ELABORACIÓN DE UNA HERRAMIENTA PARA APRENDER A PROGRAMAR, QUE MITIGUÉ LA UTILIZACIÓN DE CÓDIGO DURO

Iván Alejandro Baños Delgado

Víctor Emilio Velásquez Terán

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA SEPTIEMBRE 2013

“ELABORACIÓN DE UNA HERRAMIENTA PARA APRENDER A PROGRAMAR, QUE MITIGUÉ LA UTILIZACIÓN DE CÓDIGO DURO”.

IVAN ALEJANDRO BAÑOS DELGADO

VICTOR EMILIO VELASQUEZ TERAN

Trabajo de Grado para optar al título de Ingeniería de Sistemas

Director del Trabajo:

MSc. Edwin Alexander Puerta del Castillo

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERIA DE SISTEMAS

CARTAGENA DE INDIAS D.T. Y C.

MAYO DE 2013

Cartagena, 2 de Septiembre de 2013

Señores:

COMITÉ DE EVALUACIÓN DE PROYECTOS

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

L.C.

Respetados Señores:

Luego de revisar el trabajo de grado **"ELABORACIÓN DE UNA HERRAMIENTA PARA APRENDER A PROGRAMAR, QUE MITIGUÉ LA UTILIZACIÓN DE CÓDIGO DURO"**, de los estudiantes **VICTOR EMILIO VELASQUEZ TERAN E IVAN ALEJANDRO BAÑOS DELGADO**, considero que ha cumplido con los objetivos propuestos, por lo que estoy de acuerdo en presentarlo formalmente para su calificación y así optar por el título de **INGENIERIA DE SISTEMAS**.

Atentamente,

_____

Msc. Edwin Alexander Puerta del Castillo

Cartagena, 2 de Septiembre de 2013

Nota de aceptación

_____

_____

_____

_____

Presidente del jurado

_____

Jurado

_____

Jurado

La Universidad Tecnológica de Bolívar, se reserva el derecho de propiedad intelectual de todos los trabajos de grado aprobados y no pueden ser explotados comercialmente sin su autorización.

DEDICATORIA

A Dios primero que todo, a mis padres Iván y María Claudia, a mis hermanas Natalia y Juliana, a mi novia Daniela, a mis primos y a todas las personas que siempre se preocuparon y me apoyaron a lo largo de mi carrera.

A todas aquellas personas que me han brindado su apoyo, colaboración y ánimos para poder terminar la tesis y seguir adelante.

Con mucha gratitud

Iván Alejandro Baños Delgado

DEDICATORIA

Agradezco mucho a mis padres y familiares quienes siempre estuvieron allí apoyándome en los momentos difíciles durante el desarrollo de este trabajo también aquellos amigos, amigas y personas especiales que siempre me acompañaron durante toda mi carrera e hicieron que todo esto fuera posible. Gracias Victor y Luzvenia por todo el esfuerzo y apoyo brindado.

A todos gracias

Víctor Emilio Velásquez Terán

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

- ✓ *Edwin Alexander Puerta del Castillo. Director del presente proyecto, Gracias por el apoyo y dedicación que tuviste para poder concluir este trabajo.*
- ✓ *Giovanny Vásquez Mendoza. Amigo quien inicio este proyecto con nosotros, Gracias por todos los consejos y el constante apoyo que siempre nos brindaste.*
- ✓ *Isaac Zúñiga. Profesor, Gracias por el apoyo en los momentos difíciles durante la realización de este trabajo.*
- ✓ *A nuestros familiares. Gracias por la paciencia y el apoyo constante que nos brindaron durante toda la carrera y en este trabajo.*
- ✓ *Iván de Jesús Baños Álvarez. Padre de Iván, Gracias por siempre apoyarnos en momentos de desespero y cansancio, siempre fuiste el aire en los momentos sin aliento.*

IVAN ALEJANDRO BAÑOS DELGADO

VICTOR EMILIO VELASQUEZ TERAN

TABLA DE CONTENIDO

Contenido

Tabla de Figuras

Lista de tablas

1. TÍTULO

“ELABORACIÓN DE UNA HERRAMIENTA PARA APRENDER A PROGRAMAR, QUE MITIGUÉ LA UTILIZACIÓN DE CÓDIGO DURO”.

2. FORMULACIÓN DEL PROBLEMA

En el transcurso de la investigación indagamos con docentes de las ciencias de la computación acerca del impacto que se genera en el estudiante nuevo al intentar aprender a programar. Además de la alta tasa de estudiantes que abandonan este tipo de estudios o que siempre se encuentran frustrados, podría deberse a diferentes problemas que empiezan en el inicio de la enseñanza u otros debido a problemas de motivación de los estudiantes se une la falta de un estudio a fondo de las habilidades que deben adquirir esto hace que los cursos sean más enfocados a aprender la sintaxis de un lenguaje de programación y no enfocar más en aprender los principios básicos de la programación.

3. JUSTIFICACIÓN

La presente investigación se llevó a cabo al observar la oportunidad de crear una herramienta cuyo fin sea motivar a nuevos estudiantes a interesarse por la programación. Brindándoles una nueva forma de aprender, visual y diferente que podría ayudar a obtener nuevos los conceptos de la programación de una forma diferente a la tradicional. Esta manera visual brinda una forma diferente de presentar la información para ayudar a afianzar los conceptos e ideas para reforzar la comprensión.

También se quiere disminuir la teoría de que programar es complicado o muy difícil de aprender. La parte central del problema se encuentra en la poca comprensión que tienen las personas de los lenguajes de programación pues la lógica es algo que es desarrollado por todos. La idea es hacer una herramienta que sea amigable con las personas que empiezan a programar y un entorno que les ayude a hacer de esta una tarea fácil para ellos haciendo que de la experiencia de programar algo sencillo y que los deje enfocarse en la parte lógica de su producto.

Lo que se desea es que la herramienta se utilice en los primeros semestres de todas las carreras que estén relacionadas con las ciencias de la computación, en la Universidad Tecnológica de Bolívar. Con el fin de que se maneje este estándar en la materia de algoritmos y que todos los profesores cuenten con una herramienta que les facilite la enseñanza de esta materia a los estudiantes que apenas aprenden a programar.

4. OBJETIVOS

4.1 OBJETIVO GENERAL

Crear una herramienta en la web que permita tener un entorno de programación para un lenguaje diseñado con el fin de facilitar en cierta medida la programación de código duro (hard-code) y se puedan enfocar más en el diseño en sí de la lógica de su programa.

4.2 OBJETIVO ESPECÍFICO

- Construir una herramienta que permita a los estudiantes realizar buenas prácticas de programación sin la utilización de código duro. Implementando en la aplicación una solución a las principales necesidades de los estudiantes en los procesos de aprendizaje de la programación.
- Diseñar un lenguaje de programación que permita el desarrollo de una forma intuitiva.
- Realizar análisis y diseño del software a desarrollar utilizando diagramas y metodologías.
- Construir los analizadores (léxico sintáctico y semántico) y generador de código para utilizarlos en la herramienta.

5. ESTADO DEL ARTE

Actualmente existe gran apatía de los estudiantes con las carreras relacionadas a la ciencia de la computación. Vemos a los directivos de las grandes compañías tecnológicas realizando videos de invitación para que el público se motive a estudiar la carrera. Esto se debe a que se ha estigmatizado a que todo lo relacionado con la programación es difícil y aburrido. Además de que pocos se atreven a estudiarla, lo cual genera una taza de retiro significativa.

Los estudiantes además tienen problemas con los conceptos abstractos como: Diseñar una solución a un problema, Subdividir en pequeños pedazos de código más simples, pensar en situaciones hipotéticas de error para propósitos de pruebas y encontrar errores. Incluso entendiendo los concepto más básicos como variables, tipos de datos o direccionamiento de memoria; ya que estos no tienen una representación en el mundo real [1].

Los estudios que vamos a presentar a continuación tratando de encontrar la razón del porqué se produce este fenómeno. Además para encontrar la mejor manera para atacar los problemas que existen. Incluso hay estudios que han intentado con herramientas, modelos de cursos de estudio o mezclas entre ambos resolver el problema, arrojando resultados. Basándonos en dichas investigaciones, construiremos una aplicación útil para aprender a programar de una buena manera y teniendo en cuenta buenas prácticas.

Los estudios se dividen en dos partes: la parte cognitiva (el porqué) y otra parte procedimental (como). Ambas buscan cómo resolver los problemas en el aprendizaje de la programación. La parte cognitiva busca en las actitudes de los estudiantes, su motivación, creencias y formas de estudio. La procedimental busca cuáles temas son más complicados y la forma para aprender mejor.

Uno de los problemas que se presentan a la hora de entrar a programar es la mala reputación que se han ganado lo cursos de programación. Las personas creen que estos cursos son difíciles y aburridos, porque piensan que todos los que lo estudian son “nerds” que no salen de sus casas o porque creen que se necesitan amplios conocimientos en matemáticas o razonamiento lógico. Además según winslow una persona se convierte en un experto programando en 10 años de estudio [2], lo que puede llegar a influir a una persona a no querer estudiar por el tiempo necesario para ganar un buen sueldo o tener un buen puesto.

Muchos artículos se han escrito sobre las dificultades de los estudiantes para aprender a programar [3].

Los estudiantes de programación, por lo general, al afrontar un problema se limitan a utilizar código o pseudocódigo, sin tener un plan. Esto genera que se frustren ya que al no tener un plan, no saben qué hacer al momento de presentarse un error. Tampoco sabrán que hacer sí la lógica que implementa en el programa no es la adecuada y no sabrán identificar estos problemas. Se limitan a conocimientos superficiales de los programas y por lo general crean estos, línea a línea y no resolviendo parte por parte los problemas que se presentan tienden a utilizar el conocimiento en forma específica y no general y no aplican bien los conceptos. Los estudiantes por esta razón no progresan mucho en un curso introductorio de programación [6].

Según Winslow los modelos son importantes para el conocimiento. Si el instructor no los explica bien el alumno puede crearlos en su forma que muy probablemente no sea la mejor [8].

Según Deek, Kimmel y McHugh describen el primer año de estudio de programación en crear modelos de solución de problemas. Donde se incluye el lenguaje de programación sólo en problemas específicos [4]. Una de las principales soluciones que se dieron en un principio para este problema fue el uso de la animación para reducir al mínimo las dificultades de los alumnos. En la literatura científica podemos encontrar varios programas que solamente muestran la ejecución animada de un algoritmo en específico (Dershem & Brummund, 1998; Michail, 1996) en estos los estudiantes no pueden realizar ningún cambio así como herramientas que incluso simulan/animan cualquier programa hecho por ellos.[5] De esta forma los estudiantes miraban ejemplos resueltos y creaban estructuras para solucionar sus propios problemas.

5.1 Herramientas

De las principales herramientas construidas de este tipo fue SICAS la cual simula el algoritmo y muestra sus resultados usando la animación. Además los estudiantes pueden analizar cómo se comporta el algoritmo en detalle y a su ritmo, identificar y corregir eventuales errores. SICAS no incluye contenidos teóricos, sino que consiste en un entorno de experimentación y descubrimiento, que permite la detección de errores, la corrección y el aprendizaje basado en estas actividades. En nuestra opinión, estas actividades mejoran las habilidades de resolución de problemas que

resultan en la capacidad de construir programas. En varias pruebas que confirmaron que los estudiantes que utilizaron SICAS construyen mejores algoritmos y los hizo más rápido que aquellos que no usaron SICAS [9].

En la era de estas herramientas se crearon otras como FLINT la cual utiliza diagramas de flujo estructurados para representar algoritmos. Los estudios han demostrado que los diagramas de flujo son más fáciles de entender y más rápida de evaluar que el pseudo-código para experimentados programadores. Un estudio reciente de introductoria estudiantes de programación examinó la comprensión de los estudiantes, la confianza y la velocidad al utilizar diagramas de flujo frente a Q BASIC, C/C++. Los estudiantes en el grupo de diagrama de flujo desempeño significativamente mejor con respecto a la corrección de su trabajo, el tiempo que necesitaban para completar el trabajo y la confianza que tenían en la corrección de su trabajo. El estudio también indicó que los beneficios de los diagramas de flujo aumentado con la complejidad de los algoritmos [10].

Otro de los problemas de los novatos es que tienen problemas al aprender la sintaxis de los lenguajes de programación. En muchos casos las diferencias con el lenguaje natural pueden generar problemas. Un ejemplo es que los novatos piensan que la condición en un ciclo while se evalúa expresión por expresión y no una sola vez por iteración [6]. Por lo cual en 2004 la ACM (Association for Computing Machinery) realizó una revisión del lenguaje Java, APIs (Application Programming Interface) y herramientas desde la perspectiva de la educación informática de introducción y desarrollo de una colección estable de recursos pedagógicos que le harán más fácil la enseñanza de la computación a los estudiantes principiantes con lenguajes como Java sin abrumar a los estudiantes por su complejidad, esta iniciativa se llamó la Java Task Force.[7]

Pero este tipo de herramientas no han sido la solución definitiva a este grave problema estas a su vez tenían fallas las cuales se transmitían a los estudiantes y afectaban gravemente su desempeño y proceso de aprendizaje para comprender mejor las fallas de una herramienta describimos. A continuación en un caso de estudio presentado por los estudiantes y profesores de dos universidades de Portugal: Trás-os-Montes e Alto Douro (UTAD), in Vila Real, Portugal y Higher School of Technology and Management (ESTG—Portuguese-language acronym) of the Polytechnic Institute of Leiria.

La búsqueda realizada en el estudio dejó ver tres grandes problemas: la comunicación entre los estudiantes y profesores, proceso de aprendizaje en los estudiantes y el proceso de enseñanza.

Comunicación: como los profesores explicaban los temas y como aclaraba las dudas de los estudiantes. El primer problema de comunicación es que los mensajes que aparecían dentro de la pantalla. Por estar la mayoría desenvueltas en un entorno 3D, los mensajes aparecían en donde se encontrará el avatar que representa al locutor. Por lo que se veían mensajes pero sin ninguna forma de reconocer una línea de tiempo entre estos (esto en los canales públicos de conversación). Luego se tomó el canal público para resolver dudas generales y canales privados para que se comunicaran entre si un alumno y un profesor. Los alumnos se sintieron mejor ya que podían preguntarle al profesor sin pensar en quedar en ridículo. Sin embargo esto condujo a un retraso en las respuestas por que los estudiantes no se percataban de que el profesor podría estar respondiendo las dudas de otro estudiante y el tiempo de respuesta es importante para la motivación y comprensión de los estudiantes. Se logró reducir este problema colocando frases preestablecidas para la rápida atención a problemas comunes. Estas frases pueden ser 'falta un punto y coma', 'espera un momento, estoy resolviendo otra duda', etc. Este tipo de mejoras ayudó mucho a estas aplicaciones ya que no sólo colaboraba con los docentes sino que ayudaba a los estudiantes a tener un mejor proceso de aprendizaje [11].

La aplicación que construiremos se diferencia a las descritas. Ya que las aplicaciones allí descritas están hechas para una población entre los 5 a 15 años. Estas no enseñan a programar sino que abstraen los conocimientos en la programación y se los enseñan mediante animaciones las cuales son muy difíciles de llevar a la práctica por parte de los estudiantes.

La herramienta que se construyó enseña los conceptos de forma explícita. Ya que aunque los elementos son gráficos, estos representan los bloques utilizados en la mayoría de los lenguajes de programación, su sintaxis es más familiar a estos y comprenden lo necesario para aprender a programar y no solo abstraer su teoría.

Tomando como población objetivo las personas que entran a la universidad.

6. SÍNTESIS

Teniendo en cuenta todo lo anterior se necesita llevar a cabo el desarrollo de una herramienta ágil y sencilla que permita reducir a su mínima expresión los problemas que aquí se plantean y que se han presentado en las herramientas de este tipo desarrolladas anteriormente.

La herramienta debe permitir tanto a estudiantes como a profesores la construcción y resolución de problemas además de enseñar a los estudiantes las habilidades necesarias y las buenas prácticas para aprender a programar, debe hacer énfasis en el proceso del estudiante mostrando cómo se comporta o que hace su algoritmo evitando así la frustración del estudiante el cual sin esto no sabrá si el proceso que llevó a cabo esta bien o mal.

Así como permitirle al profesor llevar un control total de proceso de aprendizaje de sus estudiantes pudiendo llevar a cabo tutorías personalizadas que le ayuden a hacer énfasis en las debilidades individuales de cada alumno. Con esto tanto estudiantes como profesores podrán tener un ambiente de trabajo diferente y más fácil de usar centrándose así en afilar las habilidades necesarias en el proceso de aprender y enseñar a programar.

7. DESCRIPCION DE LA HERRAMIENTA

La herramienta construida facilita a los estudiantes y profesores tener un ambiente de enseñanza, en la construcción y resolución de problemas. El objetivo principal de la aplicación es que los estudiantes puedan aprender los conceptos básicos de programación, además de adquirir las habilidades necesarias para programar. Se utilizará la programación estructurada ya que varios estudios mencionados con anterioridad se enfocan en que esta es la mejor manera para aprender a programar.

Aunque la aplicación este construida expresamente para aprender y no para programar, la aplicación dejará probar los algoritmos ejecutándolos o colocando casos de prueba.

El administrador se encarga de crear y eliminar los usuarios, suministrando el código y otro tipo de información. También puede crear, actualizar o eliminar cursos. Un curso tiene un nombre, una descripción, fecha de inicio y fecha de finalización, Además tiene un grupo de estudiantes y un profesor encargado. Cada estudiante tendrá una nota ligada al curso, la cual se obtiene mediante las actividades desarrolladas en el transcurso del mismo. El profesor encargado puede crear, actualizar y eliminar una actividad.

La actividad puede consistir de completar un algoritmo o realizarlo desde cero. El profesor coloca un enunciado indicando el algoritmo que se debe realizar y deja el algoritmo para que se termine o dejarlo en blanco, así como colocar casos de prueba. Los casos de prueba se evalúan ejecutando el algoritmo con las entradas dejadas por el profesor y este analiza las salidas y de ahí se obtendrá la calificación.

La nota final del curso se computara con las notas de cada actividad dependiendo de su porcentaje en el curso.

La aplicación va a permitir la utilización de diagramas de flujo para la representación de un algoritmo. Se escogen los diagramas para que disminuya la utilización de código ya que eso hace que la abstracción de los conceptos no se haga de la mejor manera y para que el principal enfoque sea planear una solución y no entrar a resolverla. Además ayuda en la motivación del estudiante ya que esta forma de representación es menos “aburrida” que con código o pseudo-código.

En la parte de edición de un algoritmo se tendrá un diagrama principal en el cual se colocará el flujo del programa dividido en globos que representan una acción. Una acción puede ser un llamado a una función, una asignación, una expresión, de lectura o escritura. Estas acciones se conectan por líneas de transición que

complementan la acción. Un ejemplo seria en una expresión de una condición revise si el valor es falso o verdadero y tomar los respectivos caminos. Además en el programa principal estará ubicada una zona donde se encontraran las variables globales del algoritmo.

Las funciones tendrán un nombre, una descripción y el tipo de dato que retorna y al ejecutar una acción sobre ellas se pasará a la pantalla de edición de una función, que será igual a la de edición del algoritmo principal.

La zona de las variables será un recuadro donde se pueda ver el nombre, el tipo de dato y su valor inicial. Al hacer clic sobre una de ellas se desplegara un cuadro de edición donde se cambiaran sus atributos. Y al ejecutar la acción de crear una nueva se abrirá un recuadro pidiendo los atributos iniciales.

La aplicación permitirá a los estudiantes probar sus algoritmos antes de la revisión. Por lo que tendrán dos opciones: los podrán ejecutar o los podrán depurar paso a paso.

Un estudiante podrá acceder a sus cursos y seleccionar las actividades que estén habilitadas por el profesor. Durante la actividad podrá chatear con los compañeros y con el profesor. Al enviar su algoritmo a calificación, el sistema calculara su nota. Además tendrá un panel donde podrá crear algoritmos sin necesidad de estar ligado a una actividad.

MARCO REFERENCIAL

8. MARCO TEORICO

8.1 DEFINICIÓN DEL LENGUAJE

El lenguaje es fuertemente tipado ya que de esta manera se facilita su comprensión y se eliminan muchos errores que serían detectados en tiempo de compilación. Como nuestro propósito es que aprendan las bases estudiantes que no tienen conocimientos acerca de programación, evitar errores de tipos en tiempo de compilación disminuye la frustración y con esto aumentaría interés.

Este lenguaje poseerá las estructuras de control, las diferentes tipos de operaciones entre ellas las de lectura y escritura, asignaciones. Que nos ayudaran en la realización de cualquier algoritmo simple. Estos algoritmos son aquellos que solucionan problemas financieros, matemáticos, estadísticos y demás.

Los lenguajes de programación abstraen las dos partes principales de la arquitectura de von Neumann. Estas son la Memoria y El procesador. La abstracción de la memoria son las variables.

8.2 VARIABLES

Una variable representa un espacio de la memoria. Pero para que estas sean útiles necesitamos un conjunto de propiedades atadas a esta. Estas propiedades, o también llamadas atributos, son los siguientes: tipos de datos, nombre o alias, alcance y el ciclo de vida. El valor que puede ser considerado otro atributo, es un conjunto de lo que está almacenado en el espacio de memoria y de cómo este debe ser tratado dependiendo del tipo de dato que se esté manejando. Con los nombres surgen varios problemas a la hora de definirlos en el lenguaje. Uno de estos

problemas es saber si los nombres son case-sensitive[1]. El otro problema es saber si sus palabras especiales son palabras reservadas o son palabras claves[2].

El tipo de dato representa el rango de valores que una variable puede almacenar además de las operaciones definidas para este tipo de dato. El valor es el contenido de la variable en la memoria que se representa con su tipo de dato.

Existen varias formas de atar variables con su tipo. Estas formas son la estática y la dinámica. La forma estática es cuando en la declaración de la variable se especifica su tipo. La atadura dinámica por el contrario no se define su tipo en el momento de su declaración sino en el momento en que se le asigna un valor.

En ambos casos se estaría hablando de un lenguaje fuertemente tipado.

Un lenguaje se define como fuertemente tipado sí sus variables tienen una atadura con un tipo durante todo su ciclo de vida. Por el contrario es débilmente tipado sí puede ser atado a varios tipos en tiempo de ejecución durante su ciclo de vida.

Las direcciones de memoria es el espacio en las celdas de la memoria al cual está asociado. Pero no se puede pensar como una celda en la memoria sino como un espacio abstracto de esta. La atadura el espacio de memoria, como con su tipo, tiene varios tipos. Pero en este caso son cuatro los diferentes tipos: estático que las ata a un espacio antes de que el programa corra y son declaradas en un sitio específico, pila dinámica es parecida a la estática solo que su declaración se puede hacer en cualquier parte del programa, pila dinámica explícita cuando se asignan

---

[1] Case-sensitive no tiene una traducción directa al español. Pero significa que dos nombres serán iguales sólo si se escriben exactamente iguales. Es decir si su combinación de mayúsculas y minúsculas son iguales. un ejemplo sería el siguiente: ivan no sería igual que Iván.

[2] Las palabras especiales pueden ser en un lenguaje de programación reservadas o claves. la principal diferencia entre ambas es que las palabras reservadas no se pueden utilizar como aliases en ningún contexto. Las palabras claves en cambio puede que existan contextos en que se puedan utilizar como nombres.

los espacios de la memoria se hace en tiempo de ejecución con instrucciones escritas por el programador y pila dinámica implícita es cuando se ata a un espacio en memoria solo cuando se le asigna un valor.

El alcance es el rango de sentencias donde la variable va a ser visible. Es decir donde puede ser referenciada. El alcance puede ser local en un bloque que es cuando es declarada dentro de este y las variables no locales son las que se utilizan dentro de un bloque pero no son declaradas dentro de este.

Un bloque es una porción de código separado del resto (comúnmente por llaves {,}).

8.3 TIPOS DE DATOS

En esta sección vamos a explicar los diferentes tipos de datos que pueden hacer parte de un lenguaje de programación. Los primeros son los numéricos.

Los tipos de datos numéricos están en casi la mayoría de los lenguajes e inclusive hay lenguajes que solo tienen tipos de datos numéricos. Los más comunes son los enteros y los computadores soportan diferentes tamaños de estos. Los siguientes son los puntos flotantes que representan los números reales y por último están los complejos aunque estos son muy raros en lenguajes de programación.

Hay un tipo de dato numérico especial. Estos son los decimales los cuales guardan un número de decimales especificado utilizados en operaciones de negocios.

Otro tipo son los de carácter que se almacenan con una codificación numérica. Esta codificación comúnmente es la ASCII. Una variación de los caracteres son las cadenas de caracteres. Las cadenas como su nombre lo dice representa una consecución de caracteres formando palabras. Estas tienen diferentes tipos de operaciones específicas como lo son la concatenación, comparación entre otras.

Por último otro tipo de dato son los booleanos o lógicos. Los lógicos pueden tomar dos valores que son verdaderos y falsos. Estos comúnmente representan banderas o switches en los programas.

Los siguientes tipos son tipos de datos compuestos. Estos pueden ser asociados fácilmente con un conjunto de valores numéricos. Entre ellos están las enumeraciones, arreglos y matrices.

8.4 EXPRESIONES Y SENTENCIAS DE ASIGNACIÓN

8.4.1 EXPRESIONES ARITMÉTICAS

Las expresiones aritméticas son las encargadas de representar la computación aritmética. Es decir, todas las funciones predefinidas con números las cuales son la suma, resta, multiplicación y división principalmente. El resultado de estas operaciones es de tipo numérico ya sea entero flotante o complejo.

La forma de evaluar las operaciones aritméticas depende de la precedencia de los operadores y no simplemente de derecha a izquierda.

8.5 CONVERSIÓN DE TIPOS

Existen dos tipos de conversiones Explícita e Implícita. Las conversiones implícitas se hacen cuando hay promoción de tipos. Es decir, cuando pasa de un mismo tipo de datos de menor rango a uno de mayor rango. Un ejemplo seria cuando se le asigna a una variable de tipo long una expresión que retorna un valor de tipo int. Las explícitas es cuando se dice explícitamente a qué tipo de dato se va a convertir.

8.6 EXPRESIONES BOOLEANAS Y RELACIONALES

Las expresiones booleanas son las que evalúan dos expresiones de tipo booleano y como resultado nos da un valor de tipo booleano. Los operadores más comunes son Y, O, igualdad y desigualdad. Las expresiones relacionales son las que evalúan dos expresiones numéricas y dan como resultado un valor booleano estas son el mayor, menor, mayor que y menor que.

8.7 EVALUACIONES DE CORTO CIRCUITO

En ciertas ocasiones las operaciones booleanas no es necesario evaluarlas completamente para saber el resultado. Por lo que las evaluaciones de cortocircuito lo que hacen es evaluar hasta cierto punto donde ya se tenga certeza del resultado. Un ejemplo es el siguiente: cuando se evalúa lo siguiente verdadero O falso no se tiene que llegar a la segunda expresión para saber que la expresión arroja como resultado verdadero.

8.8 ESTRUCTURAS DE CONTROL

8.9 ESTRUCTURAS DE SELECCIÓN

- **Sí - Entonces:** La instrucción de selección sí realiza una acción si la condiciones verdadera, y sí la condición es falsa evita la acción.

- **Sí - Sí No:** Esta instrucción evalúa una condición y sí es verdadera realiza una acción. De no ser así ejecuta otra acción diferente.

- **Conmutador:** Esta estructura de datos tiene un conjunto de acciones representadas cada una por una llave la cual se selecciona dependiendo de una instrucción.

8.10 ESTRUCTURAS DE REPETICIÓN

- **Mientras:** esta estructura de control realiza una acción mientras se cumpla una condición. Sí la condición es inicialmente falsa no se ejecuta la acción ninguna vez.

- **Para:** esta estructura es parecida al mientras solo que en su declaración se decide a qué variable va a afectar y de qué forma va a ser afectada.

8.11 FUNCIONES

Las funciones son una porción de código que puede ser llamado desde cualquier parte del programa tomando el control de este y al finalizar le devuelve el control a él subprograma anterior.

Las funciones tienen varios propósitos entre los cuales esta subdividir el programa para facilitar su implementación. En este punto en específico las funciones nos aportaran el mayor apoyo al momento de la creación del lenguaje.

Las funciones como las variables tienen atributos y ataduras. En este aparte mencionaremos las más importantes para nosotros que son el tipo de valor de retorno, el nombre y la lista de parámetros.

8.12 ANALIZADORES

Para la construcción de un lenguaje de programación se debe construir varios analizadores. El analizador léxico o lexer, el analizador sintáctico o parser y el analizador semántico el cual consta de varias partes como se describirán a continuación.

8.13 ANALIZADOR LÉXICO

El analizador léxico se encarga de leer la secuencia de caracteres del programa fuente, carácter a carácter, y los agrupa para formar unidades con significado propio llamados tokens.

Estos tokens representan los identificadores, palabras reservadas del sistema, operadores y símbolos especiales. Los tokens diferencian cada una de las palabras que se escriban en un programa dándole una especialidad. Los tokens por lo general son representados por expresiones regulares.

Por lo general los errores hallados aquí son de tipado.

8.14 ANALIZADOR SINTÁCTICO

El analizador sintáctico es el encargado de recoger los tokens del analizador léxico y agruparlos en reglas definidas. Estas reglas por lo general son representadas por una gramática libre de contexto o reglas EBNF. Estas reglas hacen que el programa tenga un sentido. El resultado de este análisis nos arroja un árbol sintáctico o una estructura jerárquica.

Los errores en esta parte son de sintaxis. Unos ejemplos son no encontrar un token en una secuencia o encontrarse con un token que no valla de acuerdo con la regla.

8.15 ANALIZADOR SEMANTICO

El analizador semántico se encarga de la verificación de tipos por lo cual es necesario tener un sistema de tipos.

El sistema de tipos consta de dos partes la tabla de símbolos y el verificador de símbolos. La tabla de símbolos es el repositorio donde se encuentra la información de las variables y funciones. El verificador de tipos opera sobre esa tabla de símbolos para verificar que se estén usando bien estas variables y funciones dentro del programa.

El resultado de este analizador es un árbol por lo general de sintaxis abstracta que luego se utilizara para la traducción del código ya sea a un lenguaje de maquina o un lenguaje aún más bajo.

9. APLICACIÓN WEB

la aplicación web contará con un entorno de desarrollo que le facilitara al estudiante el diseño de algoritmos, así como bloques de código prediseñados que le permiten arrastrar y colocar código en su algoritmo previniendo de esta forma la producción de errores sintácticos, esta le permitirá compilar y le mostrará los errores que se encuentren en su código así como el resultado de su esto le brindara la seguridad al estudiante de que el procedimiento que está realizando se encuentra bien planteado, este tipo proceso brindará al estudiante una mejor experiencia de lo que es la construcción de un algoritmo.

10. CONSTRUCCIÓN DEL ENTORNO

La construcción del entorno se realizó bajo la metodología Scrum, que es un marco de trabajo en el que equipos cross-funcionales pueden crear productos o desarrollar proyectos de una forma iterativa e incremental. El desarrollo se estructura en ciclos de trabajo llamados Sprints (también conocidos como iteraciones). Los Sprints están acotados en el tiempo finalizan en una fecha determinada independientemente de si el trabajo ha finalizado por completo o no, y jamás se prorrogan [14].

Para hacer la aplicación primero se definirá la sintaxis del lenguaje. El lenguaje tiene que poder convertirse en un diagrama de flujo y viceversa. Esto para que el estudiante no aprenda la sintaxis del lenguaje e inclusive le sea transparente, sino que aprenda a utilizar los diagramas de flujo en la aplicación para aprender a programar. Se creara el compilador utilizando la herramienta ANTLR y se traducirá a el lenguaje java, para utilizar su compilador.

Para la creación del entrono web se construirá en java con el framework java serverFaces(JSF) la cual es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE (Java Platform, Enterprise Edition) y un complemento llamado PrimeFaces. Es un framework que cuenta con un rico conjunto de componentes que facilitan la creación de las aplicaciones web, el cual nos ayudara para la realización del entorno de la aplicación con lo cual se generara una aplicación muy intuitiva y fácil de usar y que le facilita al estudiante la creación de su algoritmo mediante ayudas como bloques de códigos predispuestos en el lenguaje.

Para la creación del algoritmo se utilizara jsPlump. Este complemento de JQuery proporciona un medio para que se pueda generar diagramas de flujos. Esta

herramienta permite conectar visualmente los elementos en sus páginas web. Este complemento sirve para la gran mayoría de navegadores utilizando Jquery. Para los navegadores que no funciona Jquery completamente, como Internet Explorer 8, Utiliza SVG (por sus siglas en inglés) Gráficos Vectoriales Redimensionables en los navegadores modernos y VML (Vector Markup Language).

Se escogió esta debido a la facilidad que brinda para la creación de la estructura del algoritmo y debido a su licencia de tipo MIT (Massachusetts Institute of Technology) que nos permite la utilización de este software de forma completamente gratuita diferente a otras opciones encontradas en el mercado.

En la parte del manejo de los datos utilizamos el motor MySQL el cual es de software libre y que nos permite la construcción del sistema de base de datos relacional así como el manejo de los datos desde la parte del servidor.
Del lado del servidor la aplicación contendrá el traductor hecho en ANTLR el cual traducirá a Java y luego se utilizará el compilador de Java, el cual compilara el código y devolverá al cliente el resultado que el algoritmo plantee o en su defecto los errores que este contenga.

11. REQUERIMIENTOS

GESTIONAR USUARIOS

AUTENTICACION DE USUARIOS DEL SISTEMA

ADMINISTRAR CURSOS

- Crear curso.
- Actualizar curso.
- Listar cursos.
- Mostrar detalle del curso.
- Eliminar curso.

ADMINISTRAR ESTUDIANTES EN UN CURSO

- Agregar estudiantes.
- Sacar estudiante del curso.
- Listar estudiantes curso.

GESTIONAR VARIABLES DEL CURSO

- Obtener nota final del curso.
- Pantalla de estadísticas.

ADMINISTRAR ACTIVIDADES

- Crear actividad.
- Editar actividad.
- Obtener nota de actividad.
- Listar actividades de un curso.

GESTIONAR VARIABLES DE LA ACTIVIDAD

- Calificar actividad.
- Agregar caso de calificación

GESTIONAR DRIAGRAMA DE FLUJO DEL ALGORITMO

- Agregar bloque.
- Eliminar bloque.
- Editar bloque.
- Agregar variable.
- Editar variable.
- Eliminar variable.

ADMINISTRAR ACCIONES SOBRE EL ALGORITMO

- Ejecutar algoritmo.
- Guardar algoritmo.
- Probar los casos de calificación del algoritmo.

12. **DISEÑO METODOLÓGICO**

1. En la primera fase de nuestro proyecto construimos y diseñamos el lenguaje de programación basándonos en la principal necesidad que es la construcción de un lenguaje fácil de utilizar y que se acomode a las estructuras de diagrama de flujos respetando los principios principales de todo lenguaje de programación.
2. Luego de culminar este punto se construyó las herramientas, los analizadores y la generación de código en este proceso fue vital mantener todos los puntos analizados en el diseño para lograr una construcción óptima y acorde a las necesidades.

3. Esta construcción se realizó mediante las siguientes fases:
    - Fase I: Diseño del lenguaje, este debe estar acorde con las necesidades de la población objetivo.
    - Fase II: Análisis de los requisitos, formulación de los requerimientos del software deseado y análisis de los mismos para el fin de la aplicación.
    - Fase III: Construcción de los analizadores y generador de código, esta es una fase crítica ya que sin esta no se puede continuar con los siguientes pasos de la construcción de la herramienta.
    - Fase IV: Construcción de la herramienta.

13. CRONOGRAMA

Tabla

	OBJETIVO	ACTIVIDADES	NOV	DIC	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO
1	Construir de una herramienta que permita a los estudiantes realizar buenas prácticas de programación sin la utilización de código duro.	Diseño del lenguaje de programación										
		Construcción de los analizadores y generadores de código										
2	Implementar en la aplicación una solución a las principales necesidades de los estudiantes en los procesos de aprendizaje de la programación.	Implementación de la aplicación										
		Análisis y definición de requerimientos y construcción de documentos técnicos.										

Tabla 1. Cronograma de actividades

14. RECOMENDACIONES Y LIMITACIONES DEL ESTUDIO

En esta sección se realizan las recomendaciones de estudio, las limitaciones del estudio y por último las sugerencias para futuras investigaciones.

14.1 Recomendaciones

La principal recomendación se puede tomar esta aplicación para realizar un estudio detallado y estadístico que indique que efectivamente la utilización de diagramas facilita la obtención de un conocimiento claro de las bases de programación.

Se recomienda también incluir esta aplicación dentro de los programas de ingeniería, en los cursos de programación y algoritmos.

14.2 Limitaciones del Estudio

Las limitaciones de este estudio o este trabajo, por falta de tiempo no se logró realizar un estudio detallado para corroborar que la aplicación cumpla con su objetivo.

14.3 Sugerencias para Futuras Investigaciones

De acuerdo a las experiencias y conocimientos adquiridos en el desarrollo de este trabajo se considera apropiado sugerir los siguientes trabajos:

Desarrollar e implementar un módulo de integración con plataformas de ambientes educativos virtuales.

Realizar una investigación que busque datos estadísticos que corroboren que la aplicación ayuda a facilitar la obtención de los conocimientos necesarios en el ámbito de la programación de una manera más sencilla que con otros métodos.

15. CONCLUSIONES

Se logró crear la herramienta en una primera fase, cumpliendo con la funcionalidad establecida. La aplicación permitirá crear diagramas de flujo para representar los algoritmos. También brindará un ambiente de trabajo para estudiantes y profesores que faciliten la realización y revisión de actividades de programación.

Buscando maximizar las buenas prácticas de programación la aplicación no permitirá dejar al libre albedrio la utilización de los bloques. Por el contrario se brindaran bloques preestablecidos haciendo de esta la única manera de construcción de los algoritmos.

Basándonos en los estudios e investigaciones mencionadas anterior mente en este documento, la utilización de diagramas de flujo es la mejor forma para abstraer los conocimientos de programación.

REFERENCIAS

[1] Micaela Esteves, Benjamim Fonseca, Leonel Morgado and Paulo Martins (2010), Improving teaching and learning of computer programming through the use of the Second Life virtual world, British Journal of Educational Technology.

[2] Winslow, L. E. (1996). Programming pedagogy—a psychological overview. SIGCSE Bulletin

[3] ECKERDAL, ANNA. "Novice Students' Learning of Object-Oriented Programming". Dissertation for the degree of Licentiate of Philosophy in Computer Science with specialization in Computer Science Education Research. UPPSALA UNIVERSITY, DIVISION OF SCIENTIFIC COMPUTING, 2006.

[4] Anthiny Robins Janet Rountree, and Nathan Rountree, Learning and teaching programming: A review and discussion, Computer Science, University of Otago, Dunedein, New Zealand. Computer Science Education, 2003, Vol. 13. No. 2, pp 137-172.

[5] Micaela Esteves, Benjamim Fonseca, Leonel Morgado and Paulo Martins British Journal of Educational Technology, 2010.

[6] Kirsti Ala-Mutka (2004), *PROBLEMS I N LEARNING AND TEACHING PROGRAMMING*

- a literature study for developing visualizations in the Codewitz-Minerva project, Institute of Software Systems, Tampere University of Technology, Finland.

[7] http://jtf.acm.org/, August 25, 2006

[8] Anthiny Robins Janet Rountree, and Nathan Rountree, Learning and teaching programming: A review and discussion, Computer Science, University of Otago, Dunedein, New Zealand. Computer Science Education, 2003, Vol. 13. No. 2, pp 137-172.

[9] Integrating Educational Tools for Collaborative Computer Programming Learning, Crescencio Bravo, Maria Jose Marcelino, Anabela Gomes, Micaela Esteves, Antonio Jose Mendes, Journal of Universal Computer Science, vol. 11, no. 9 (2005), 1505-1517

[10] Bowling Green, The Flowchart Interpreter for Introductory Programming Courses Thad Crews Uta Ziegler Department of Computer Science Western Kentucky University, KY 42101

[11] CONCEPTS OF PROGRAMMING LANGUAGES, Robert Sebesta, 9na edición, capítulo 5, Names, Bindings, and Scopes.

[12] http://informatica.uv.es/docencia/iiguia/asignatu/2000/PL/2008/tema2.pdf

[13] Natural Language Syntactic Analyzer with Editable Rules for Generating UML Primitives. Carlos M. Zapata, Ph.D. y Juan C. Hernández, Ing.
Grupo de Ingeniería de Software, Escuela de Ingeniería de Sistemas, Universidad Nacional de Colombia

[14] http://www.scrumprimer.org/primers/es_scrumprimer20.pdf

Anexo 1

Diagrama Casos de uso

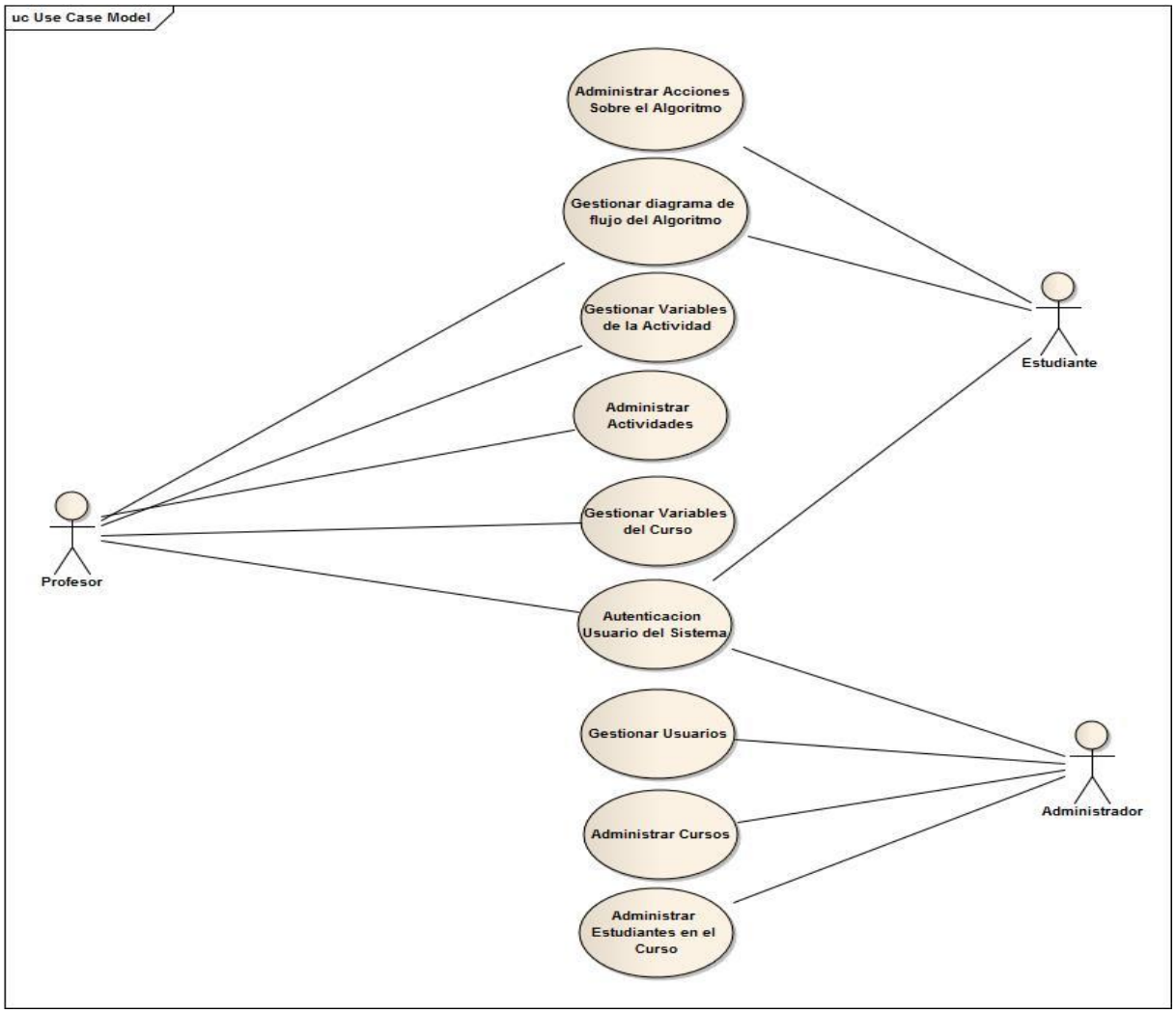


Figura 1. Diagrama de casos de uso de la aplicación

Anexo 2

Diagramas de Actividad:

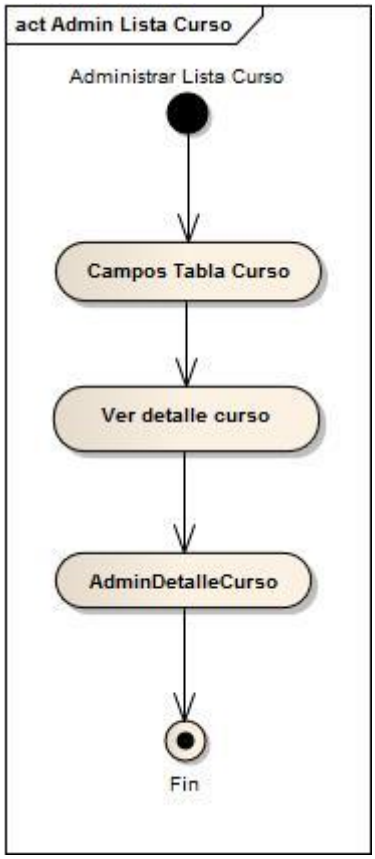


Figura 2. Diagrama de actividad administrar lista del curso

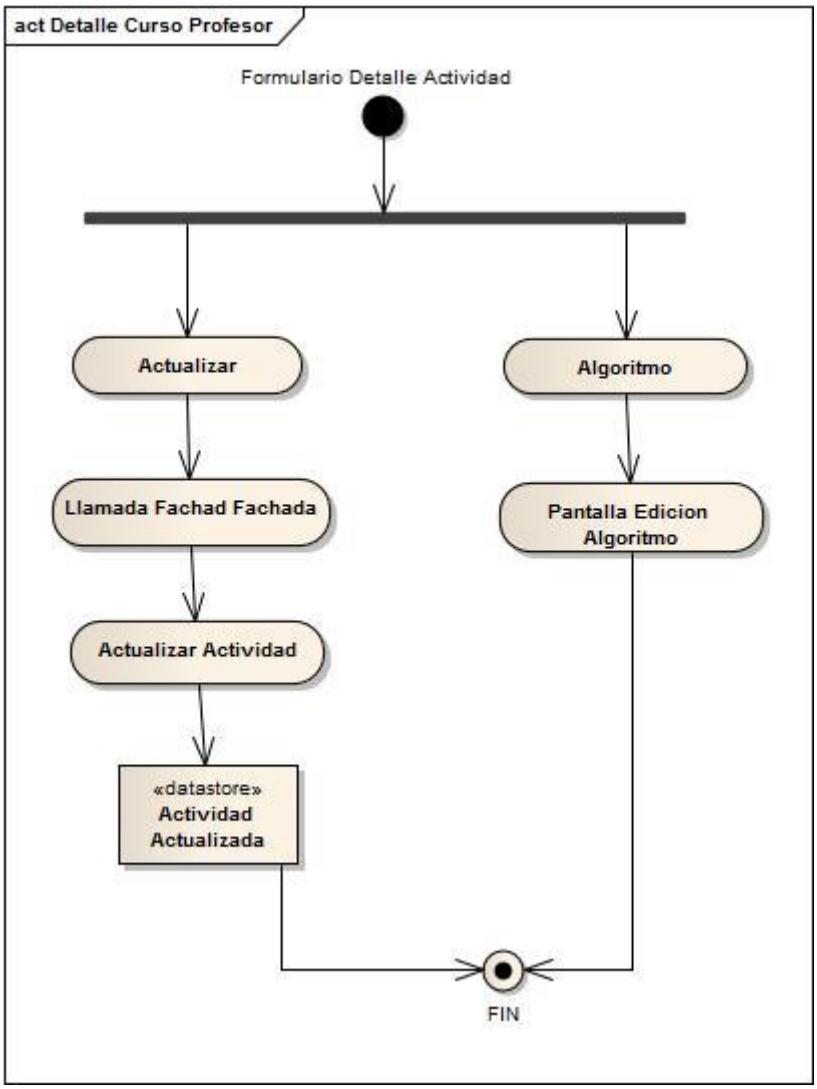


Figura 3. Diagrama de actividad formulario detalle actividad.

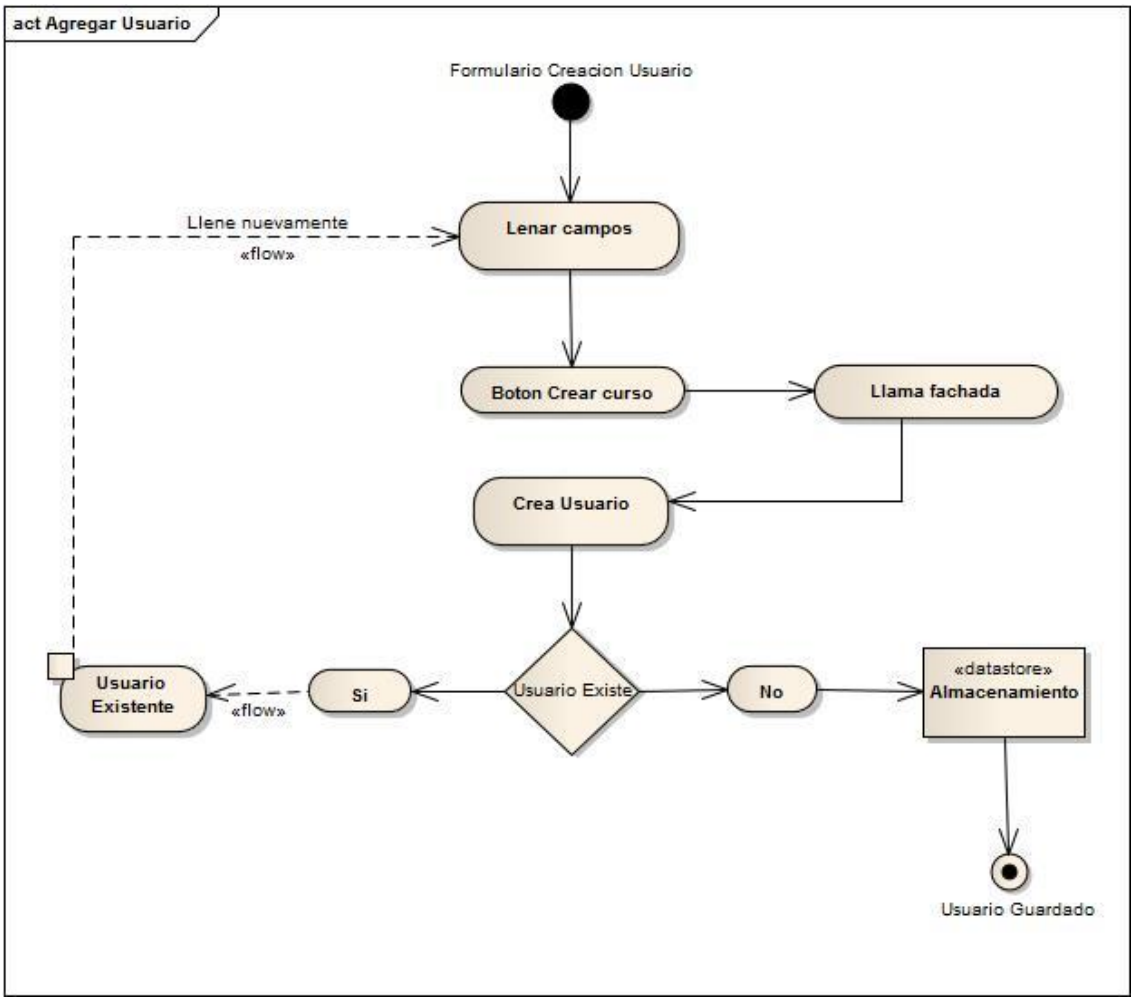


Figura 4. Diagrama de actividad formulario creación de usuario

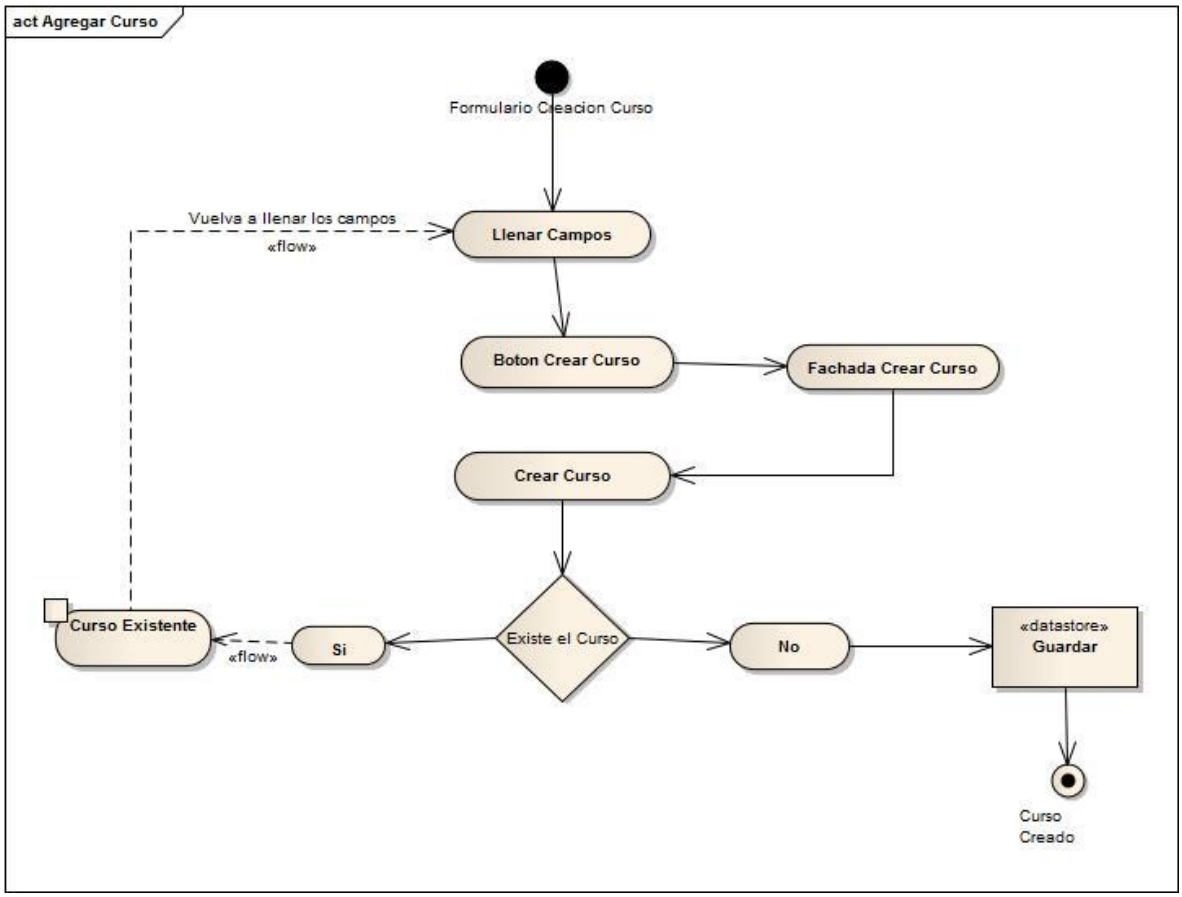


Figura 5. Diagrama de actividad formulario creación curso

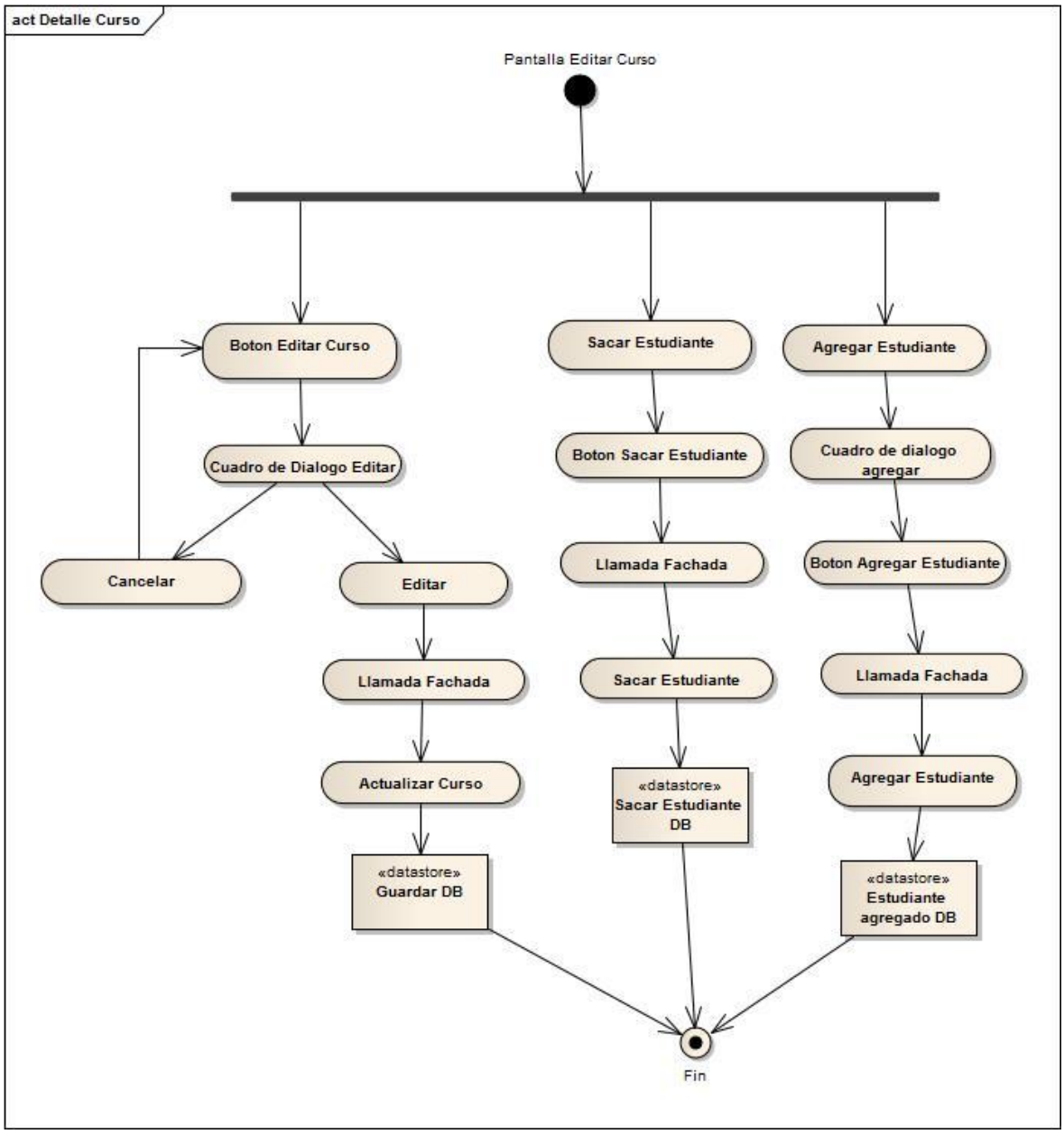


Figura 6. Diagrama de actividad pantalla editar curso.

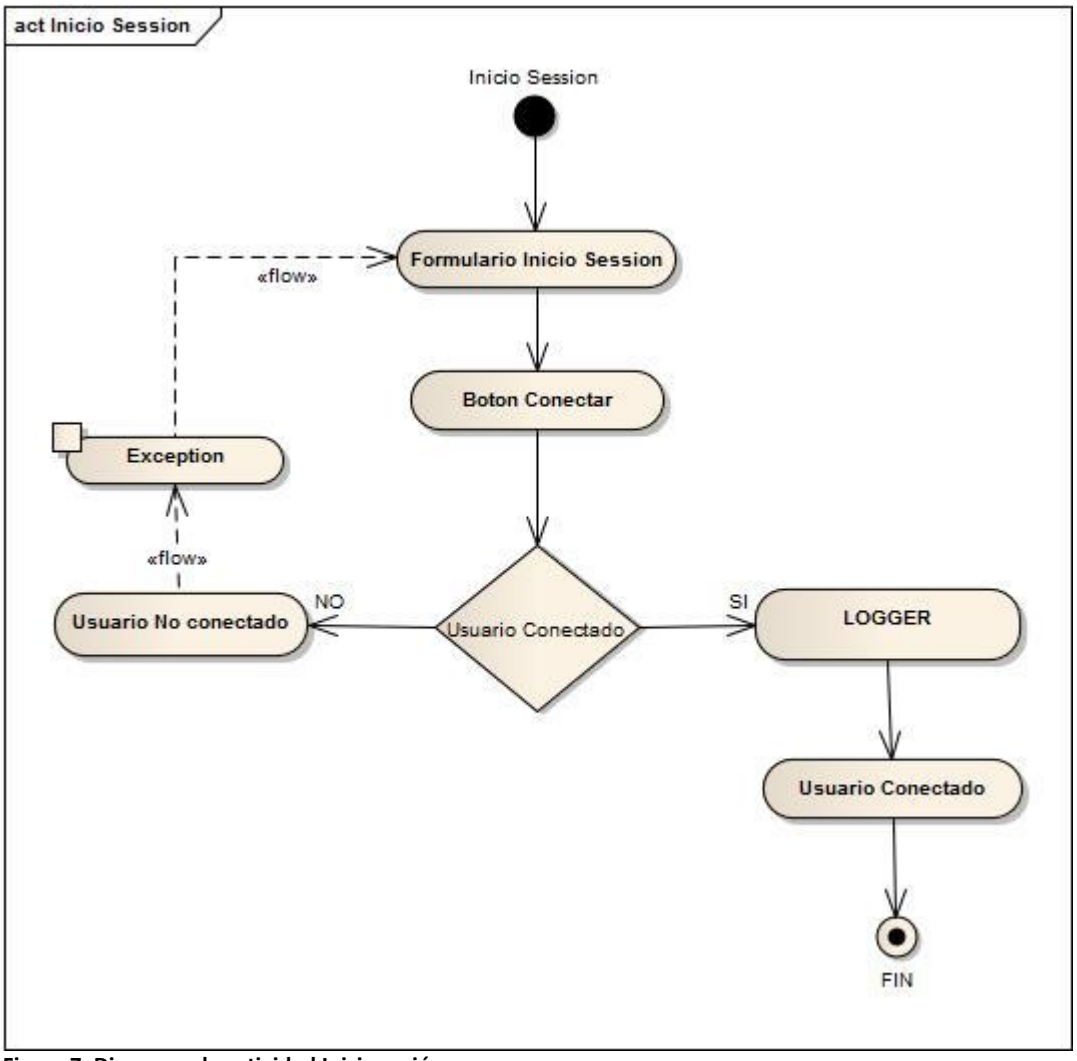


Figura 7. Diagrama de actividad Inicio sesión.

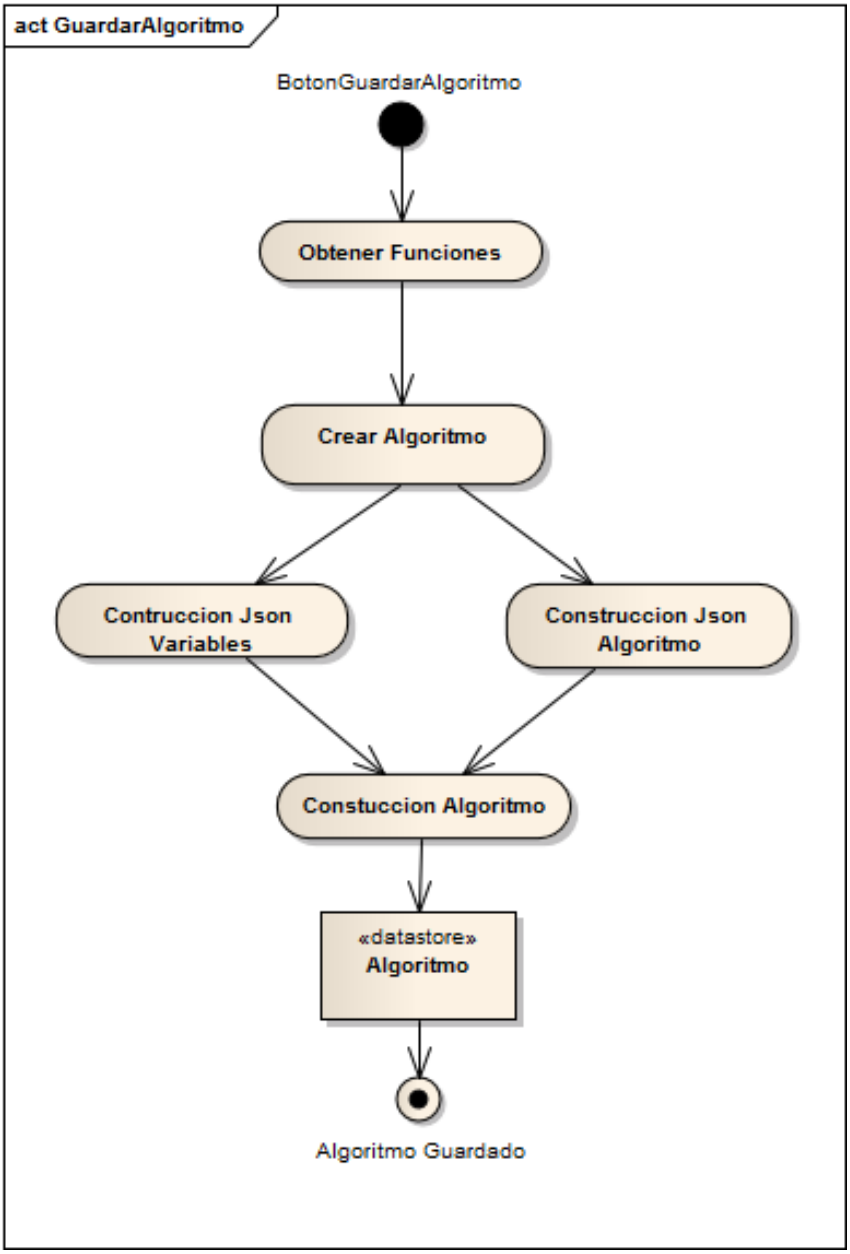


Figura 8. Diagrama de actividad guardar algoritmo.

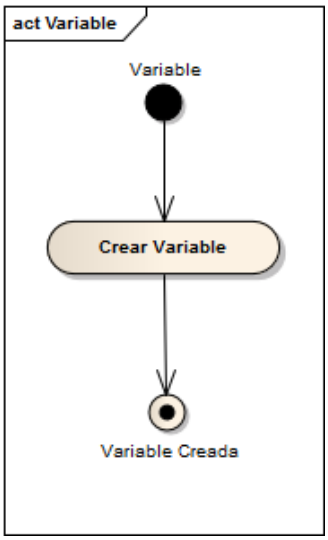


Figura 9. Diagrama de actividad crear variable.

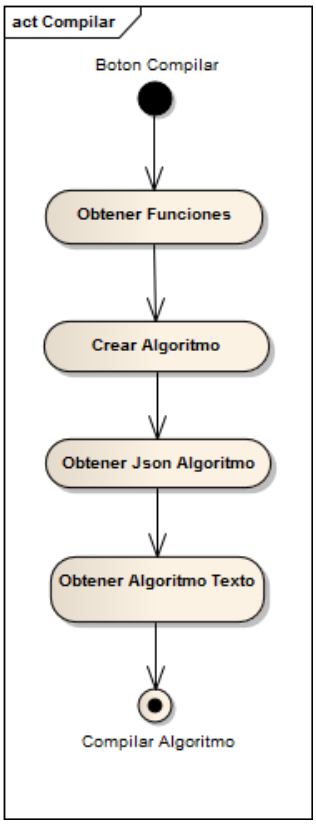


Figura 10. Diagrama de actividad compilar

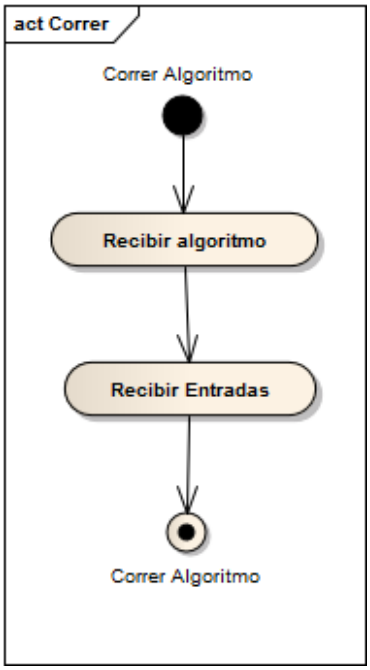


Figura 11. Diagrama de actividad correr algoritmo

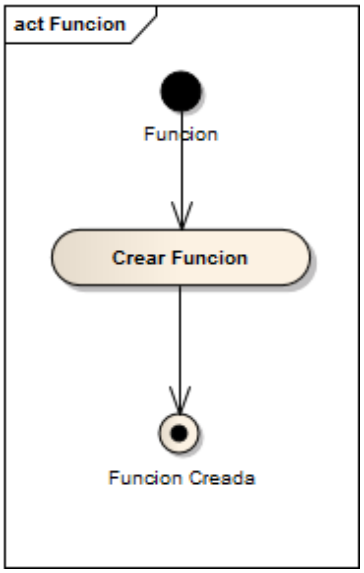


Figura 12. Diagrama de actividad crear funcion

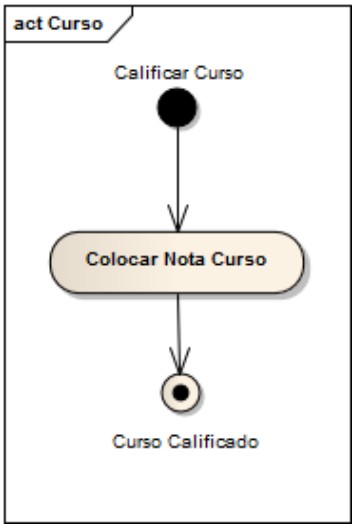


Figura 13. Diagrama Actividad Calificar curso

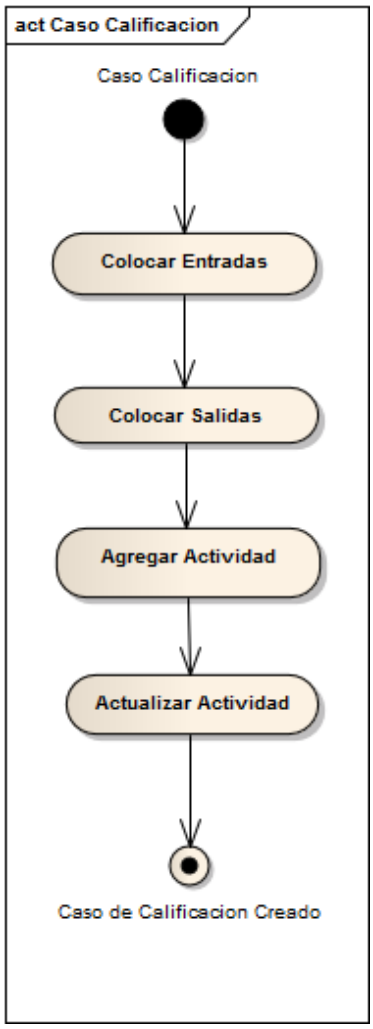


Figura 14. Diagrama de actividad caso calificación.

Anexo 3

Modelo de datos:

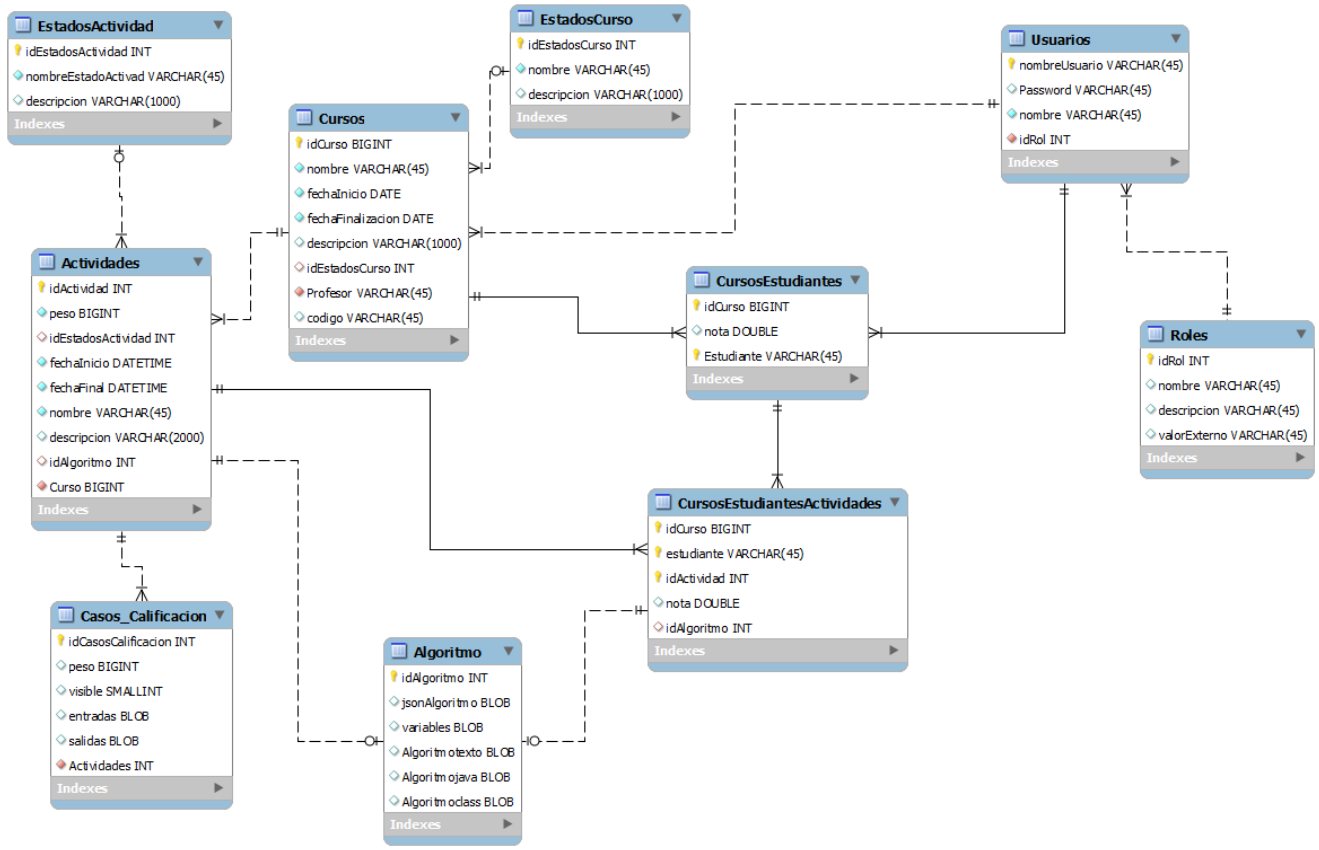


Figura 15. Diagrama modelo de datos.

Anexo 4

Diagramas de Clase

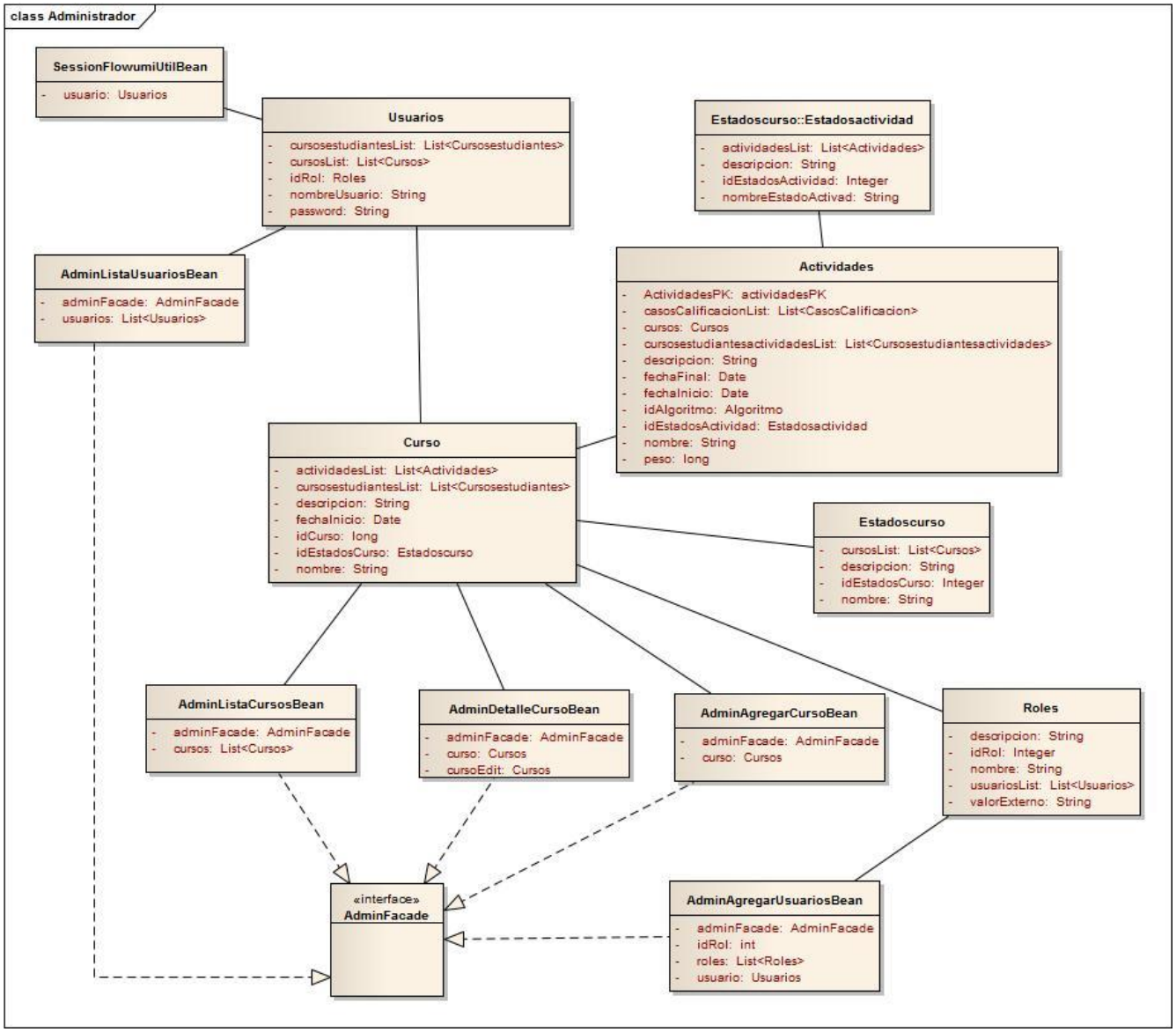


Figura 16. Diagrama de clases administrador

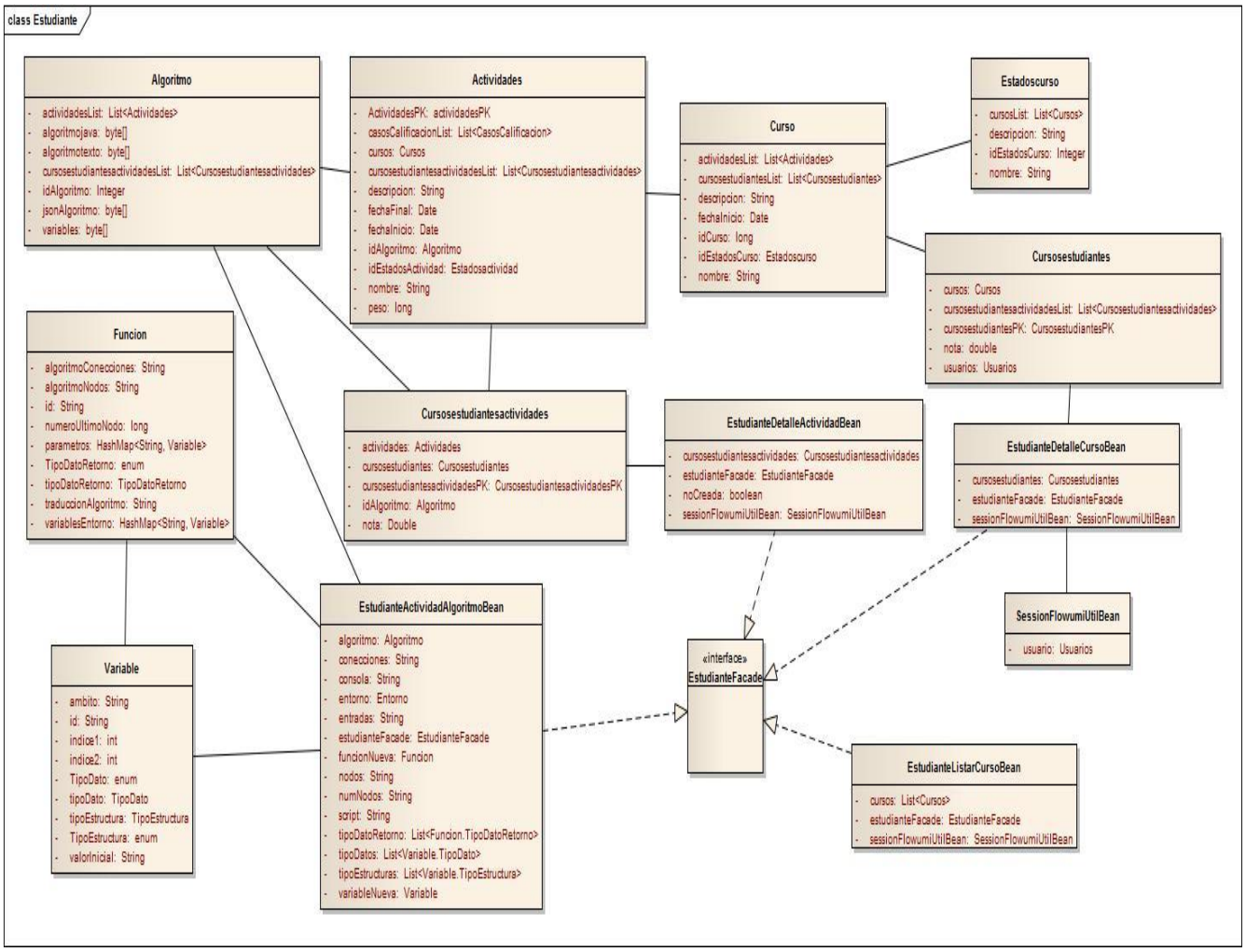


Figura 17. Diagrama de clases Estudiante

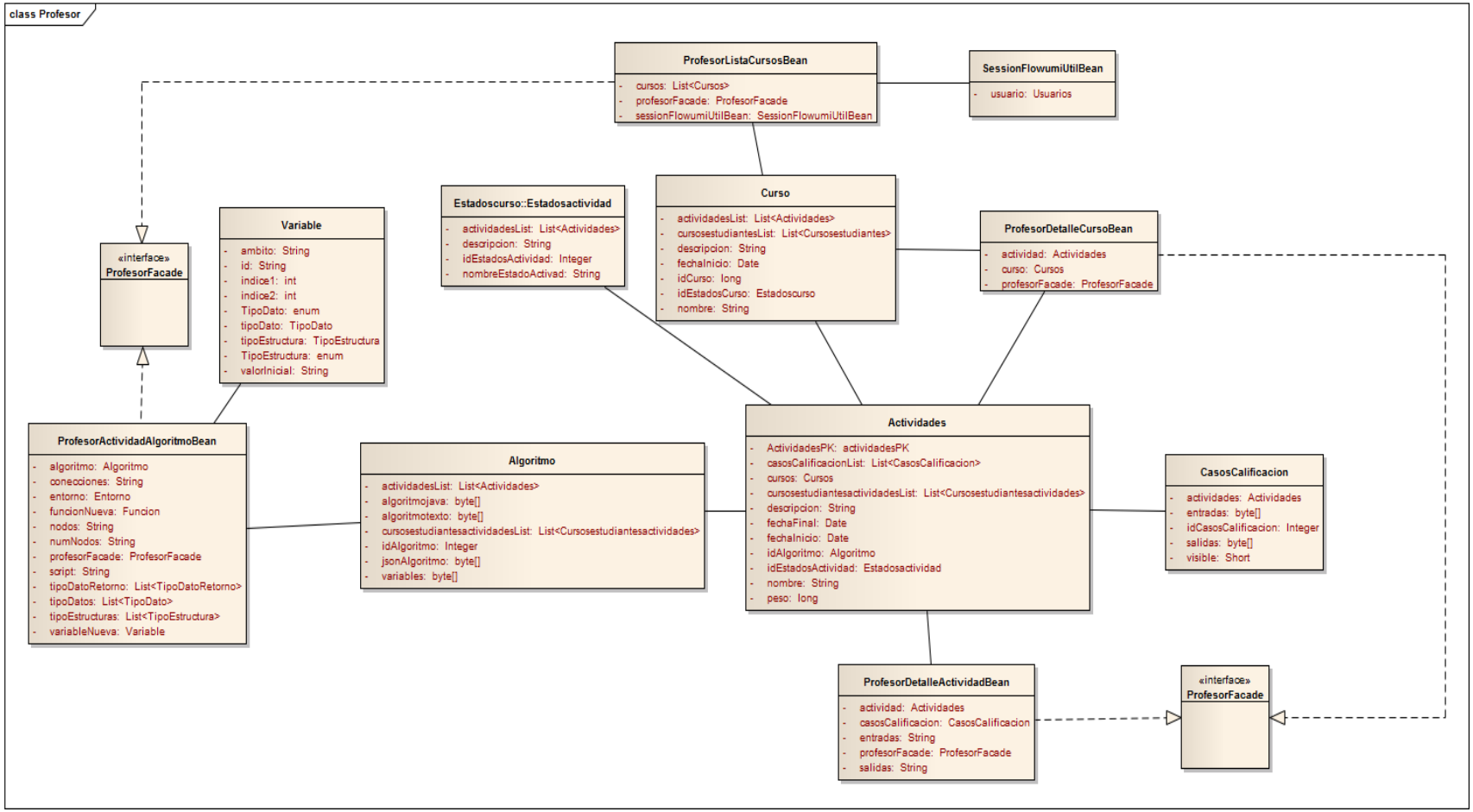


Figura 18. Diagrama de clase profesor

Anexo 5

MANUAL DE USUARIO FLOWUMI SYSTEM

Flowumi System es una herramienta de apoyo para aprender a programar. Creando algoritmos en base a diagramas de flujo.

INTRODUCCIÓN

Este manual le enseñara a utilizar las funcionalidades básicas de la aplicación Flowumi System.

¿Cómo ingresar a Flowumi System?

Acceda a http://sistemas.unitecnologica.edu.co:8080/FlowUmiSystem-web/. Esta vista es de inicio de sesión.

Cuando introduce su usuario y contraseña se direccionara a una vista dependiendo de su rol en el sistema. Los roles son tres: Administrador, Profesor y Estudiante.

A continuación se explicará las funcionalidades dependiendo del Rol.

ADMINISTRADOR

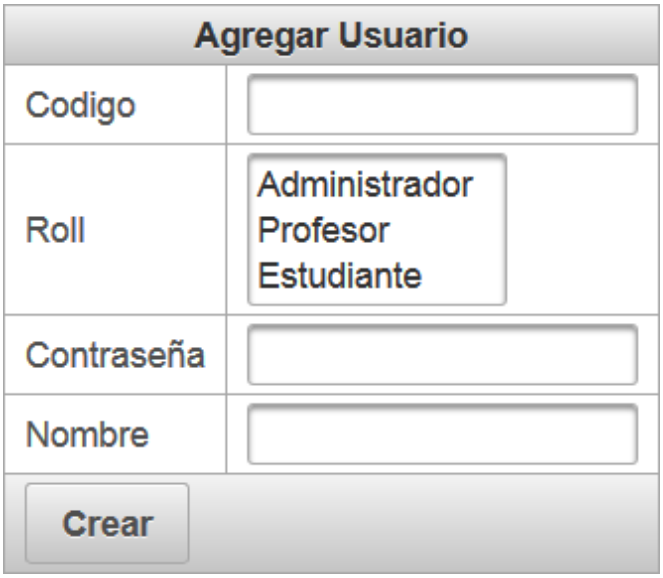
Al entrar como administrador se direccionara a una pantalla de inicio de administrador, la cual contiene solo datos institucionales y de información. En la parte superior está el menú al cual podrá ir a cuatro páginas distintas estas son:

- Lista de usuarios
- Agregar usuario
- Crear curso
- Lista de cursos

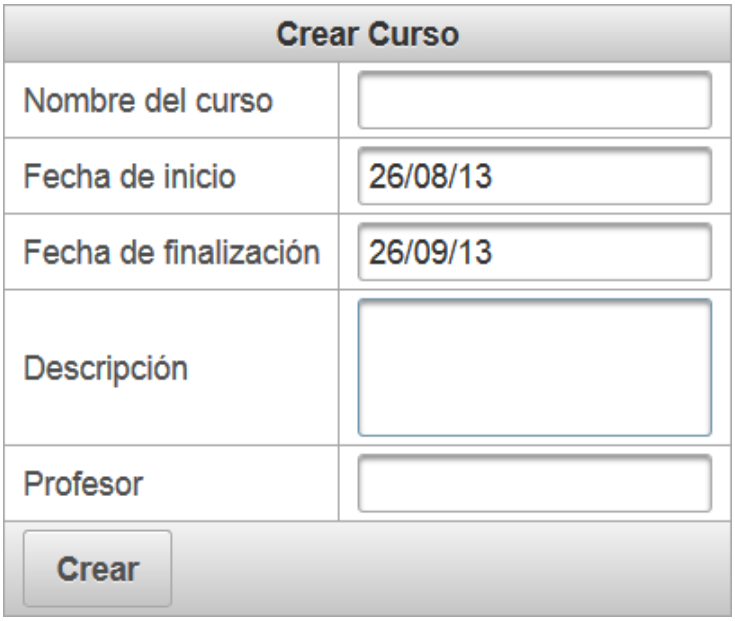
Lista de usuarios: En la pantalla de lista de usuarios aparece una tabla donde nos muestra el código o nombre de usuario, su nombre y su rol.

Codigo	Nombre	Roll
admin	admin	Administrador
est1	est1	Estudiante
est2	est2	Estudiante
est3	est3	Estudiante
T00000001	Profesor	Profesor
T00020609	Ivan	Profesor

Agregar usuario: Esta pantalla tiene un formulario para agregar un nuevo usuario. Los campos son: Código, Rol, Contraseña y Nombre; se llenan los campos y se oprime el botón Crear. Al agregar un nuevo usuario se direcciona a la página de Lista de usuarios.



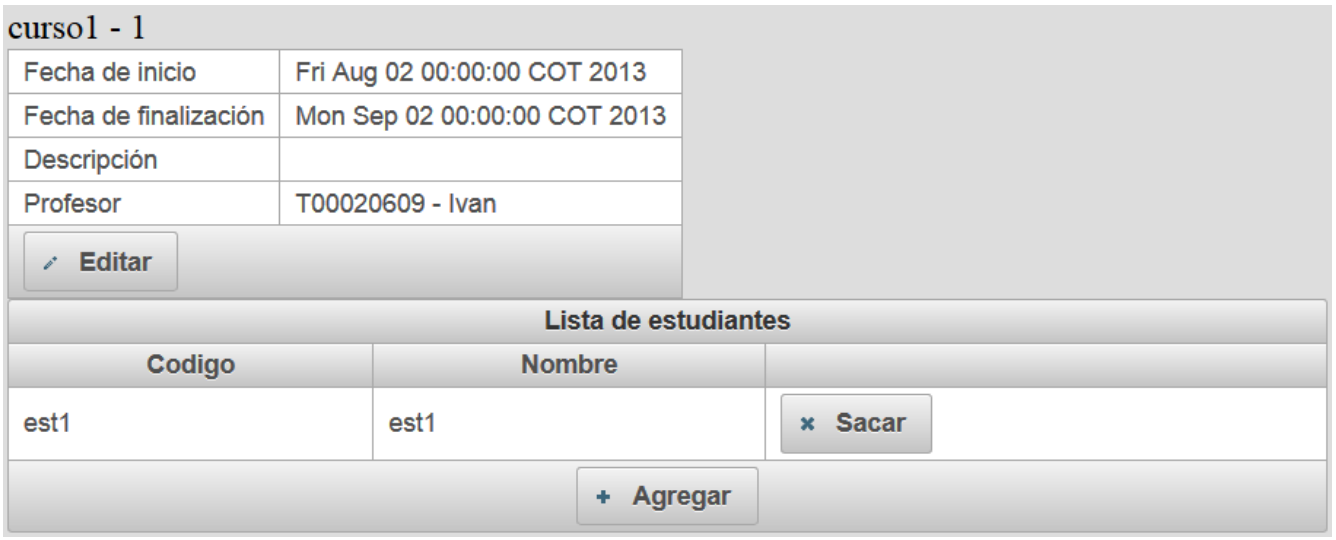
Crear curso: Esta vista tiene un formulario para agregar un nuevo curso. Los campos son: Nombre de curso, Fecha de inicio, Fecha final, Descripción y código de un profesor; se llenan los campos y se oprime el botón Crear. Al agregar un nuevo curso se direcciona a la página de Lista de cursos.



Lista de Cursos: En la pantalla de lista de cursos se muestra una tabla con su nombre, código del curso y código del profesor. Además un botón en cada registro de curso que lo lleva al detalle de cada curso.

Codigo	Nombre del curso	Codigo Profesor
1	curso1	T00020609
2	curso2	T00020609
3	Curso3	T00000001

Detalle de curso: En la vista de detalle se muestra los datos del curso y la lista de estudiantes de este. En esta pantalla se puede editar los datos del curso oprimiendo el botón editar, el cual abre una pantalla de dialogo que será parecida a la pantalla de creación de curso. Al llenar los datos se oprime en editar y se actualizará.



Para agregar un estudiante se oprime el botón agregar y se abrirá una pantalla de dialogo con la lista de estudiante que no están en el curso. Para agregar solo se oprime el botón agregar al lado de cada registro. Cada registro en la tabla de estudiantes de cursos aparece un botón sacar que al presionar saca el estudiante del curso.

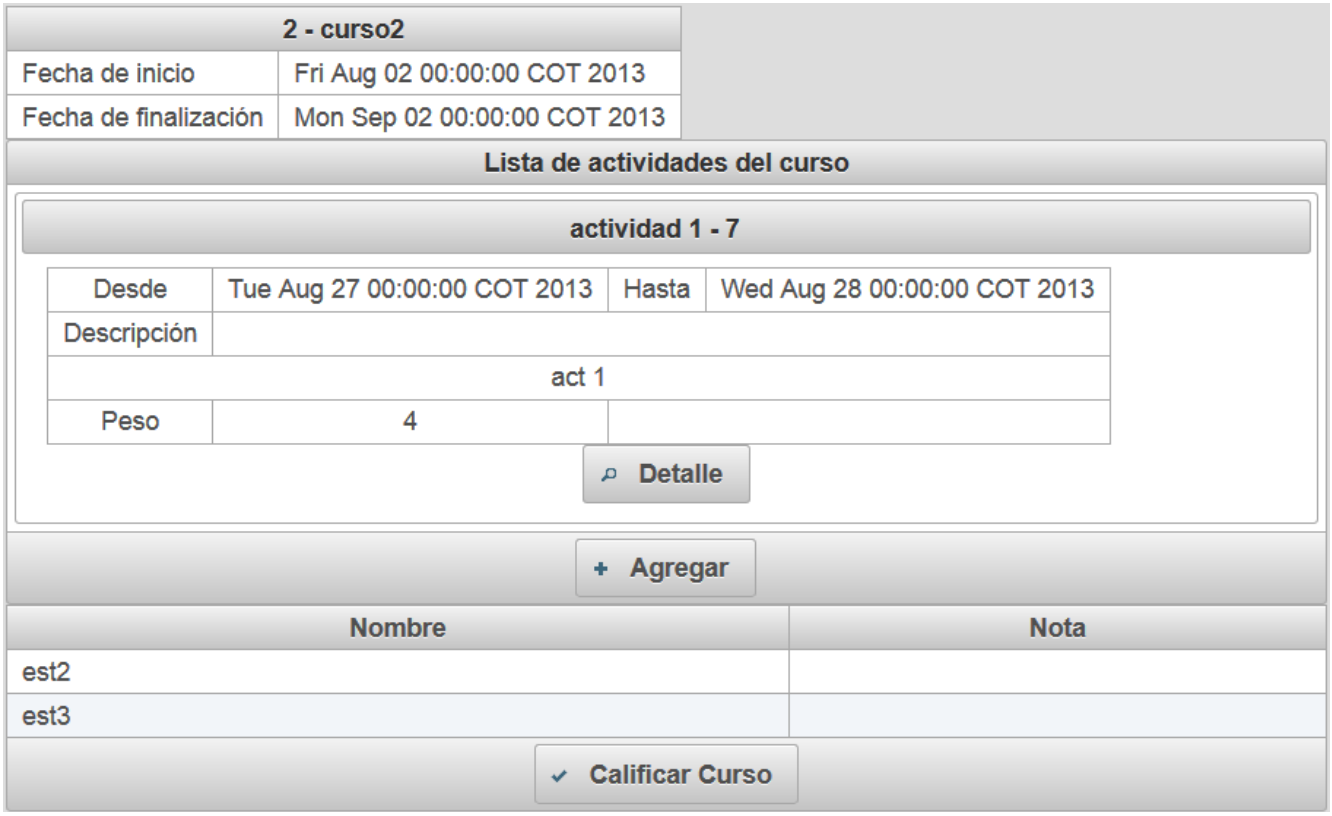
PROFESOR

Al entrar como profesor se direccionara a una vista de inicio de profesor, la cual contiene la lista de cursos. En el menú de profesor solo aparece inicio que direcciona a la página de lista de cursos.

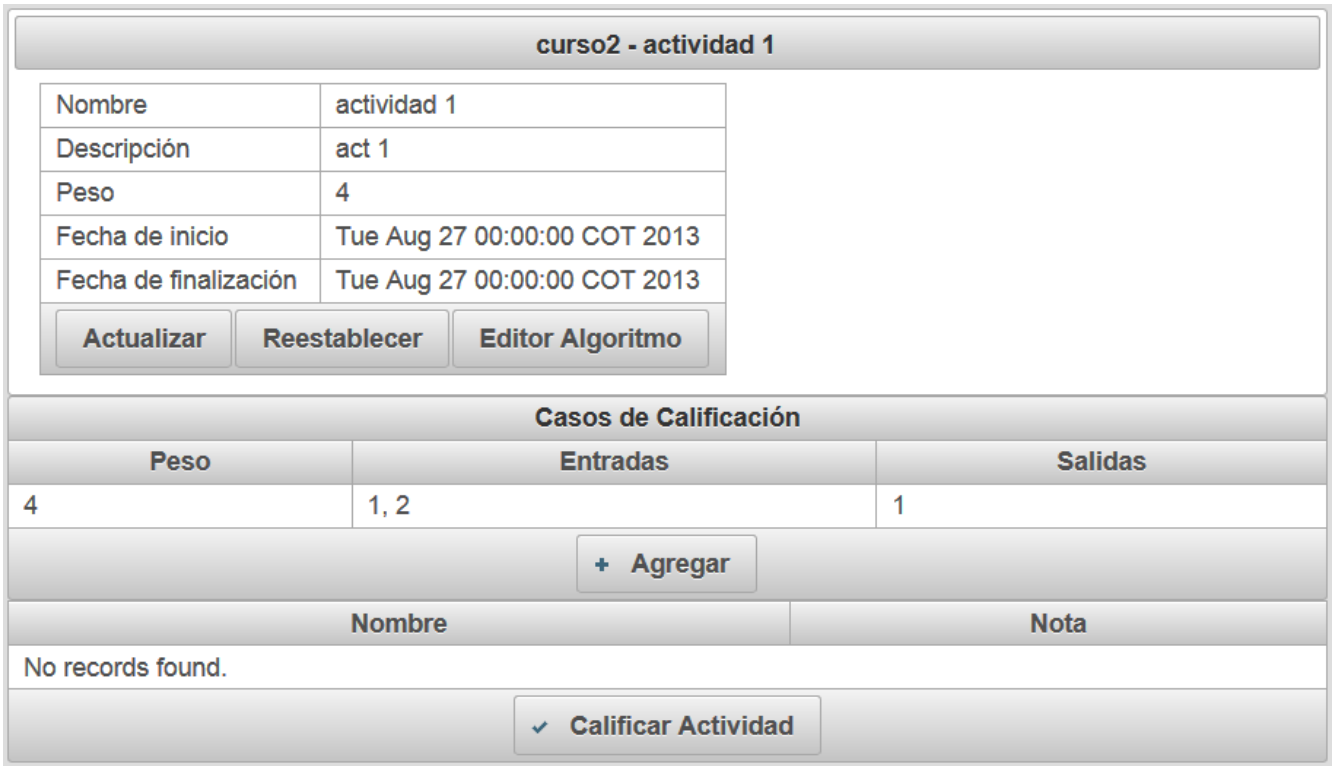
Codigo	Nombre del curso	
1	curso1	
2	curso2	

En cada registro de cursos hay un botón con una lupa que lleva al detalle de dicho curso.

En la pantalla de detalle del curso aparecen los datos del curso y un dashboard de las actividades.



Al final del dashboard Hay un botón agregar que sirve para agregar una nueva actividad. Al presionarlo se abre una pantalla de dialogo con un formulario. Al llenar los campo se presiona el botón confirmar y se agregara la nueva actividad. También aparece la lista de los estudiantes con un botón que dice calificar curso. Al presionarlo el sistema calculara la nota y se la colocara a cada estudiante.



Las actividades tienen un botón detalle que lo direcciona al detalle de la actividad.

La pantalla de actividades para el profesor. Aparece un formulario para actualizar la información general de la actividad. También una tabla con los casos de calificación y un botón para abrir un dialogo y agregar un nuevo caso. Además una tabla con los estudiantes y un botón con el que el sistema calcula la nota de dicha actividad.

En la parte del formulario hay un botón que dice editor algoritmo. Al presionarlo direcciona a la pantalla de edición de algoritmo (esta pantalla se explica más adelante en este manual).

ESTUDIANTE

Al entrar como estudiante se direccionara a una página de inicio de estudiante, la cual contiene la lista de cursos en los que está matriculado el estudiante. En el menú de estudiante aparece inicio que direcciona a la página de lista de cursos.

En cada registro de cursos hay un botón con una lupa que lleva al detalle de dicho curso.

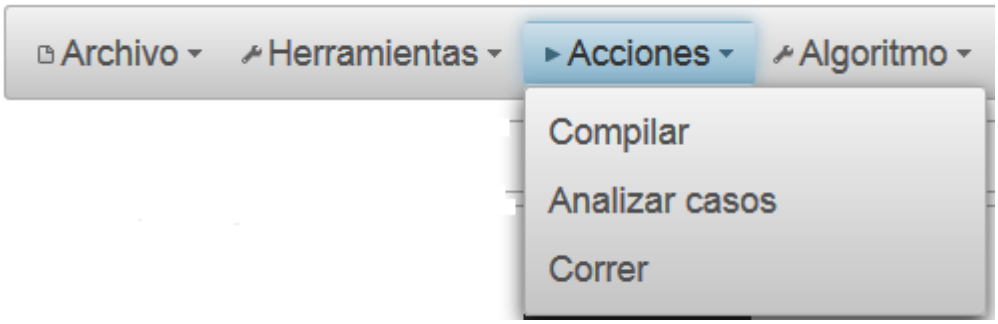
En la pantalla de detalle del curso aparecen los datos del curso y un dashboard de las actividades. Las actividades tienen un botón detalle que lo direcciona al detalle de la actividad.

La pantalla de detalles de la actividad solo muestra los datos de la actividad y un botón que lo direcciona a la página de edición del algoritmo.

VISTA DE EDICION DEL ALGORITMO

La pantalla de edición de algoritmo varía entre la del profesor y estudiante por que la de profesor le faltan unas acciones. Así que se explica la de estudiante en vez.

La pantalla de edición tiene un menú propio en el cual se encuentran las acciones del entorno. Las cuales son:



ARCHIVO: guardar algoritmo, enviar para calificación

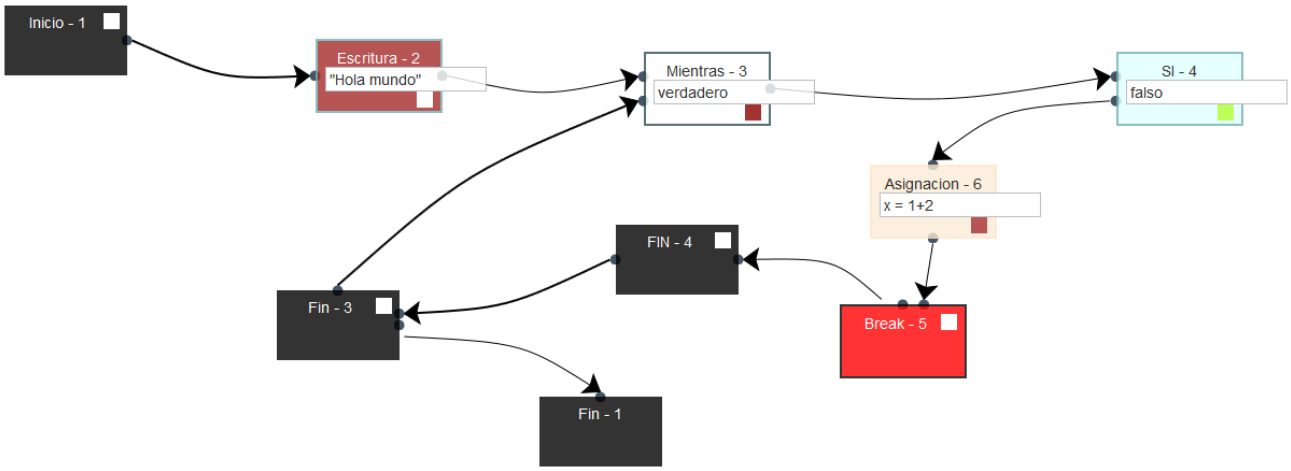
HERRAMIENTA: algoritmo, bloques, consola (estos tres sirve para reabrir las barras de algoritmo, bloques y consola).

ACCIONES: Compilar, -analizar casos y correr.

ALGORITMO: agregar función y agregar variable (cada una abre un dialogo para agregar ya sea función o variable).

La parte central tiene el diagrama de flujo del algoritmo. En el cual aparecerá un bloque al hacer doble clic en la pantalla. El bloque que aparecerá es el que este seleccionado en la barra de bloques. Para borrar se hace clic en la palabra borrar dentro de cada bloque, y para conectar se presiona

dentro del cuadro dentro del bloque y se selecciona otro bloque donde se conectara. Si se quiere eliminar una conexión solo se presiona en la flecha.



En la parte de bloques se listan los bloques para crear los algoritmos. En la parte de algoritmo aparecen las funciones y variables y parámetros de esta, en la cual se podrá borrar al presionar en el botón con la X. Por último la consola donde se mostrara el resultado de compilar y de ejecución.

Anexo 6

CASOS DE USO

GESTIONAR USUARIO

ID	CU-001
NOMBRE	GESTIONAR USUARIOS
ACTORES	ADMINISTRADOR
DESCRIPCION	En este caso de uso se las operaciones CRUD de los usuarios.
REQUERIMIENTOS INCLUIDOS	• Administración de usuarios en el sistema.
PRECIONDICONES	1. Tener los datos del usuario
POSTCONDICIONES	1. Si se creó un usuario, debe quedar el registro nuevo. 2. Si se actualizó un usuario, debe quedar el registro actualizado. 3. Si se eliminó, el registro se debe borrar.
FLUJO NORMAL	1. Se ingresan los datos. 2. Se crea, Actualiza o se elimina el usuario.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	USUARIO NO CREADO, USUARIO NO ACTUALIZADO, USUARIO NO ELIMINADO, USUARIO INEXISTENTE, YA EXISTE REGISTRO DEL USUARIO.
PRIORIDAD	ALTA
FRECUENCIA DE USO	Una vez, al principio de un periodo.

Tabla 2. Caso de uso gestionar Usuario

ID	CU-002
NOMBRE	AUTENTICACIÓN DE USUARIOS DEL SISTEMA
ACTORES	ADMINISTRADOR, PROFESOR, ESTUDIANTE.
DESCRIPCION	Se le permite al usuario el ingreso al sistema.
REQUERIMIENTOS INCLUIDOS	• Autenticación de usuarios del sistema
PRECIONDICONES	1. Nombre de usuario y contraseña
POSTCONDICIONES	1. Usuario autenticado.
FLUJO NORMAL	1. Se ingresan los datos de usuario y contraseña. 2. Se autentica el usuario. 3. Se redireciona a la página dependiendo del rol del usuario.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	NOMBRE DE USUARIO O CONTRASEÑA EQUIVOCADA
PRIORIDAD	ALTA
FRECUENCIA DE USO	Una vez, al principio de un periodo.

Tabla 3. Caso de uso Autenticación usuarios del sistema.

ID	CU-003
NOMBRE	ADMINISTRAR CURSOS
ACTORES	ADMINISTRADOR
DESCRIPCION	En este caso de uso se las operaciones CRUD de los cursos.
REQUERIMIENTOS INCLUIDOS	• Crear curso. • Actualizar curso. • Listar cursos. • Mostrar detalle del curso. • Eliminar curso.
PRECIONDICONES	1. Tener los datos del curso
POSTCONDICIONES	1. Si se creó un curso, debe quedar el registro nuevo. 2. Si se actualizó un curso, debe quedar el registro actualizado. 3. Si se eliminó, el registro se debe borrar.
FLUJO NORMAL	1. Se ingresan los datos. 2. Se crea, Actualiza o se elimina el curso.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	CURSO NO CREADO, CURSO NO ACTUALIZADO, CURSO NO ELIMINADO, CURSO INEXISTENTE.
PRIORIDAD	ALTA
FRECUENCIA DE USO	Una vez, al principio de un periodo.

Tabla 4. Caso de uso Administrar Curso.

ID	CU-004
NOMBRE	ADMINISTRAR ESTUDIANTES EN UN CURSO
ACTORES	ADMINISTRADOR, SISTEMA
DESCRIPCION	Se ejecutan acciones sobre los cursos para la administración de los estudiantes
REQUERIMIENTOS INCLUIDOS	• Agregar estudiantes. • Sacar estudiante del curso. • Listar estudiantes curso.
PRECIONDICONES	1. que existan estudiantes
POSTCONDICIONES	1. La tabla cursoestudiante actualizada.
FLUJO NORMAL	1. El administrador escoge los estudiantes que entran y salen de un curso. 2. Se actualiza la tabla con los nuevos registros.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	NO TIENE
PRIORIDAD	ALTA
FRECUENCIA DE USO	Una vez, al principio del un periodo.

Tabla 5. Casos de uso Administrar Estudiante en un curso.

ID	CU-005
NOMBRE	GESTIONAR VARIABLES DEL CURSO
ACTORES	PROFESOR
DESCRIPCION	En este caso de uso se las operaciones CRUD y autenticación de los usuarios.
REQUERIMIENTOS INCLUIDOS	• Obtener nota final del curso. • Pantalla de estadísticas.
PRECIONDICONES	1. Que el curso haya terminado
POSTCONDICIONES	1. El curso actualice su nota por estudiante.
FLUJO NORMAL	3. Se le pide al sistema obtener las notas 4. se actualizan los registros de notas en la tabla cursoestudiante.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	NO TIENE
PRIORIDAD	MEDIA
FRECUENCIA DE USO	Cuando el curso se finalice.

Tabla 6. Gestionar Variables del curso

ID	CU-006
NOMBRE	ADMINISTRAR ACTIVIDADES
ACTORES	PROFESOR
DESCRIPCION	En este caso de uso se las operaciones CRUD de las actividades.
REQUERIMIENTOS INCLUIDOS	• Crear actividad. • Editar actividad. • Obtener nota de actividad. • Listar actividades de un curso.
PRECIONDICONES	1. Tener los datos de la actividad.
POSTCONDICIONES	1. Si se creó una actividad, debe quedar el registro nuevo. 2. Si se actualizó una actividad, debe quedar el registro actualizado. 3. Si se eliminó, el registro se debe borrar.
FLUJO NORMAL	1. Se ingresan los datos. 2. Se crea, Actualiza o se elimina la actividad.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	ACTIVIDAD NO CREADA, ACTIVIDAD NO ACTUALIZADA, ACTIVIDAD NO ELIMINADA, ACTIVIDAD INEXISTENTE.
PRIORIDAD	ALTA
FRECUENCIA DE USO	Durante el ciclo de vida de un curso.

Tabla 7. Administrar Actividades.

ID	CU-007
NOMBRE	GESTIONAR VARIABLES DE LA ACTIVIDAD
ACTORES	PROFESOR, SISTEMA
DESCRIPCION	En este caso de uso se las operaciones sobre las actividades que no tienen que ver con su administración
REQUERIMIENTOS INCLUIDOS	• Calificar actividad. • Agregar caso de calificación
PRECIONDICONES	2. Para calificar la actividad, Que esta haya finalizado
POSTCONDICIONES	4. Si se creó un caso de calificación, que exista su registro. 5. Si se calificó la actividad, que su nota se haya actualizado.
FLUJO NORMAL	Para un nuevo caso de calificación: 3. Se ingresan los datos de un nuevo caso de calificación. 4. Se crea su registro. Para calificar la actividad: 1. se le pide al sistema que califique. 2. Este obtiene la nota por medio de los casos de calificación
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	NO TIENE
PRIORIDAD	MEDIA
FRECUENCIA DE USO	Por cada actividad creada.

Tabla 8. Gestionar Variables de la Actividad.

ID	CU-008
NOMBRE	GESTIONAR DRIAGRAMA DE FLUJO DEL ALGORITMO
ACTORES	PROFESOR, ESTUDIANTE
DESCRIPCION	Este casi de uso explica la pantalla de creación de un algoritmo.
REQUERIMIENTOS INCLUIDOS	• Agregar bloque. • Eliminar bloque. • Editar bloque. • Agregar variable. • Editar variable. • Eliminar variable.
PRECIONDICONES	NO TIENE
POSTCONDICIONES	NO TIENE
FLUJO NORMAL	1. El usuario crea su algoritmo mediante los bloques y creación de funciones y variables.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	NO TIENE
PRIORIDAD	MUY ALTA
FRECUENCIA DE USO	Por cada actividad una vez el profesor y una vez cada estudiante de el curso.

Tabla 9. Gestionar Diagrama de flujo del algoritmo.

ID	CU-009
NOMBRE	ADMINISTRAR ACCIONES SOBRE EL ALGORITMO
ACTORES	ESTUDIANTE
DESCRIPCION	Este caso de uso explica las acciones sobre un algoritmo.
REQUERIMIENTOS INCLUIDOS	• Ejecutar algoritmo. • Guardar algoritmo. • Probar los casos de calificación del algoritmo.
PRECIONDICONES	1. Terminar el algoritmo.
POSTCONDICIONES	1. Resultado del algoritmo al ejecutarlo.
FLUJO NORMAL	5. Se realiza el algoritmo. 6. se ejecuta una acción. 7. Se obtiene el resultado.
FLUJO ALTERNAIVO	NO TIENE
EXEPCIONES	EXCEPCION ALGORITMO ERROR.
PRIORIDAD	ALTA
FRECUENCIA DE USO	Siempre

Tabla 10. Administrar Acciones sobre el Algoritmo.