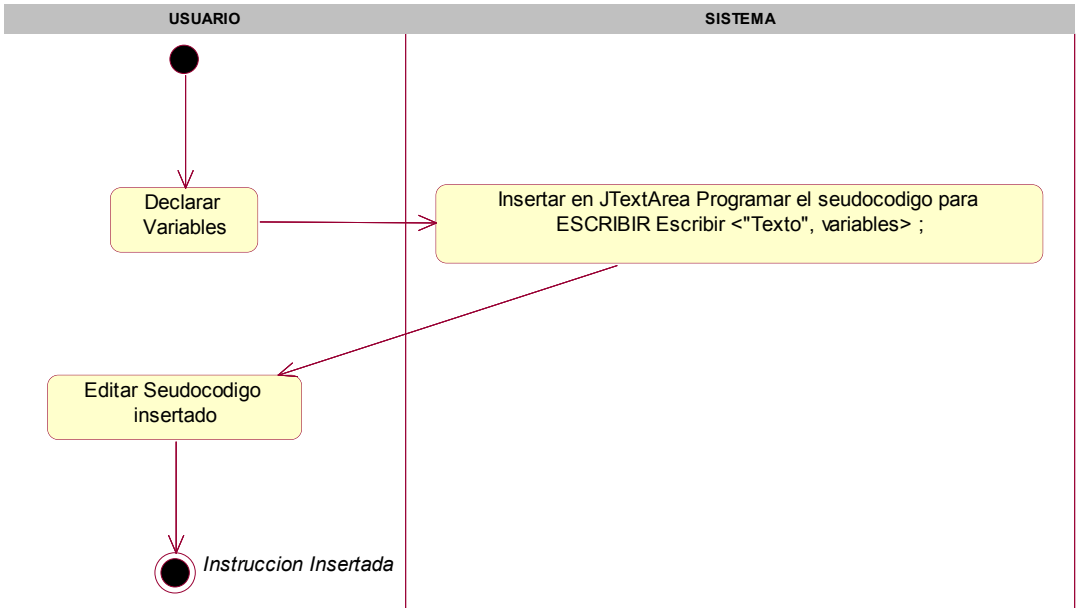
Ilustración 53: Diagrama de actividades Declarar Variables



Fuente: El Autor

7.4.3.9. Diagrama de Actividades Insertar ESCRIBIR

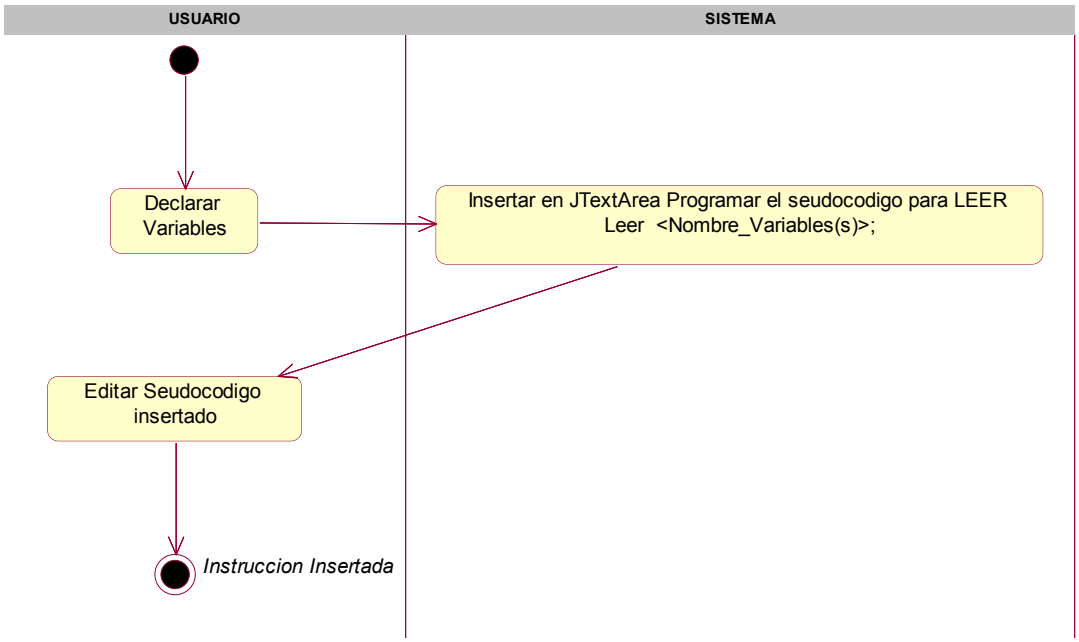
Ilustración 54: Diagrama de actividades Insertar ESCRIBIR



Fuente: El Autor

7.4.3.10. Diagrama de Actividades Insertar LEER

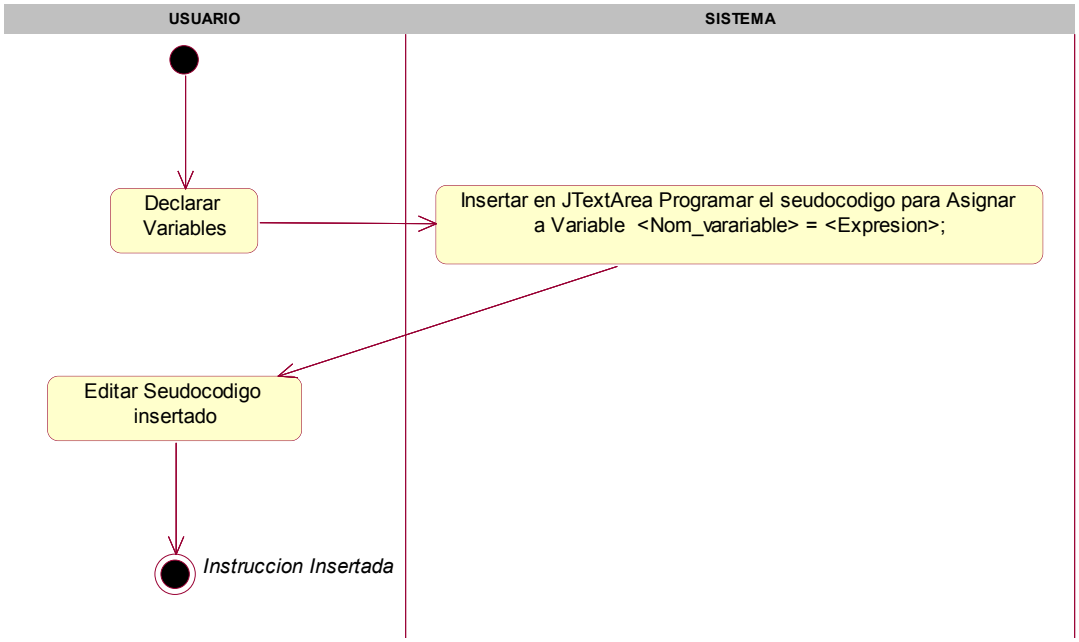
Ilustración 55: Diagrama de actividades Insertar LEER



Fuente: El Autor

7.4.3.11. Diagrama de Actividades Asígnar a Variable

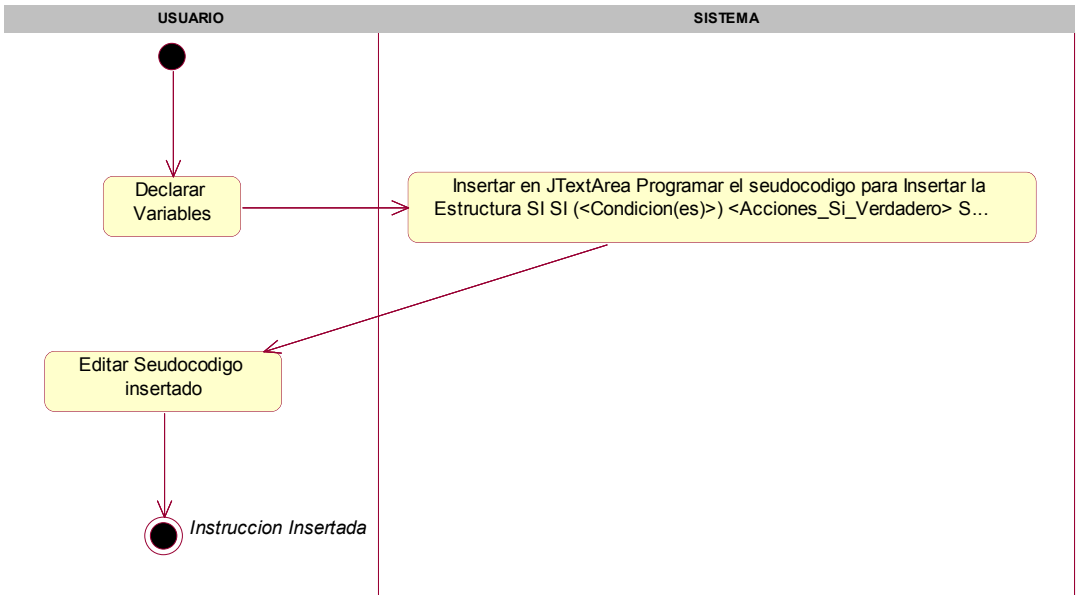
Ilustración 56: Diagrama de actividades Asígnar a Variable



Fuente: El Autor

7.4.3.12. Diagrama de Actividades Insertar SI

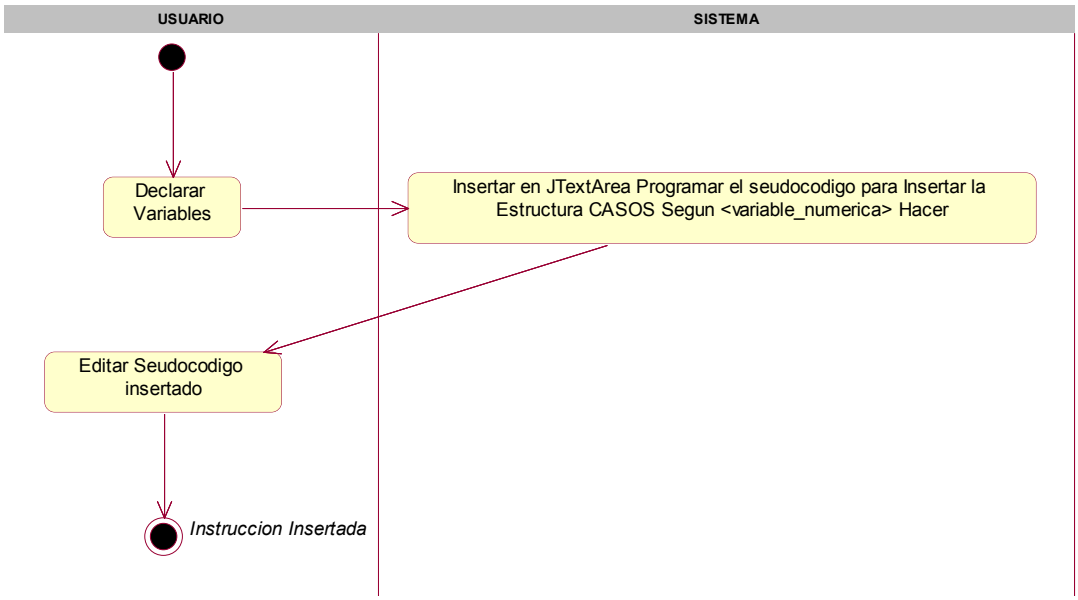
Ilustración 57: Diagrama de actividades Insertar SI



Fuente: El Autor

7.4.3.13. Diagrama de Actividades Insertar CASOS

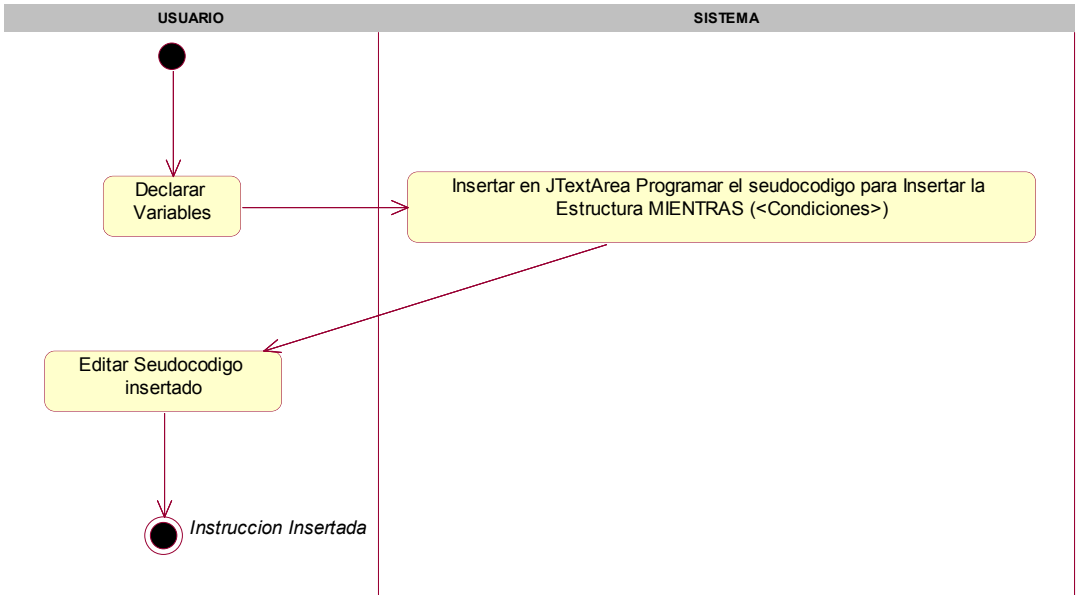
Ilustración 58: Diagrama de actividades Insertar CASOS



Fuente: El Autor

7.4.3.14. Diagrama de Actividades Insertar MIENTRAS

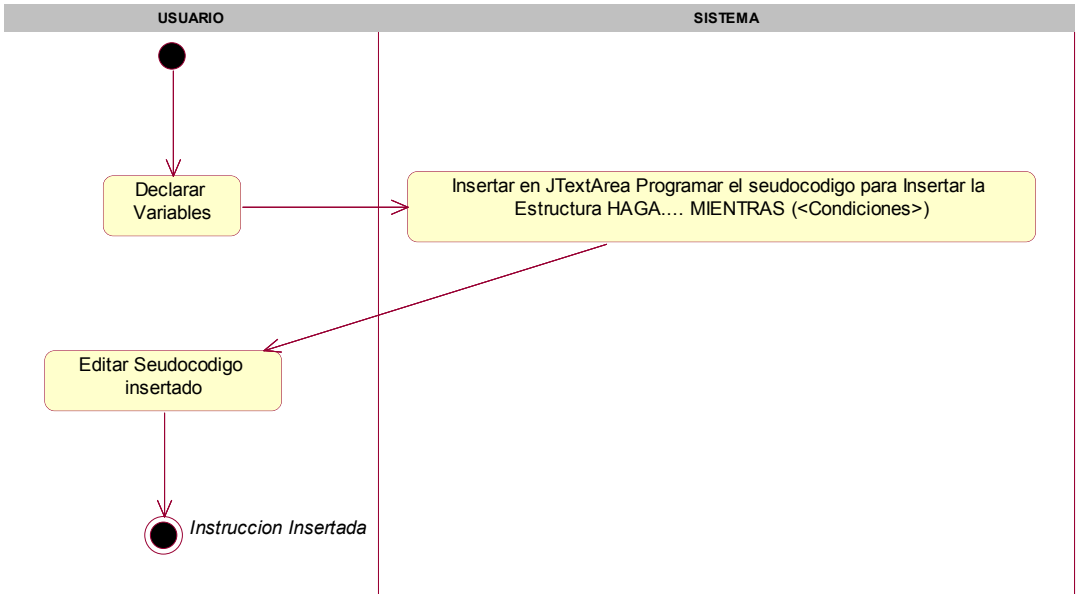
Ilustración 59: Diagrama de actividades Insertar MIENTRAS



Fuente: El Autor

7.4.3.15. Diagrama de Actividades Insertar HAGA-MIENTRAS

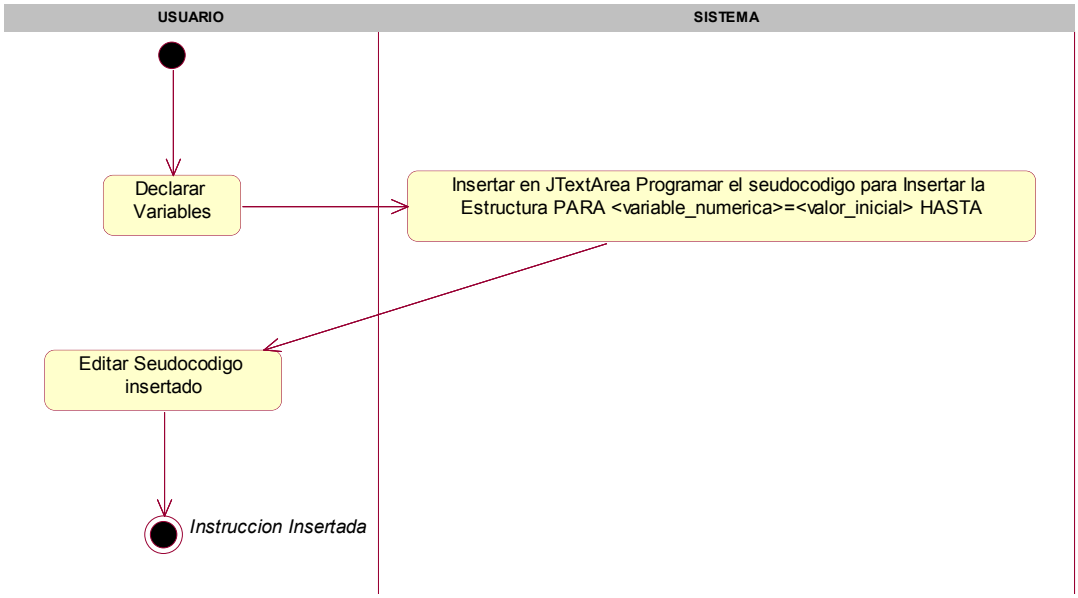
Ilustración 60: Diagrama de actividades Insertar HAGA-MIENTRAS



Fuente: El Autor

7.4.3.16. Diagrama de Actividades Insertar PARA

Ilustración 61: Diagrama de actividades Insertar PARA



Fuente: El Autor

7.5 MODELO DE ESTADOS

Consisten en hacer un modelo de estados para cada clase.

En este modelo de estados solo se muestra el modelo de estados de la clase usuario y el modelo de estados de la clase evaluación, ya que las demás clases permanecen en un solo estado.

Estado

El objeto es un sistema neutral, sus atributos no están divididos en entradas y salidas. El estado de un objeto se refiere a la combinación de los valores de todos sus atributos.

Transición

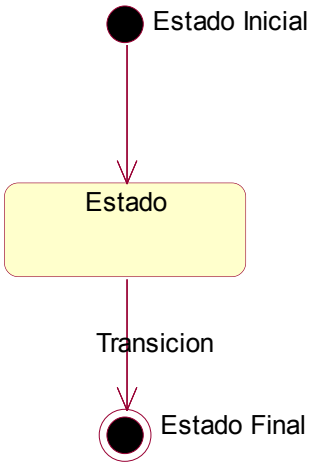
Sucede cuando un sistema cambia de un estado a otro, se consideran instancias.

Estado Inicial: Generalmente se llama "No Existe". Es el primer estado del objeto, cuando no ha sido creado. El estado inicial origina la primera transición. Una clase solo puede tener un estado inicial.

Estado final: Indica la desaparición del objeto o cuando el objeto llega a un

Estado del cual no va cambiar a otro estado.

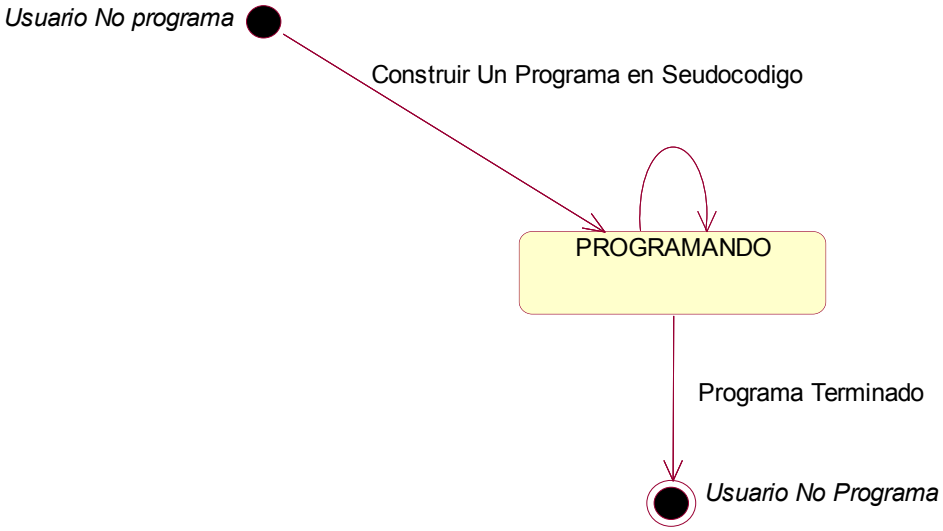
Ilustración 62: Representación del diagrama de estados



Fuente: El Autor

7.5.1 MODELO DE ESTADOS USUARIO

Ilustración 63: Diagrama de estados usuario



Fuente: El Autor

7.6 REQUERIMIENTOS NO FUNCIONALES

7.6.1. CONFIABILIDAD

Es necesario que los programas que se desarrollen en la herramienta queden funcionando correctamente.

7.6.2. USABILIDAD

Se requiere que el sistema este a disposición de los Usuarios en cualquier momento.

7.6.3. RELATIVOS A LA INTERFACE

Es necesaria una interface amigable para el usuario desde el punto de vista estético y funcional.

7.6.4. CAPACIDAD DE PROCESO

La aplicación debe permitir crear y ejecutar programas con gran cantidad de líneas de código fuente, siempre y cuando se ejecute con las instrucciones básicas permitidas.

7.6.5. FLEXIBILIDAD

El programar debe permitir la interacción con otros lenguajes de programación como JAVA.

8. DISEÑO DEL SISTEMA

8.1. DISEÑO ARQUITECTÓNICO

8.1.1. SELECCIÓN DE LA ARQUITECTURA

8.1.1.1. Sistema Monousuario

Se necesita una única persona para que pueda trabajar el sistema.

El Sistema tiene una interfaz gráfica de Usuario (GUI) q se encarga de la interacción directa con el usuario que desea realizar programas en seudocódigo.

8.1.1.2. Sistema operativo requerido

- Windows XP o Windows 7

  Se usa el sistema operativo Windows XP o Windows 7 Inicialmente, debido a que es el más popular y es el que generalmente usan la mayoría de usuarios promedio. Su soporte técnico es aceptable, ya que continuamente Microsoft lanza nuevas actualizaciones y correcciones del sistema, lo que facilita que el usuario no tenga problemas a la hora de usar este servicio. Además el sistema tiene muchas utilidades importantes que de una manera u otra ayudan a desarrollar el software en cuestión. Está contemplado realizar en un futuro la migración a Sistemas Operativos libres como Ubuntu o Debían q utilizan el núcleo Linux y la mayor parte de las herramientas básicas vienen del Proyecto GNU.

8.1.1.3. Herramientas para el desarrollo del Software

- **Programación**

  NETBEANS

  Para el desarrollo de la aplicación y su interfaz grafica se utiliza la plataforma grafica NETBEANS que facilita un entorno de desarrollo integrado (IDE) para programar en lenguajes de programación C, C++, JAVA y Python. Además ofrece

un gran número de características avanzadas de programación, son software libre y de código abierto, disponible bajo la Licencia Pública General de GNU.

LENGUAJE DE PROGRAMACION JAVA

Como lenguaje de programación se utiliza JAVA por ser un lenguaje multiplataforma que permite desarrollar programas y ejecutarlos sin problemas en diferentes sistemas operativos como Windows, Linux, Mac , Unix y permite que se ejecuten en gran variedad de dispositivos como computadoras, teléfonos móviles, consolas de juegos, electrodomésticos y algunos microcontroladores. Además no es necesario comprar ningún programa o entorno de desarrollo puesto que Java no es código libre pero permite que cualquiera utilice su **JRE** (maquina virtual) para que desarrolle programas.

- **Diseño y Edición de Imágenes**

DIA

Se usa esta herramienta porque es sencilla de utilizar y permite realizar diseños gráficos de una manera simple y eficaz. Además maneja diferentes formatos de imagen lo que permitirá poder adecuar nuestros diseños a las capacidades brindadas por el lenguaje de desarrollo y porque no las brindadas también por el sistema operativo. Además es gratuita y de código abierto para uso general en diagramación y utiliza un control interfaz de único documento (CSDI) similar a GIMP.

8.1.1.4. Requerimientos Mínimos para Instalar la Aplicación

- Sistema operativo Windows XP o Windows 7
- Instalación del JDK.
- Memoria RAM 521 MB

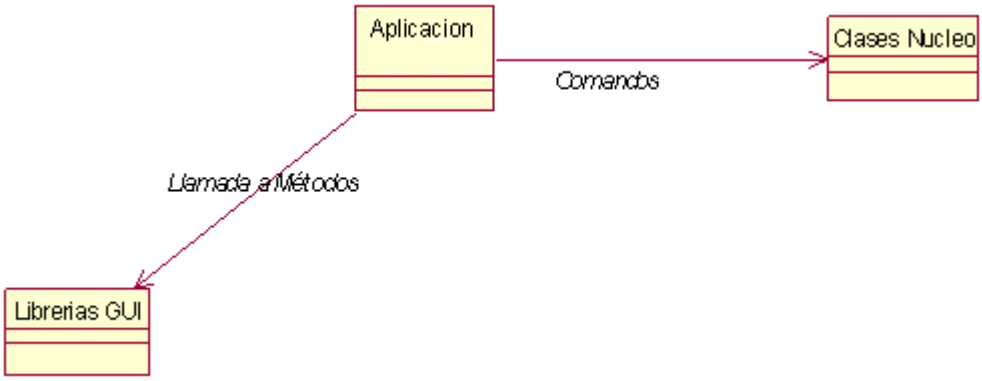
- 5 GB de Espacio en Disco Duro

8.1.2. DIAGRAMAS DE LOS SUBSISTEMAS

Agrupa los elementos del modelo de diseño con el objeto de obtener una visión más clara y contiene los siguientes elementos: paquetes: agrupación de elementos, casos de uso, clase o componentes y dependencias entre paquetes.

8.1.2.1. Diagrama Dinámico de Subsistema

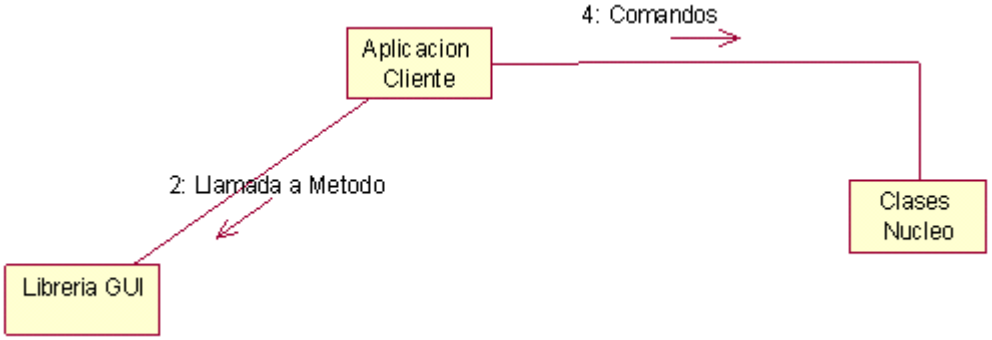
Ilustración 64: Diagrama dinámico de Subsistemas



Fuente: El Autor

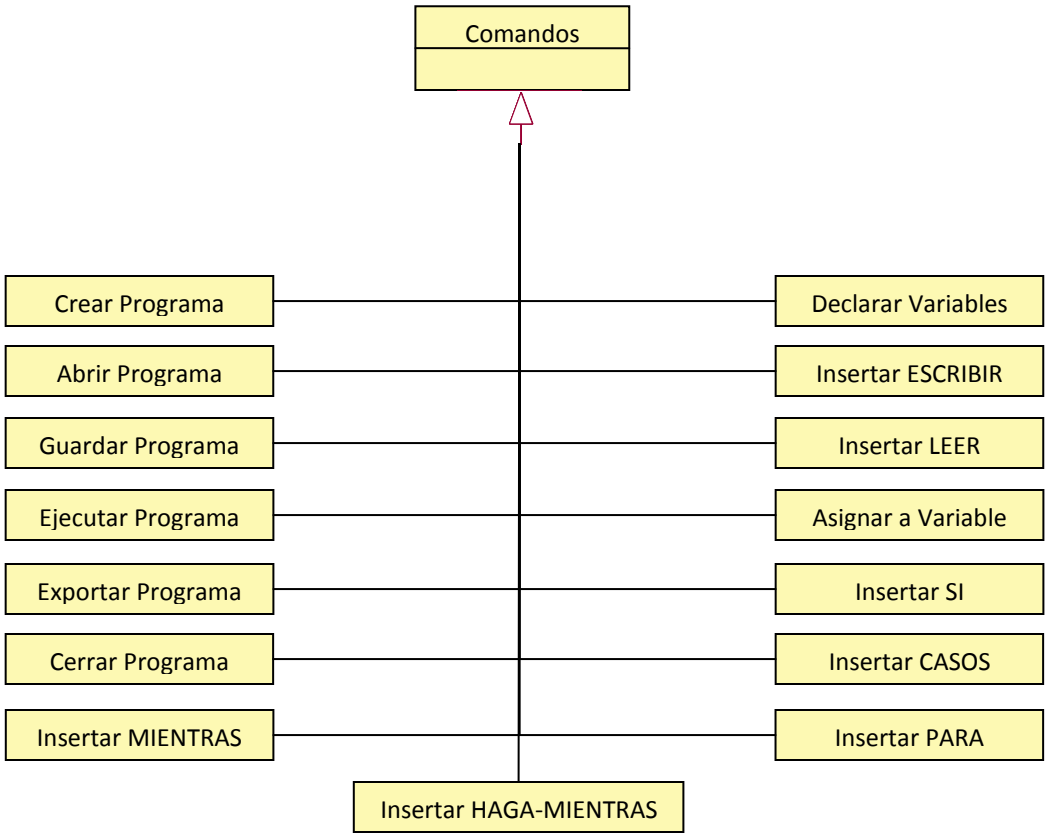
Generalizando el diagrama:

Ilustración 65: Generalización diagrama dinámico de subsistemas



Fuente: El Autor

Ilustración 66: Comandos



Fuente: El Autor

8.1.3. DESCRIPCIÓN DE SUBSISTEMAS

Los subsistemas que ya están construidos y simplemente reutilizamos, se describirán en el diseño de objetos, bajo el titulo *Ingeniería Inversa de la Herramienta de Desarrollo.*

Este es el caso de la librería GUI.

8.1.4. DISEÑO ARQUITECTÓNICO DE APLICACIONES

8.1.4.1. Modelo Estático

Se hace un diagrama que muestra los módulos del subsistema. Para la tecnología actual un módulo es una ventana o formulario. La herramienta que usamos es Diagrama de Secuencia de Ventanas. Estos Formularios se relacionan porque unos se encargan de activar a los otros.

- **8.1.4.1.1. Diagrama de Secuencia de Ventanas**

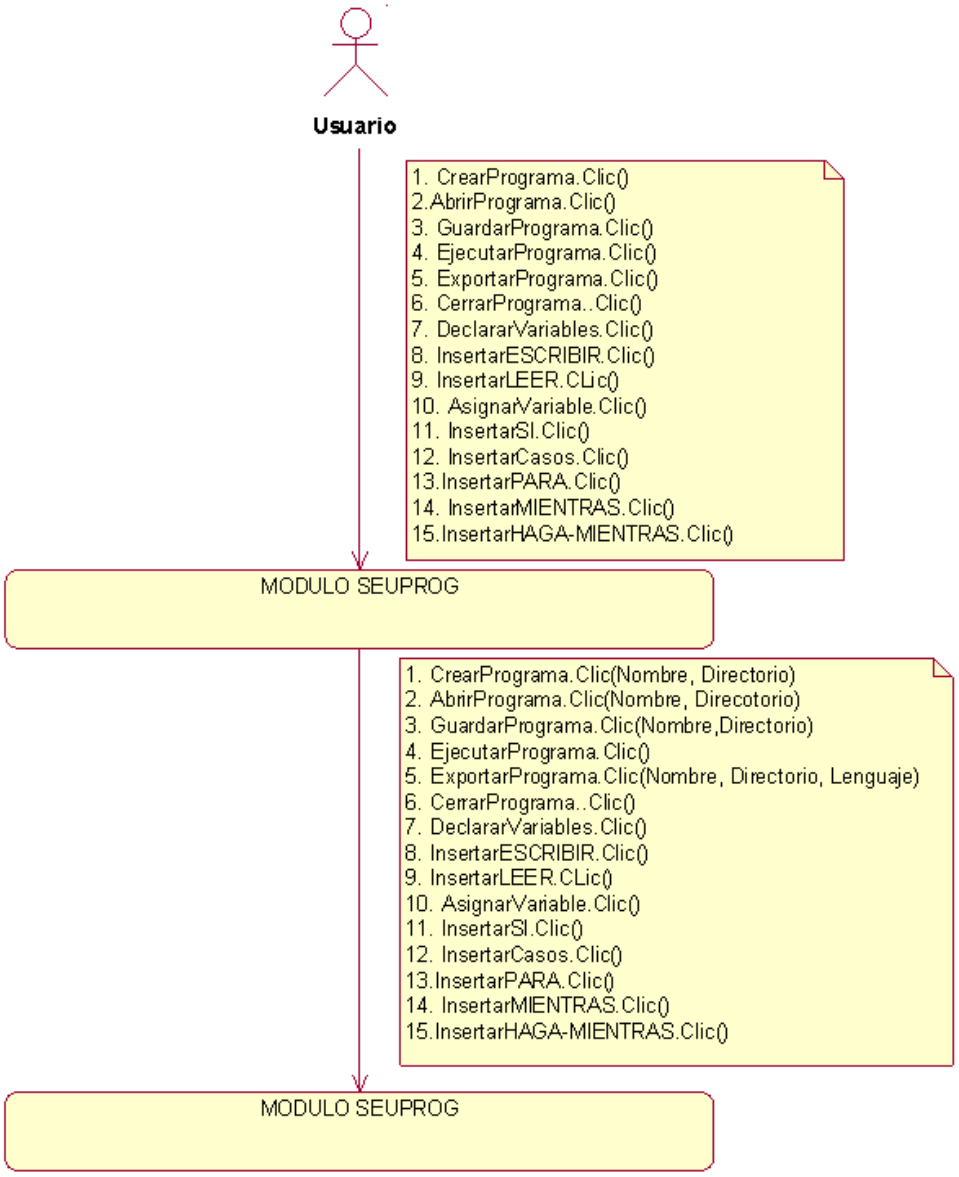
La Aplicación solo presenta una ventana, por lo tanto no es necesario modelarla, esta interfaz permite crear, editar, ejecutar y exportar Programas en seudocódigo y muestra cada uno de los comandos que puede utilizar el usuario, los errores, los resultados de la compilación y ejecución de programas.

8.1.4.2. Modelo dinámico de la Ventana Aplicación Cliente

Su objetivo es modelar los eventos que el usuario puede generarle a la ventana o módulo y los mensajes que éste módulo le envía a las clases núcleo.

- #### 8.1.4.2.1. Diagrama de Interacción Seuprog

Ilustración 67: Diagrama de interacción Seuprog



Fuente: El Autor

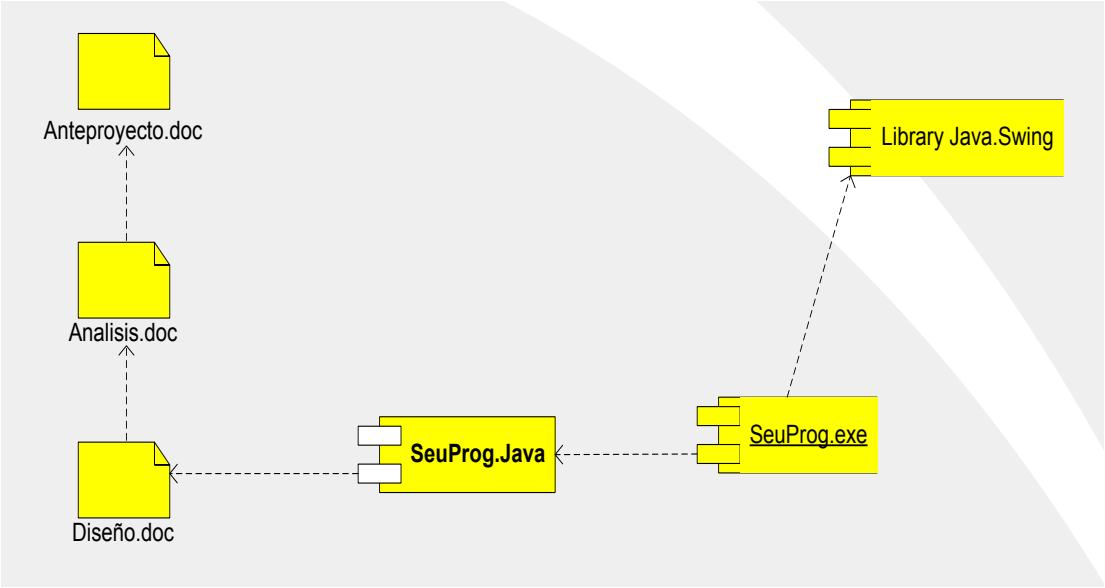
8.1.5. MODELO DE COMPONENTES FÍSICOS

Su objetivo es mostrar los componentes físicos del sistema y sus dependencias.

Un componente físico es algo que podemos ver en el disco duro. Generalmente archivos o tablas.

Es el mapa de todos los archivos y tablas que va a manejar la aplicación.

Ilustración 68: Modelo de componentes físicos

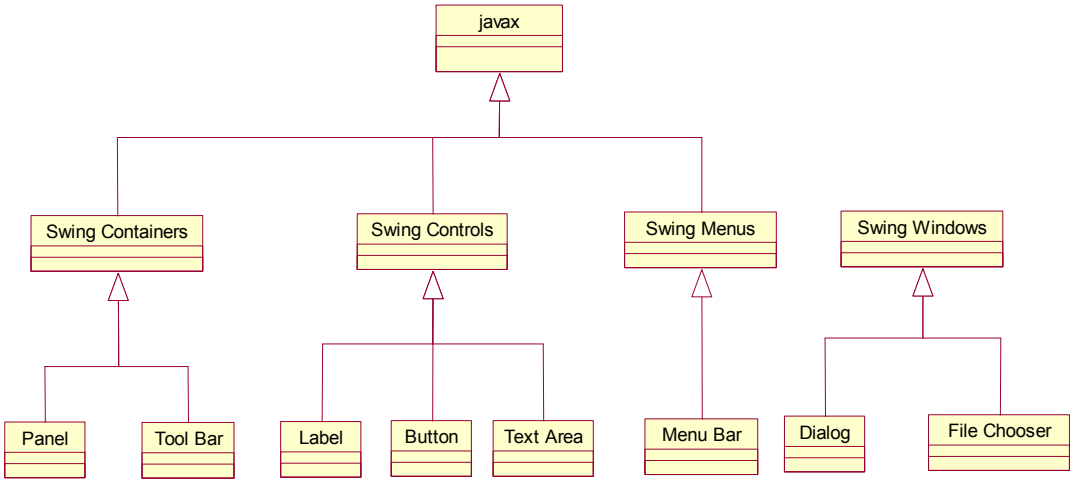


Fuente: El Autor

8.2. DISEÑO DETALLADO DE OBJETOS

8.2.1. INGENIERÍA INVERSA DE LA HERRAMIENTA NETBEANS

Ilustración 69: Ingeniería inversa de la herramienta



Fuente: El Autor

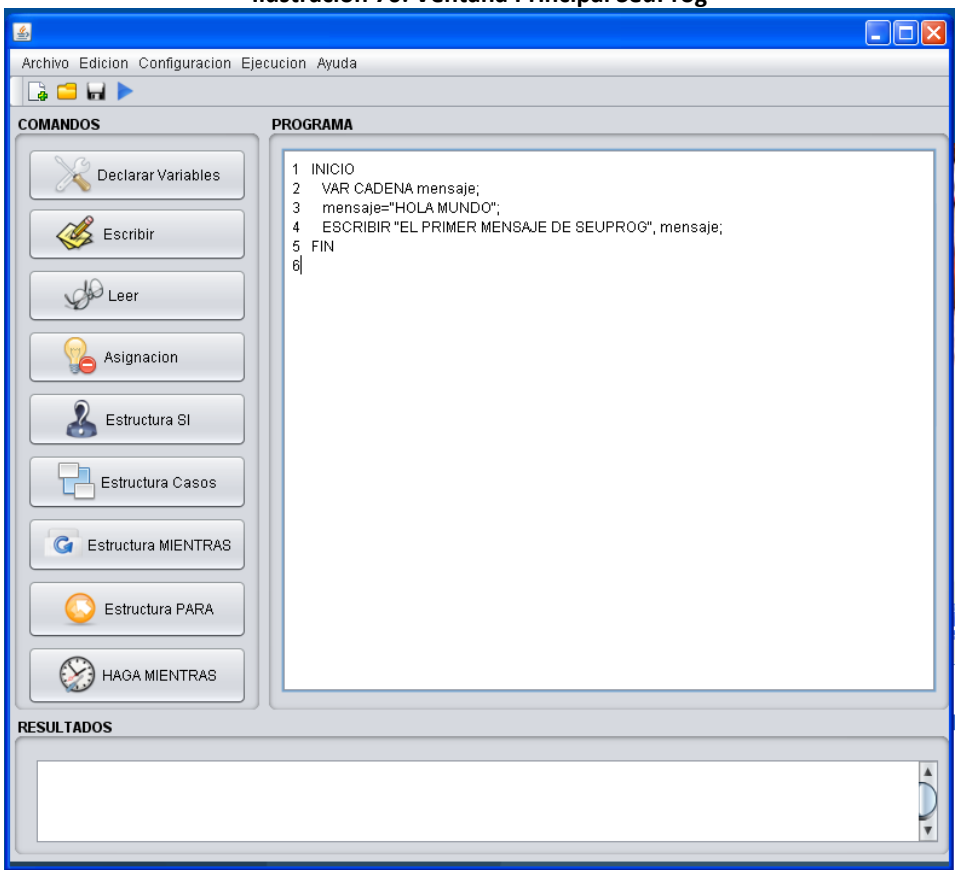
8.2.2. DISEÑO DE INTERFACES HOMBRE-MAQUINA

El diseño de interfaces Hombre-Máquina, es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles.

8.2.2.1. DISEÑO ESTÉTICO

- **Aplicación SeuProg - Desarrollo de un programa en seudocodigo**

Ilustración 70: Ventana Principal SeuProg



Fuente: El Autor

Esta ventana permite observar la interface de interacción entre el Usuario y SeuProg, muestra las diferentes opciones que permite ejecutar la aplicación.

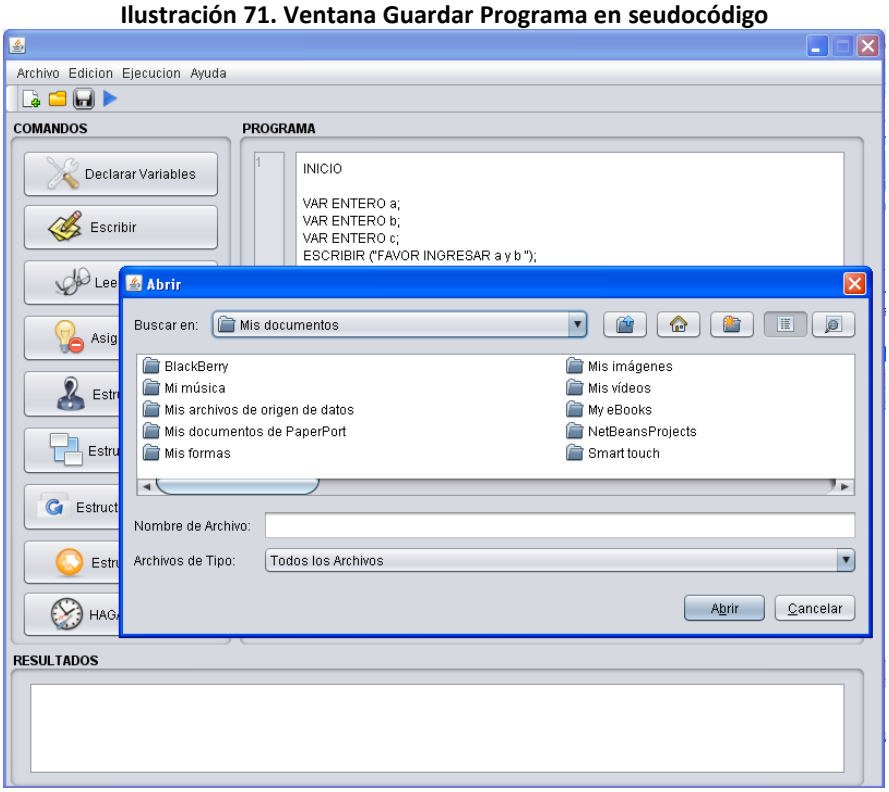
- En el panel Izquierdo o Panel de COMANDOS se encuentran los comandos que se pueden ejecutar al crear el programa en seudocódigo.

- En el panel derecho o Panel de PROGRAMA se encuentra el software en seudocódigo generado por el usuario.
- En el panel Inferior o Panel de RESULTADOS nos muestra los resultados de la ejecución de los programas creados por el usuario.

- **Aplicación SeuProg - Guardar Programa en Seudocódigo**

  Muestra un cuadro de dialogo para seleccionar el directorio y el nombre del archivo para guardar el programa como lo muestra la siguiente Figura.

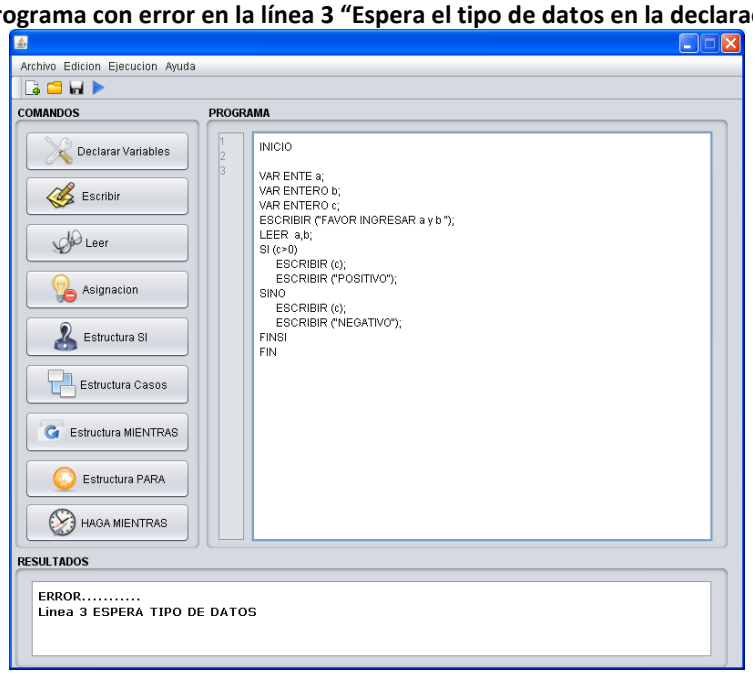
Ilustración 71. Ventana Guardar Programa en seudocódigo



Fuente: el autor

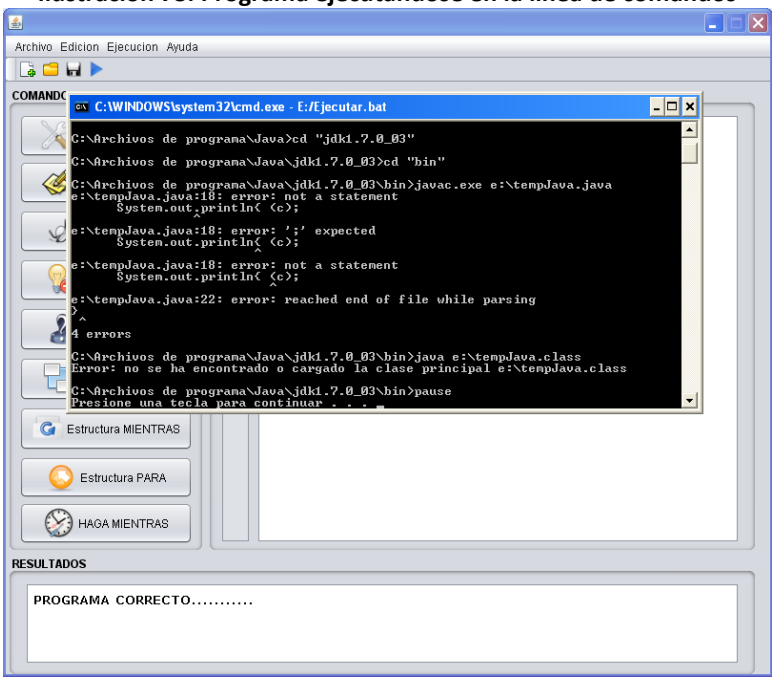
- **Aplicación SeuProg - Ejecutar Programa**
  Si un programa es correcto lo ejecuta en una línea de comandos de lo contrario muestra el error en el área de resultados. Ver figuras siguientes:

Ilustración 72. Programa con error en la línea 3 "Espera el tipo de datos en la declaración de variables



Fuente: El autor

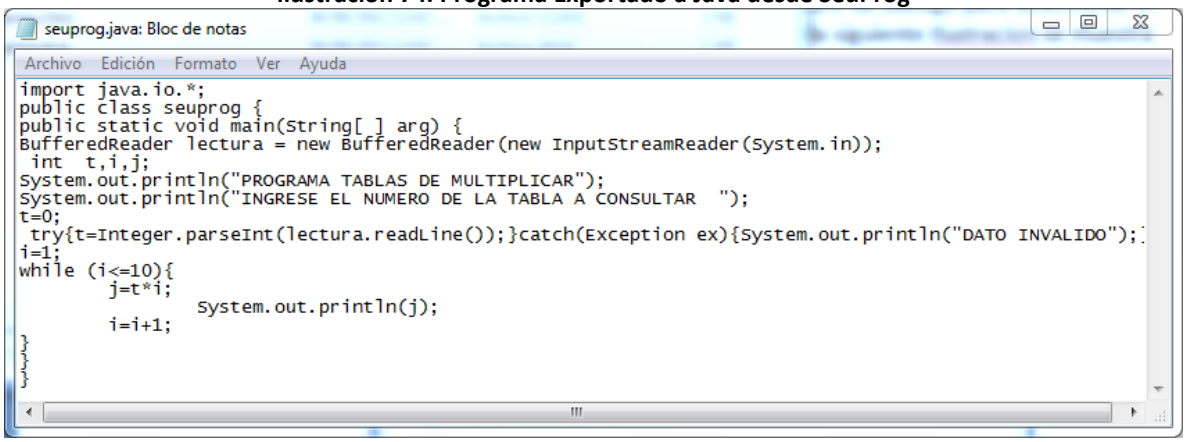
Ilustración 73. Programa ejecutándose en la línea de comandos



Fuente: El autor

- **Exportar Programa**
  Si un programa esta correcto se habilita un cuadro de dialogo para seleccionar el directorio y el nombre del archivo a exportar, A la siguiente Ilustracion se muestra un programa exportado a JAVA desde SeuProg.

Ilustración 74. Programa Exportado a Java desde SeuProg



Fuente: el Autor

- ***Insertar Instrucciones y Estructuras***
  Para insertar una Instrucción o Estructura en un programa se debe dar clic en los botones del área de comando que se muestra en la figura siguiente y SeuProg insertara la Instrucción seleccionada en la posición actual del cursor dentro del área del programa.

Ilustración 75. Comandos SeuProg

Fuente: el Autor

9. MÉTODO DE INVESTIGACIÓN

9.1 ENFOQUE METODOLÓGICO

El enfoque temático es Cuantitativo y se utilizó la recolección y el análisis de datos para verificar los problemas de aprendizaje de la asignatura Lógica de Programación confiando en el uso de la estadística para establecer con exactitud los patrones de comportamiento de los estudiantes de Ingeniería de la UTCH.

9.2 IDENTIFICACIÓN DE LA INFORMACION PERTINENTE: CATEGORIAS, VARIABLES, IDENTIFICADORES

CATEGORÍA 1: Metodología de desarrollo

	VARIABLES	IDENTIFICADORES
1.	El alumno describe el significado del uso de variables?	SI___ NO___
2.	El alumno tabula el programa para fácil entendimiento?	SI___ NO___

CATEGORÍA 2: Funcionamiento de los programas

	VARIABLES	IDENTIFICADORES
3.	El alumno descubre el algoritmo Correcto?	SI___ NO___
4.	El alumno generaliza la solución del algoritmo?	SI___ NO___
5.	El alumno utiliza delimitadores Inicio-Fin correctamente	SI___ NO___ NO PUDO EVALUAR ___
6.	El alumno comprende el objetivo del problema?	SI___ NO___
7.	El alumno logra funcionamiento del programa?	SI___ NO___ SI CON ALGUN ERROR___
8.	El alumno realiza prueba de escritorio?	SI___ NO___
9.	El alumno realiza asignaciones correctamente?	SI___ NO___
10.	El alumno comete errores de sintaxis?	SI___ NO___

CATEGORÍA 3: Calidad del diseño

	VARIABLES	IDENTIFICADORES
11.	El alumno obtiene una solución lógica?	Muy buena___ Buena___ Regular____ Mala___
12.	El alumno controla finalización ciclo repetitivo?	SI___ No___ No siempre___ No evaluado___
13.	El alumno usa conectores lógicos correctamente?	SI___ No___ No evaluado
14.	El alumno desarrolla un ciclo infinito?	SI ___ No___ No evaluado
15.	El alumno inicializa variables de condición antes del mientras?	SI___ No___ No evaluado
16.	El alumno usa indistintamente mientras y Repetir?	Sí___ No___ Sólo Repetir___ Sólo Mientras___
17.	El alumno optimiza uso de recursos?	SI___ NO___
18.	El alumno evita repetición de código?	SI___ NO___
19.	El alumno utiliza apropiadamente estructura Según?	SI___ NO___
20.	El alumno anida estructuras selectivas?	SI___ NO___

9.3 UNIVERSO O POBLACION

Estudiantes que cursan la asignatura Lógica de Programación en los programas de Ingeniería en la Universidad Tecnológica del Choco UTCH.

9.4 MUESTRA O UNIDAD DE ANALISIS

Se extrae una muestra representativa de 50 estudiantes de ingeniería que cursan Lógica de Programación.

9.5 ELABORACION, SELECCIÓN Y DESARROLLO DE INSTRUMENTOS

Se utilizo como instrumento de recolección de datos una encuesta con 20 Preguntas (Ver Anexo 1)

9.6 PRUEBA PILOTO

Se realizo la encuesta a 50 Estudiantes así:

- 25 estudiantes que vieron la asignatura sin utilizar el aplicativo SeuProg
- 25 estudiantes que vieron la asignatura utilizando SeuProg.

9.7 VALIDEZ O CONSISTENCIA

Debido a que la aplicación del instrumento documentó un buen nivel de consistencia, se justifica continuar con su uso como instrumento de evaluación de la calidad de la información contenida en las encuestas.

9.8 CONFIABILIDAD O CONGRUENCIA

95% de confiabilidad

9.9 APOYOS PARA EL PROCESAMIENTO DE LA INFORMACION

Se utilizo como apoyo para el procesamiento de la Información Archivos de tipo Hoja de cálculo que permiten representar los resultados por medio de Tablas y Gráficos.

9.10 PLAN DE PRESENTACION DE LOS RESULTADOS

Se realizara un análisis de los identificadores por variable y por categoría, de tal manera que se pueda representar gráficamente los resultados de la investigación y el logro de los objetivos del proyecto.

10. RESULTADOS DE LA INVESTIGACION

A continuación se plasma una explicación detallada de los resultados de la investigación y el efecto sobre el Grupo de estudiantes de la universidad que NO utilizaron la herramienta SeuProg **G1** y el Grupo de estudiantes que SI utilizaron la herramienta **G2**:

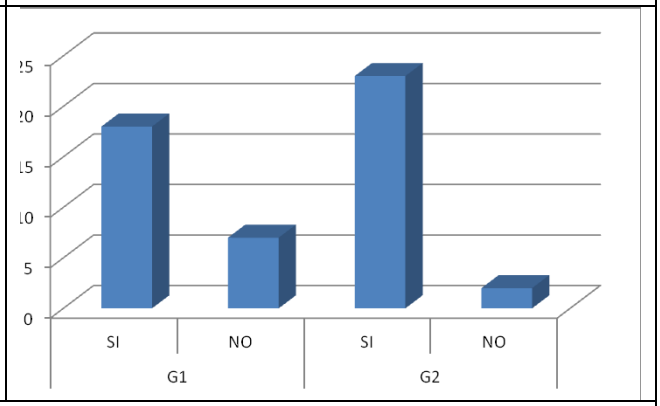
- **CATEGORÍA 1:** Metodología de desarrollo

1. El alumno describe el significado del uso de variables?

Tabla 17: Resultados de la Investigación Pregunta 1

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
18	7	23	2



ANALISIS
Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una Leve Mejora en el entendimiento del uso de las variables, esto se debe a que el aplicativo no les permite realizar algoritmos sin antes haber declarado las variables correctamente.

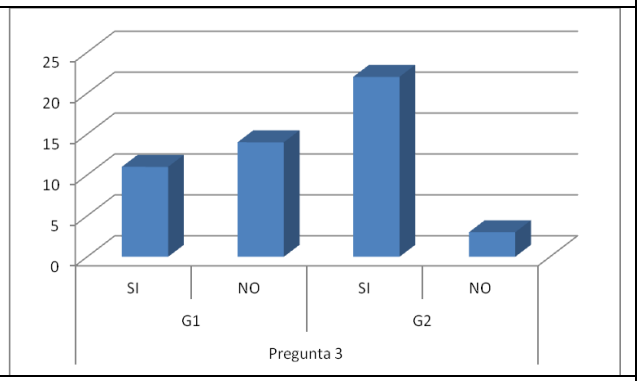
Fuente: El autor

2. El alumno tabula el programa para fácil entendimiento?

Tabla 18: Resultados de la Investigación Pregunta 2

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

Pregunta 3			
G1		G2	
SI	NO	SI	NO
11	14	22	3



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una Gran Mejora en la metodología para Tabular los Programas, Esto se debe a que el aplicativo les muestra el código tabulado y automáticamente se van acostumbrando a ver los algoritmos con ese grado de organización.

Fuente: El autor

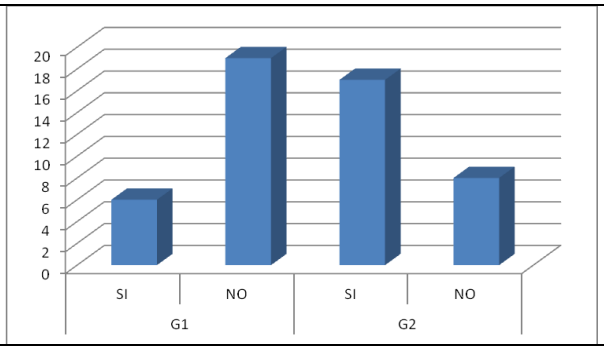
- **CATEGORÍA 2:** Funcionamiento de los programas

3. El alumno descubre el algoritmo Correcto?

Tabla 19: Resultados de la Investigación Pregunta 3

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
6	19	17	8



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora en el desarrollo de algoritmos correctos, Esto se debe a que el tiempo que se dedicaban a la redacción y revisión de la sintaxis de los algoritmos disminuye y se pueden dedicar más tiempo a la construcción del algoritmo.

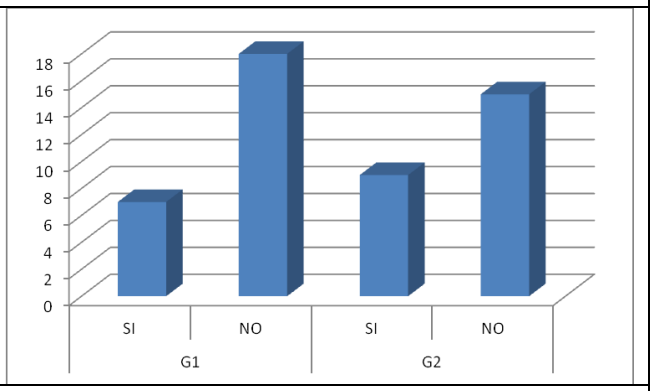
Fuente: El autor

4. El alumno generaliza la solución del algoritmo?

Tabla 20: Resultados de la Investigación Pregunta 4

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
7	18	9	15



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una MUY LEVE Mejora en la generalización de la solución del algoritmo, Esto se debe a que esta habilidad se desarrolla realizando gran cantidad de ejercicios para volver casí cotidiana la solución de problemas ya presentados con anterioridad.

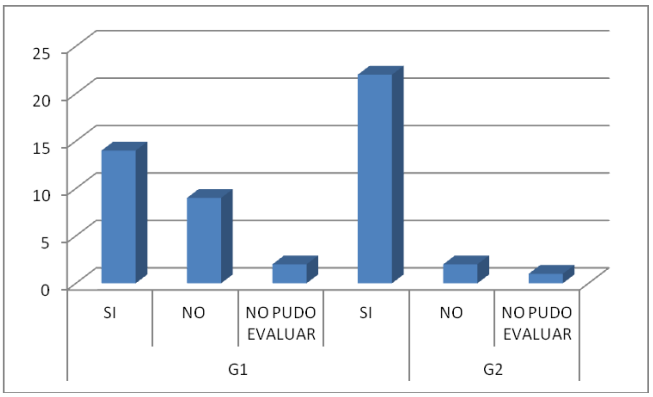
Fuente: El autor

5. El alumno utiliza delimitadores Inicio-Fin correctamente

Tabla 21: Resultados de la Investigación Pregunta 5

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1			G2		
SI	NO	NO PUDO EVALUAR	SI	NO	NO PUDO EVALUAR
14	9	2	22	2	1



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una Mejora en el uso de delimitadores INICIO-FIN, Esto se debe a que SeuProg les genera el código y les muestra el error en el momento en que se equivocan.

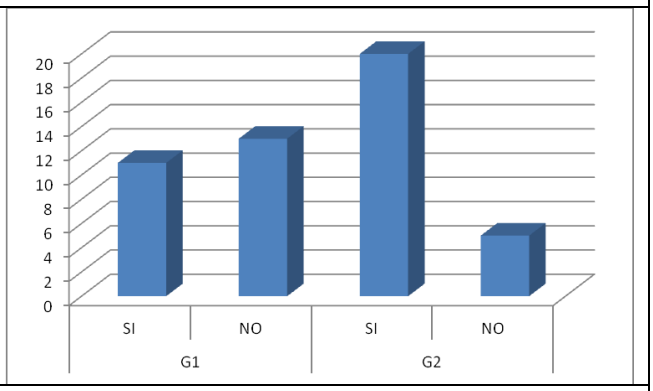
Fuente: El autor

6. El alumno comprende el objetivo del problema?

Tabla 22: Resultados de la Investigación Pregunta 6

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
11	13	20	5



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al comprender el objetivo del problema, Esto se debe a que el tiempo que se dedicaban a la redacción y revisión de la sintaxis de los algoritmos disminuye y se pueden dedicar más tiempo a analizar los requerimientos de los problemas planteados.

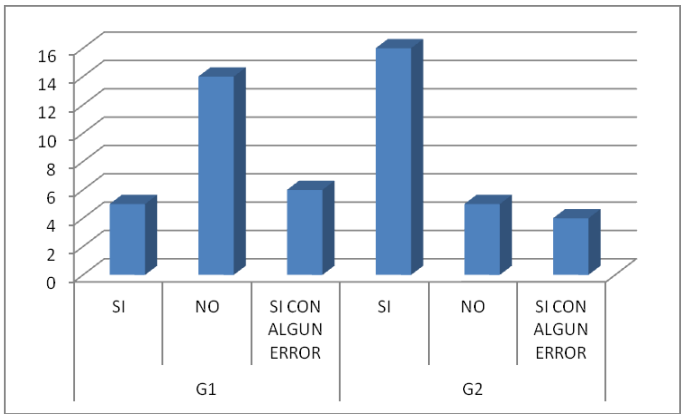
Fuente: El autor

7. El alumno logra funcionamiento del programa?

Tabla 23: Resultados de la Investigación Pregunta 7

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1			G2		
SI	NO	SI CON ALGUN ERROR	SI	NO	SI CON ALGUN ERROR
5	14	6	16	5	4



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al descubrir la naturaleza del problema, Esto se debe a que con el aplicativo SeuProg puede crear los programas y ejecutarlos para detectar errores sin limitaciones hasta llegar a la solución correcta.

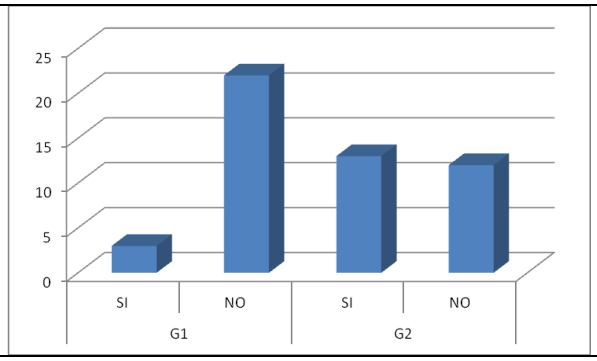
Fuente: El autor

8. El alumno realiza prueba de escritorio?

Tabla 24: Resultados de la Investigación Pregunta 8

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
3	22	13	12



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al realizar las pruebas de escritorio, Esto se debe a que el tiempo que se dedicaban a la redacción y revisión de la sintaxis de los algoritmos disminuye y se pueden dedicar más tiempo a la Realización de Pruebas para verificar el funcionamiento correcto de sus algoritmos

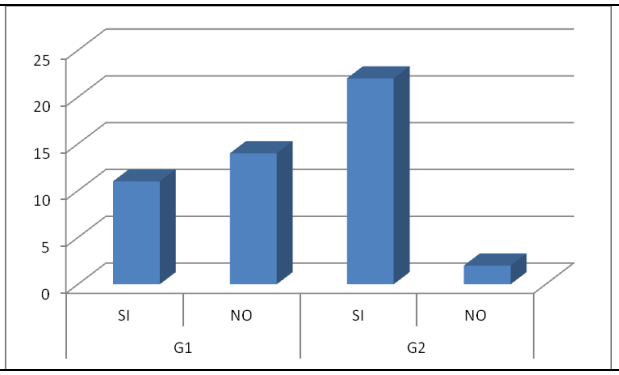
Fuente: El autor

9. El alumno realiza asignaciones correctamente?

Tabla 25: Resultados de la Investigación Pregunta 9

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
11	14	22	2



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al realizar asígnaciones, Esto se debe a que el tiempo que se dedicaban a la redacción y revisión de la sintaxis de los algoritmos disminuye por que este código es generado directamente por SeuProg, el usuario solo debe modificar datos.

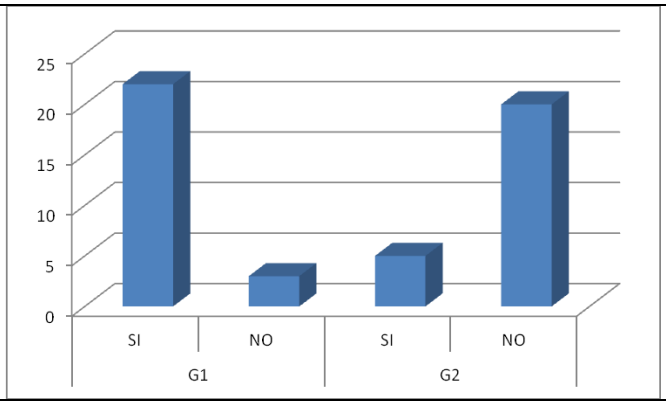
Fuente: El autor

10. El alumno comete errores de sintaxis?

Tabla 26: Resultados de la Investigación Pregunta 10

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
22	3	5	20



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Disminución en la cantidad de errores de sintaxis cometidos, Esto se debe a que SeuProg les genera el código y los usuarios solo deben editar los datos.

Fuente: El autor

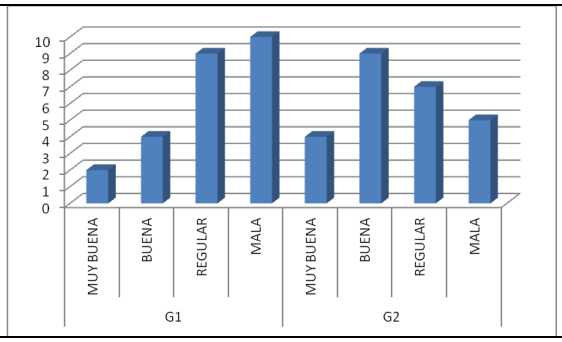
- **CATEGORÍA 3:** Calidad del diseño

11. El alumno obtiene una solución lógica?

Tabla 27: Resultados de la Investigación Pregunta 11

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1				G2			
MUY BUENA	BUENA	REGULAR	MALA	MUY BUENA	BUENA	REGULAR	MALA
2	4	9	10	4	9	7	5



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al obtener Soluciones Lógicas, Esto se debe a que el tiempo que se dedicaban a la redacción y revisión de la sintaxis de los algoritmos disminuye y se pueden dedicar más tiempo a la construcción de algoritmos desarrollando su lógica para programar.

Fuente: El autor

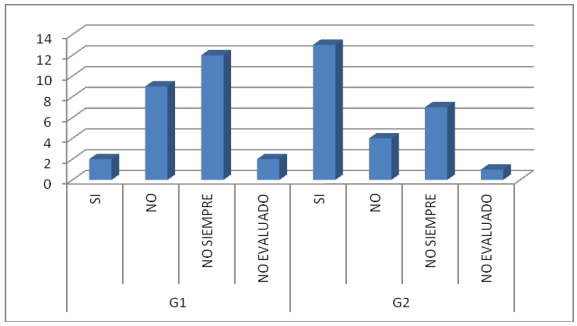
12. El alumno controla finalización ciclo repetitivo?

Tabla 28: Resultados de la Investigación Pregunta 12

TABLA DE RESULTADOS

G1				G2			
SI	NO	NO SIEMPRE	NO EVALUADO	SI	NO	NO SIEMPRE	NO EVALUADO
2	9	12	2	13	4	7	1

REPRESENTACION GRAFICA



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al Aprender a controlar correctamente la finalización de los ciclos, Esto se debe a que estructuras como él PARA solo les pide editar los limites de Inicio y Finalización de las estructuras repetitivas.

Fuente: El autor

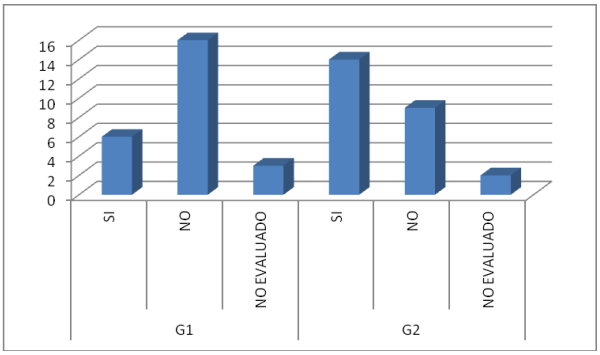
13. El alumno usa conectores lógicos correctamente?

Tabla 29: Resultados de la Investigación Pregunta 13

TABLA DE RESULTADOS

G1			G2		
SI	NO	NO EVALUADO	SI	NO	NO EVALUADO
6	16	3	14	9	2

REPRESENTACION GRAFICA



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al usar los conectores lógicos, Esto se debe a que el tiempo que se dedicaban a la redacción y revisión de la sintaxis de los algoritmos disminuye y se pueden dedicar más tiempo al análisis y la construcción de algoritmos desarrollando su lógica para programar.

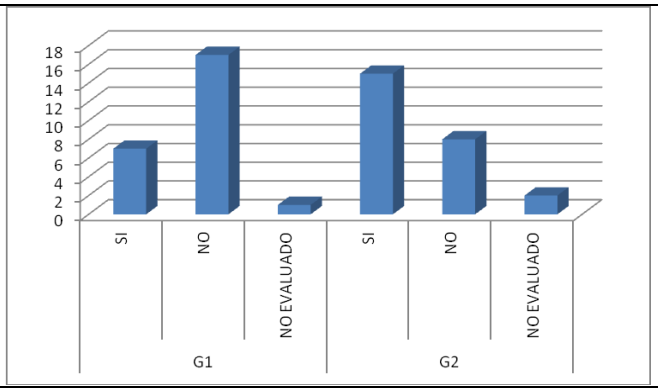
Fuente: El autor

14. El alumno desarrolla un ciclo infinito?

Tabla 30: Resultados de la Investigación Pregunta 14

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1			G2		
SI	NO	NO EVALUADO	SI	NO	NO EVALUADO
7	17	1	15	8	2



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al desarrollar ciclos repetitivos controlados, Esto se debe a que estructuras como él Mientras permiten crear ciclos infinitos voluntarios (como los utilizados para crear menús).

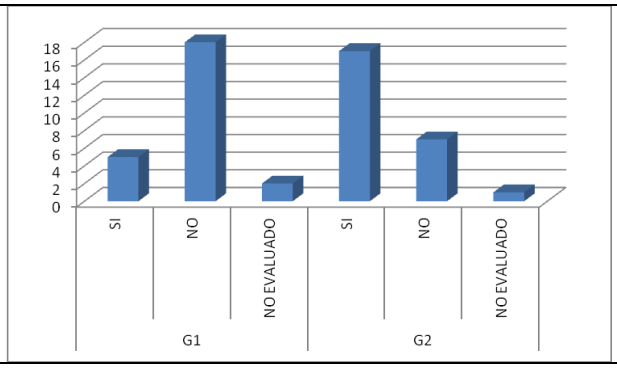
Fuente: El autor

15. El alumno inicializa variables de condición antes del mientras?

Tabla 31: Resultados de la Investigación Pregunta 15

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1			G2		
SI	NO	NO EVALUADO	SI	NO	NO EVALUADO
5	18	2	17	7	1



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al Aprender a controlar correctamente la inicialización y finalización de los ciclos, Esto se debe a la experiencia y Lógica para programar que van tomando los estudiantes al desarrollar muchos programas y esto lo permite la facilidad que entrega la herramienta SeuProg.

Fuente: El autor

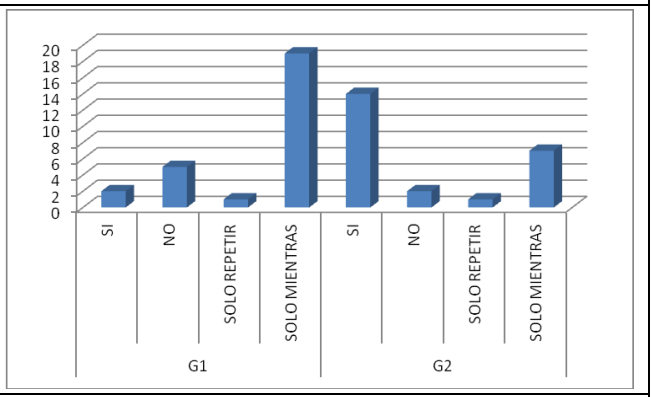
16. El alumno usa indistintamente mientras y Repetir?

Tabla 32: Resultados de la Investigación Pregunta 16

TABLA DE RESULTADOS

G1				G2			
SI	NO	SOLO REPETIR	SOLO MIENTRAS	SI	NO	SOLO REPETIR	SOLO MIENTRAS
2	5	1	19	14	2	1	7

REPRESENTACION GRAFICA



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al distinguir entre el Mientras y el Repetir, Esto se debe a la experiencia y Lógica para programar que van tomando los estudiantes al desarrollar muchos programas.

Fuente: El autor

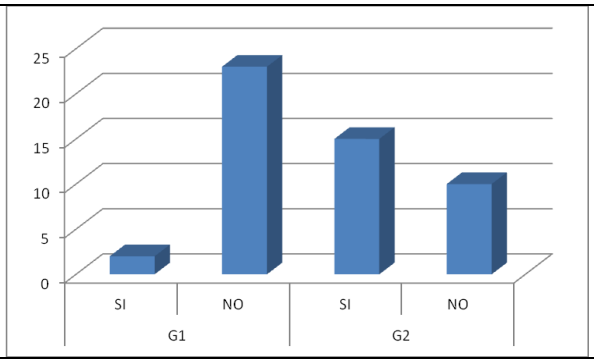
17. El alumno optimiza uso de recursos?

Tabla 33: Resultados de la Investigación Pregunta 17

TABLA DE RESULTADOS

G1		G2	
SI	NO	SI	NO
2	23	15	10

REPRESENTACION GRAFICA



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al optimizar el uso de recursos de sus programas, Esto se debe a que ellos pueden verificar en tiempo real y comparar los tiempos de ejecución de sus programas dependiendo de las sentencias y estructuras utilizadas.

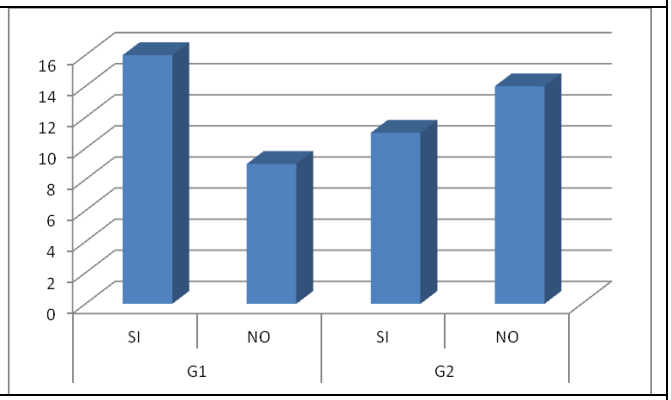
Fuente: El autor

18. El alumno evita repetición de código?

Tabla 34: Resultados de la Investigación Pregunta 18

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
16	9	11	14



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron un LEVE Descenso en evitar la repetición de código, Esto se debe a la facilidad de realizar los procedimientos de Copiar y Pegar Código del programa, mientras que el usuario que programa en papel procura buscar soluciones breves.

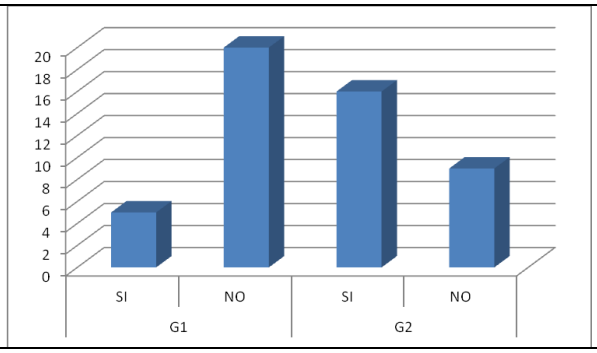
Fuente: El autor

19. El alumno utiliza apropiadamente estructura Según?

Tabla 35: Resultados de la Investigación Pregunta 19

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
5	20	16	9



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora el uso de la estructura Según, Esto se debe a que SeuProg les genera el código y los usuarios solo deben editar los datos.

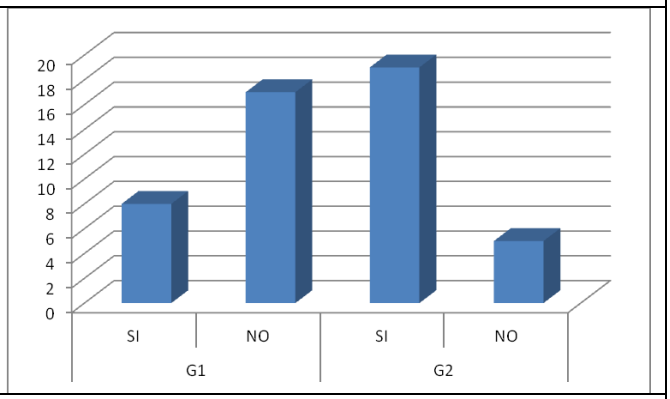
Fuente: El autor

20. El alumno anida estructuras selectivas?

Tabla 36: Resultados de la Investigación Pregunta 20

TABLA DE RESULTADOS	REPRESENTACION GRAFICA

G1		G2	
SI	NO	SI	NO
8	17	19	5



ANALISIS

Los Resultados de esta pregunta Muestran que Los usuarios que utilizaron el Aplicativo SeuProg tuvieron una GRAN Mejora al desarrollar algoritmos utilizando Estructuras Selectivas Anidadas, Esto se debe a la experiencia y Lógica para programar que van tomando los estudiantes al desarrollar muchos programas.

Fuente: El autor

Con toda esta información se obtiene como resultado lo siguiente:

- Se logro realizar un análisis comparativo de los resultados arrojados por las encuestas aplicadas a los estudiantes antes y después del uso de la herramienta, determinando que los usuarios de la herramienta Seuprog mostraron mejores resultados en casi todos los aspectos respectos a los compañeros que no la utilizaron, lo cual permite deducir que el uso de Seuprog ayuda a mejorar el desempeño académico de los estudiantes y facilita la enseñanza de la asignatura a los docentes.

- En el desarrollo del estado del arte se obtuvo información sobre estrategias y modelos utilizados en otras instituciones educativas que sirven como base para el diseño del entorno de aprendizaje y se logro conseguir un listado de aplicaciones construidas con el mismo objeto para revisar sus posibles fallos y aciertos.

- Se obtuvo la aprobación del anteproyecto para llevar a cabo su Desarrollo, Sustentación y Entrega ante el jurado.

- Se selecciono un lenguaje de programación y un entorno de desarrollo Libres que permitieron construir ágilmente la aplicación Seuprog.

- Se Consiguió desarrollar un modelo funcional del programa a través del desarrollo de un buen análisis detallado del sistema.

- Se realizo el diseño de una aplicación funcional, con buena interfaz visual y fácil de usar que permite crear correctamente algoritmos en cursos introductorios.

- Se logro obtener la traducción correcta de los algoritmos en seudocódigo a su correspondiente código fuente del lenguaje de programación JAVA.

- Se Ejecutaron pruebas de la aplicación Seuprog con estudiantes de Ingeniería de la Universidad Tecnológica del Choco y se compararon los resultados de las encuestas verificando que el nivel de aprendizaje se optimizo con el uso de la herramienta.

- Se logro plasmar un documento final y un artículo del proyecto con todo el proceso hasta llegar a las conclusiones y realizar la presentación final de la tesis de grado.

11. CONCLUSIONES

- Se determinó que al diseñar una herramienta personalizada para desarrollar algoritmos en seudocódigo en el idioma que dominan los aprendices (para este caso español) pero con sintaxis similar a la de lenguajes de programación de alto nivel, se facilita el desarrollo correcto de los programas y la familiarización de los aprendices con los lenguajes de programación avanzados para su futuro uso.

- Los Estudiantes que utilizaron la herramienta Seuprog mostraron mejores resultados en casi todos los aspectos respecto a sus compañeros que no la utilizaron, lo cual permite concluir que su uso ayuda a mejorar el desempeño académico de los estudiantes y facilita la enseñanza de la asignatura a los docentes.

- Se logró determinar que los estudiantes utilizan mucho tiempo comprendiendo la sintaxis de los lenguajes de programación dificultando encontrar una solución algorítmica a los problemas planteados.

- Se logró realizar un análisis comparativo de los resultados arrojados por las encuestas aplicadas a los estudiantes antes y después del uso de la herramienta SeuProg logrando establecer donde se encontraban los principales problemas en el aprendizaje de la asignatura para los aprendices.

- Se verifico que al realizar la traducción de la sentencia LEER de seudocódigo a JAVA era necesario trabajarlas como excepciones para controlar los posibles errores presentados durante la lectura de teclado permitiendo el flujo normal de las sentencias.

12. RECOMENDACIONES Y TRABAJOS FUTUROS

Teniendo en cuenta el proyecto SeuProg, Se recomienda tomar las siguientes acciones:

- Mejorar el proceso investigativo, ampliando los instrumentos para recolectar la información, incluyendo a docentes y estudiantes que son parte primordial del proceso educativo, y así poder direccionar la información que se genere en relación hacia el aprendizaje autónomo que ellos desarrollan empleando la herramienta SeuProg.

- Organizar una comisión de docentes de la asignatura lógica de programación que se encarguen de producir, evaluar, asesorar y divulgar, posibles utilidades o mejoras que contribuyan al desarrollo del proyecto dentro del área de programación.

13. BIBLIOGRAFIA

- AL-IMAMY, Samer. JOURNAL OF INFORMATION TECHNOLOGY EDUCATION, “On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process”, 2006. Vol 5, 7 p.

- JIMENEZ, Rey. IDENTIFICACIÓN DE PROBLEMAS DE APRENDIZAJE DE PROGRAMACIÓN CON EXPLOTACIÓN DE INFORMACIÓN. Buenos Aires: 2008. 8 p.

- MORONI, Norma. UN ENTORNO PARA EL APRENDIZAJE DE LA PROGRAMACIÓN. Bahía Blanca: 1996, 7 p.

- NOVARA, Pablo. PSEINT una invitación a entrar en el maravilloso mundo de la programación. Santa fe: 2003. http://pseint.sourceforge.net/

- POWERS, Kris. Tools for Teaching Introductory Programming: What Works?. Medford: 2006. 2 p.

- WOLVERING, Gus. Conceptos básicos de Programación, disponible en http://www.monografias.com/trabajos38/programacion/programacion.shtml

- MORENO, Robert. ANALISIS, DISEÑO E IMPLEMENTACION DE UN SOFTWARE PARA UN SALON VIRTUAL DE POQUER GRATUITO. Pereira: 2007, 135 p.

- DEBIAN, Website, Sitio Web principal del Sistema operativo Linux Debian, Disponible en http://www.debian.org/index.es.html