

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



GRADO EN INGENIERÍA INFORMÁTICA
TRABAJO FIN DE GRADO

APRENDIZAJE DIRIGIDO A LA PROGRAMACIÓN EN C
MEDIANTE DETECCIÓN AUTOMÁTICA DE ERRORES

Autor: Daniel Martínez Navarro

Tutor: Valentín Moreno Pelayo

Octubre 2015

Título: APRENDIZAJE DIRIGIDO A LA PROGRAMACIÓN EN C MEDIANTE DETECCIÓN AUTOMÁTICA DE ERRORES

Autor: Daniel Martínez Navarro

Tutor: Valentín Moreno Pelayo

EL TRIBUNAL

Presidente: Marco Romano

Secretario: María Paz Sesmero Lorente

Vocal: Enrique San Millán Heredia

Suplente: Ricardo Aler Mur

Realizado el acto de defensa del Trabajo Fin de Grado el día 13 de Octubre de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Presidente

Secretario

Vocal

Agradecimientos

Siendo como soy alguien bastante parco en palabras, y siendo este un punto de la memoria con una importancia bastante escasa, no me extenderé demasiado en mis palabras.

Primeramente acordarme del tutor de mi proyecto Valentín. Cuando le pedí que me presentara los proyectos que tenía, me dio a elegir entre una lista de varios, y al elegir desarrollar esta aplicación me comentó que era una gran elección y que siendo mi carrera Grado en Ingeniería Informática, en 7 o 10 días lo tendría casi acabado. Mi gozo en un pozo. En todo lo demás un tutor excepcional, estando accesible en cuanto se le necesitaba, y marcando una pauta pero dejando hacer a uno mismo bastante durante el desarrollo del proyecto.

Por lo que respecta a la gente de la universidad, podría acordarme de mucha gente a la que he conocido durante todos estos años, aunque en realidad únicamente me sale acordarme de mi hermano Juanra, y de mis padres Juan e Isabel.

Acordarme de mis padres por las muchas veces que creo que les he decepcionado, no dando todo lo que podía dar. Por todas aquellas veces en las que he tropezado y me han ayudado a levantarme.

Y a mi hermano porque si no fuera por él sé que no estaría ni de lejos cerca de donde estoy hoy. Apoyándome en cada momento, y aguantándome aunque en ciertos momentos sea bastante insoportable.

Lo único que puedo decir es que gracias a los tres, y una cosa más ¡Hala Madrid!

Resumen

La tecnología está cambiando muchos aspectos de nuestra vida cotidiana: nuestra forma de relacionarnos con otras personas, nuestra manera de trabajar o, incluso, nuestra manera de aprender. El uso de la tecnología en las aulas no nos es extraño; sin embargo, la tecnología se utiliza como una especie de "instrumento pasivo" que convierte al alumno en un usuario de distintas aplicaciones y servicios [\[1\]](#).

La interacción personalizada entre profesorado y alumnado va a ser imprescindible, al depender de ésta el posterior éxito o fracaso del alumno. El problema es que fuera del entorno académico esa interacción no existe, o es difícil, si no imposible, que se produzca.

La realidad es que en los primeros instantes de acercamiento a la programación, el programador novel cuenta con la ayuda del compilador para corregir los errores de sintaxis de su código, pero a menos que tenga a un programador experto cerca, nadie le ayudará a corregir los errores de estilo. Queda patente que los compiladores actuales son insuficientes para los programadores inexpertos, de hecho, hay trabajos que lo corroboran y desarrollan herramientas específicas para suplir estas deficiencias.

El objetivo de este Trabajo Fin de Grado es desarrollar una aplicación que permita tanto al alumno como al profesor, determinar si un programa escrito en lenguaje de programación C cumple con una serie de normas de estilo. Algo que no sólo ayudará al alumno a mejorar el desarrollo correcto de sus programas, si no que permitirá al profesor poner más énfasis en unos determinados errores u otros, según el momento del análisis del programa.

Índice General

Agradecimientos	4
Resumen.....	5
Índice General	6
Índice de Tablas.....	8
Índice de Ilustraciones	12
Capítulo 1. Introducción.....	14
1.1 Motivación del proyecto	14
1.2 Objetivos del proyecto	16
1.3 Contenidos de la memoria	16
Capítulo 2. Estado del arte	18
2.1 Introducción a los lenguajes de la programación	18
2.2 Introducción al lenguaje C.....	18
2.3 Ventajas y desventajas del lenguaje C	20
2.4 Proceso de compilación de un programa en lenguaje C.....	20
2.5 Análisis y detección de errores	21
2.5.1 Introducción	21
2.5.2 Tipos de errores	22
2.5.3 Clasificación de los errores.....	23
2.6 Herramientas de compilación	24
2.6.1 Compilador gcc.....	24
2.6.2 Herramienta de depuración GDB.....	25
2.6.3 Herramienta de depuración Valgrind.....	27
2.6.4 Conclusiones finales	28
Capítulo 3. Fase de Análisis, Diseño e Implementación.....	29
3.1 Fase de Análisis	29
3.1.1 Marco regulador.....	29
3.1.2 Definición del sistema	30
3.1.3 Alcance del sistema	30
3.1.4 Restricciones generales.....	30
3.1.5 Entorno de desarrollo	30
3.1.6 Requisitos de la herramienta	31
3.1.7 Especificación de casos de uso.....	48
3.2 Fase de Diseño e Implementación	53

3.2.1	Clase ventana_ppal	54
3.2.2	Base de datos	55
3.2.3	Clase ventana_ayuda	56
3.2.4	Clase ventana_prof	56
3.2.5	Clase ventana_alum	66
Capítulo 4. Resultados Obtenidos		67
Capítulo 5. Planificación		81
5.1	Planificación estimada.....	81
5.2	Planificación Real	83
Capítulo 6. Presupuesto		86
6.1	Recursos Materiales	86
6.2	Recursos Humanos	86
6.3	Presupuesto final.....	86
Capítulo 7. Conclusiones		87
Capítulo 8. Líneas Futuras		88
Anexo. Manual de Usuario		90
a.1	Instalación	90
a.2	Pantallas	92
Referencias		98

Índice de Tablas

Tabla 1. Plantilla de requerimientos.....	31
Tabla 2. REQ_FU_01	32
Tabla 3. REQ_FU_02	32
Tabla 4. REQ_FU_03	32
Tabla 5. REQ_FU_04	33
Tabla 6. REQ_FU_05	33
Tabla 7. REQ_FU_06	33
Tabla 8. REQ_FU_07	33
Tabla 9. REQ_FU_08	34
Tabla 10. REQ_FU_09	34
Tabla 11. REQ_FU_10	34
Tabla 12. REQ_FU_11	34
Tabla 13. REQ_FU_12	35
Tabla 14. REQ_FU_13	35
Tabla 15. REQ_FU_14	35
Tabla 16. REQ_FU_15	35
Tabla 17. REQ_FU_16	36
Tabla 18. REQ_FU_17	36
Tabla 19. REQ_FU_18	36
Tabla 20. REQ_FU_19	36
Tabla 21. REQ_FU_20	37
Tabla 22. REQ_FU_21	37
Tabla 23. REQ_FU_22	37
Tabla 24. REQ_FU_23	37
Tabla 25. REQ_FU_24	38
Tabla 26. REQ_FU_25	38
Tabla 27. REQ_FU_26	38
Tabla 28. REQ_FU_27	39
Tabla 29. REQ_FU_28	39
Tabla 30. REQ_FU_29	39
Tabla 31. REQ_FU_30	39
Tabla 32. REQ_FU_31	40
Tabla 33. REQ_FU_32	40
Tabla 34. REQ_FU_33	40
Tabla 35. REQ_FU_34	40
Tabla 36. REQ_FU_35	41
Tabla 37- REQ_NF_01	41
Tabla 38. REQ_NF_02	41
Tabla 39. REQ_NF_03	42
Tabla 40. REQ_NF_04	42
Tabla 41. REQ_NF_05	42
Tabla 42. REQ_NF_06	42

Tabla 43. REQ_NF_07	43
Tabla 44. REQ_NF_08	43
Tabla 45. REQ_NF_09	43
Tabla 46. REQ_NF_10	43
Tabla 47. REQ_NF_11	44
Tabla 48. REQ_NF_12	44
Tabla 49. REQ_NF_13	44
Tabla 50. REQ_NF_14	44
Tabla 51. REQ_NF_15	45
Tabla 52. REQ_NF_16	45
Tabla 53. REQ_NF_17	45
Tabla 54. REQ_NF_18	45
Tabla 55. REQ_NF_19	46
Tabla 56. REQ_NF_20	46
Tabla 57. REQ_NF_21	46
Tabla 58. REQ_NF_22	46
Tabla 59. REQ_NF_23	47
Tabla 60. REQ_NF_24	47
Tabla 61. REQ_NF_25	47
Tabla 62. REQ_NF_26	47
Tabla 63. REQ_NF_27	48
Tabla 64. REQ_NF_28	48
Tabla 65. REQ_NF_29	48
Tabla 66. Plantilla de casos de uso.....	50
Tabla 67. CdU_01	50
Tabla 68. CdU_02	50
Tabla 69. CdU_03	51
Tabla 70. CdU_04	51
Tabla 71. CdU_05	51
Tabla 72. CdU_06	51
Tabla 73. CdU_07	52
Tabla 74. CdU_08	52
Tabla 75. CdU_09	52
Tabla 76. CdU_10	53
Tabla 77. CdU_11	53
Tabla 78. Plantilla de Casos de Prueba	67
Tabla 79. CdP_01	67
Tabla 80. CdP_02	68
Tabla 81. CdP_03	68
Tabla 82. CdP_04	68
Tabla 83. CdP_05	69
Tabla 84. CdP_06	69
Tabla 85. CdP_07	69
Tabla 86. CdP_08	70
Tabla 87. CdP_09	70

Tabla 88. CdP_10	70
Tabla 89. CdP_11	70
Tabla 90. CdP_12	71
Tabla 91. CdP_13	71
Tabla 92. CdP_14	71
Tabla 93. CdP_15	71
Tabla 94. CdP_16	71
Tabla 95. CdP_17	72
Tabla 96. CdP_18	72
Tabla 97. CdP_19	72
Tabla 98. CdP_20	72
Tabla 99. CdP_21	73
Tabla 100. CdP_22	73
Tabla 101. CdP_23	73
Tabla 102. CdP_24	73
Tabla 103. CdP_25	74
Tabla 104. CdP_26	74
Tabla 105. CdP_27	74
Tabla 106. CdP_28	74
Tabla 107. CdP_29	75
Tabla 108. CdP_30	75
Tabla 109. CdP_31	75
Tabla 110. CdP_32	75
Tabla 111. CdP_33	76
Tabla 112. CdP_34	76
Tabla 113. CdP_35	76
Tabla 114. CdP_36	76
Tabla 115. CdP_37	77
Tabla 116. CdP_38	77
Tabla 117. CdP_39	77
Tabla 118. CdP_40	77
Tabla 119. CdP_41	78
Tabla 120. CdP_42	78
Tabla 121. CdP_43	78
Tabla 122. CdP_44	78
Tabla 123. CdP_45	78
Tabla 124. CdP_46	79
Tabla 125. CdP_47	79
Tabla 126. CdP_48	79
Tabla 127. CdP_49	79
Tabla 128. CdP_50	80
Tabla 129. CdP_51	80
Tabla 130. Desarrollo del trabajo.....	83
Tabla 131. Presupuesto recursos Hardware	86
Tabla 132. Presupuesto recursos Software	86

Tabla 133. Presupuesto recursos humanos	86
Tabla 134. Presupuesto final	86

Índice de Ilustraciones

Ilustración 1. Porcentaje aprobados-evaluados	14
Ilustración 2. Datos representativos	15
Ilustración 3. Etapas de compilación	21
Ilustración 4. Clasificación de errores según efectos generados	23
Ilustración 5. Clasificación de errores según el momento en que se producen	23
Ilustración 6. Clasificación de errores según su naturaleza	23
Ilustración 7. Clasificación de errores según su tratamiento	24
Ilustración 8. Clasificación de errores según su importancia	24
Ilustración 9. Ejecución sencilla del gcc	24
Ilustración 10. Ejecución gcc con error	25
Ilustración 11. Ejecución gcc con warning	25
Ilustración 12. Ejecución sencilla del GDB	26
Ilustración 13. Ejecución normal del GDB.....	26
Ilustración 14. Ejecución de Valgrind.....	27
Ilustración 15. Diagrama de casos de uso para rol usuario.....	49
Ilustración 16. Diagrama de casos de Uso para rol Alumno	49
Ilustración 17. Diagrama de casos de Uso para rol Profesor	49
Ilustración 18. Clase ventana_ppal	54
Ilustración 19. Método aceptar_ppal_Click.....	54
Ilustración 20. Método comprobar_con_click.....	55
Ilustración 21. Método salir_ppal_Click	55
Ilustración 22. Tablas de ejemplo	55
Ilustración 23. Creación base de datos y tablas	56
Ilustración 24. Clase ventana_ayuda	56
Ilustración 25. Método aceptar_ayuda_click	56
Ilustración 26. Clase ventana_prof	57
Ilustración 27. Método salir_prof_click.....	58
Ilustración 28. Método cerrar_sesión_click.....	58
Ilustración 29. Método botón_datos_click.....	58
Ilustración 30. Método botón_resul_click.....	58
Ilustración 31. Método caja_líneas_keypress	59
Ilustración 32. Método ayuda_click.....	59
Ilustración 33. Método ventana_prof_load	59
Ilustración 34. Método guardar_config.....	60
Ilustración 35. Método comprobar_formato	60
Ilustración 36. Método comprobar_indentación	61
Ilustración 37. Método comprobar_comentarios	61
Ilustración 38. Método comprobar_saltos	61
Ilustración 39. Método comprobar_líneas_prog.....	62
Ilustración 40. Método comprobar_proc_fun.....	62
Ilustración 41. Método comprobar_proc_fun_2.....	63
Ilustración 42. Método comprobar_inter2.....	63

Ilustración 43. Método comprobar_inter3	64
Ilustración 44. Método comprobar_var_ini	64
Ilustración 45. Método comprobar_var_asig	64
Ilustración 46. Método comprobar_var_uti	65
Ilustración 47. Método comprobar_var_glob	65
Ilustración 48. Método An_prof_Click	65
Ilustración 49. Clase ventana_alum	66
Ilustración 50. Tabla Diagrama de Gantt – Planificación Estimada	81
Ilustración 51. Diagrama de Gantt – Planificación Estimada	82
Ilustración 52. Tabla Diagrama de Gantt – Planificación Real	83
Ilustración 53. Diagrama de Gantt – Planificación Real	84
Ilustración 54. Conectar Servidor MySQL	90
Ilustración 55. Mensaje MySQL	90
Ilustración 56. Creación base de datos y tablas	91
Ilustración 57. Ejecutable de la aplicación	91
Ilustración 58. Ventana principal	92
Ilustración 59. Ventana principal profesor	92
Ilustración 60. Ventana Principal alumno	93
Ilustración 61. Ventana principal botones	94
Ilustración 62. Ventana de ayuda	94
Ilustración 63. Ventana Ejemplo configuración	95
Ilustración 64. Carpeta configuración	95
Ilustración 65. Ejemplo config.txt	96
Ilustración 66. Ventana alumno configuración	96
Ilustración 67. Archivo solución	97

Capítulo 1. Introducción

En este apartado se representan los propósitos de este trabajo. La motivación, los objetivos y los contenidos de la memoria son los puntos esenciales de este apartado.

1.1 Motivación del proyecto

Reza un dicho popular, que el hombre es el único animal capaz de tropezar dos (e incluso más veces) con la misma piedra. Somos la única especie que caemos varias veces en los mismos errores, nos tropezamos ante los mismos obstáculos, y caemos en la misma trampa varias veces. Por supuesto la informática no es una excepción, y siempre acabamos cometiendo los mismos fallos, y más todavía si no tenemos ningún indicador claro que nos señale donde cometemos el error.

El único capaz de indicarle al alumno que fallos comete, y como pueden ser los mismos subsanados es el profesor, pero el tiempo que este puede dedicar a cada alumno suele ser bastante escaso. Esto puede provocar que los errores de programación vayan acumulándose, conforme la dificultad de los ejercicios se va complicando, alcanzando finalmente un punto de no retorno donde el alumno acaba suspendiendo sin ni tan siquiera alcanzar la mitad del cuatrimestre.

La media general de aprobados de los universitarios españoles es del 61% de los créditos matriculados (se puede decir que los alumnos van a medio curso por año, ya que no se suelen matricular del cursos completo), pero los extremos van desde el 45% en la carrera de Ingeniería Técnica Informática, hasta el 84% de aprobados en la carrera de Enfermería, según los datos de 2008, de la Conferencia de Rectores de las Universidades Españolas ^[2].

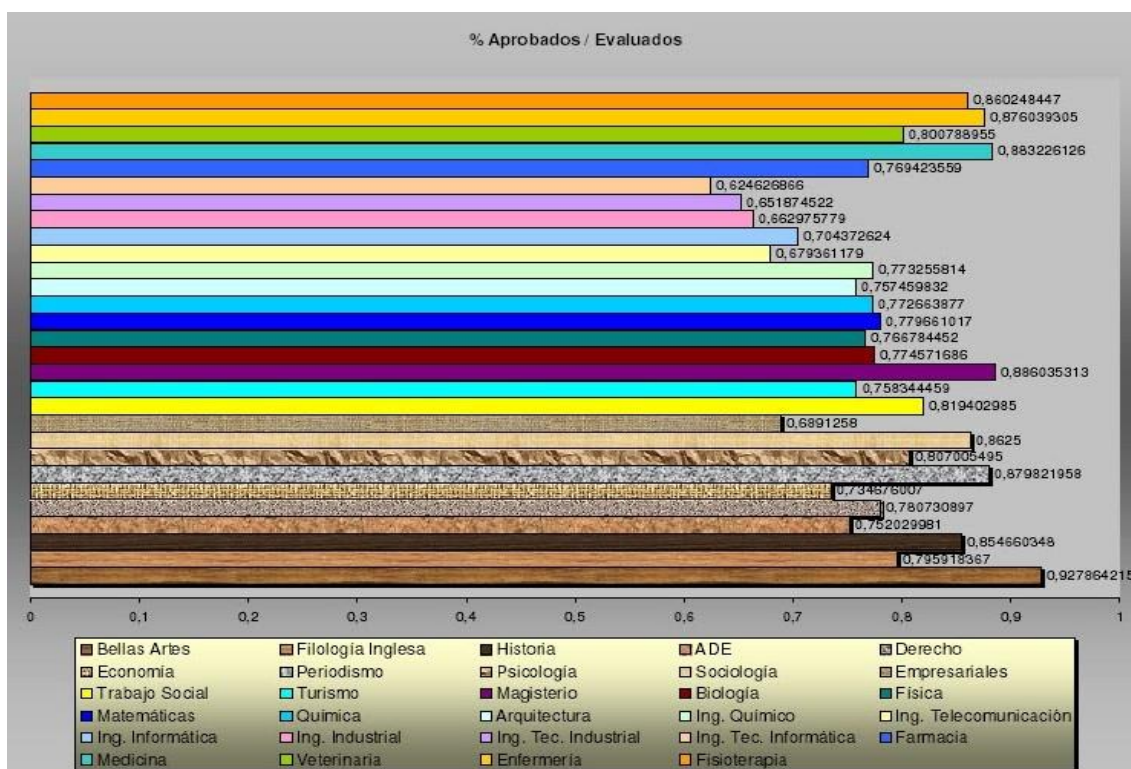


Ilustración 1. Porcentaje aprobados-evaluados

TITULACIONES	Matriculados	Evaluidos	Aprobados	Apr vs Mat	Apr vs Eval
Bellas Artes	837	707	656	78,38%	92,79%
Filología Inglesa	783	588	468	59,77%	79,59%
Historia	882	633	541	61,34%	85,47%
ADE	4157	3202	2408	57,93%	75,20%
Derecho	3883	2709	2115	54,47%	78,07%
Economía	1603	1142	839	52,34%	73,47%
Periodismo	781	674	593	75,93%	87,98%
Psicología	1741	1456	1175	67,49%	80,70%
Sociología	348	240	207	59,48%	86,25%
Empresariales	3362	2345	1616	48,07%	68,91%
Trabajo Social	776	670	549	70,75%	81,94%
Turismo	900	749	568	63,11%	75,83%
Magisterio	2110	1869	1656	78,48%	88,60%
Biología	1429	1109	859	60,11%	77,46%
Física	426	283	217	50,94%	76,68%
Matemáticas	331	236	184	55,59%	77,97%
Química	962	717	554	57,59%	77,27%
Arquitectura	1546	1307	990	64,04%	75,75%
Ing. Química	633	516	399	63,03%	77,33%
Ing. Telecomunicaciones	981	814	553	56,37%	67,94%
Ing. Informática	1304	1052	741	56,83%	70,44%
Ing. Industrial	1584	1445	958	60,48%	66,30%
Ing. Tec. Industrial	1706	1307	852	49,94%	65,19%
Ing. Tec. Informática	1842	1340	837	45,44%	62,46%
Farmacia	1072	798	614	57,28%	76,94%
Medicina	2248	2021	1785	79,40%	88,32%
Veterinaria	625	507	406	64,96%	80,08%
Enfermería	1388	1323	1159	83,50%	87,60%
Fisioterapia	346	322	277	80,06%	86,02%

Ilustración 2. Datos representativos

Según estos datos de 2008, la carrera de informática es la que peor relación matriculados/aprobados tiene de todas las carreras españolas. Esto se debe a que el alto contenido práctico que tiene esta carrera, muchos de los alumnos no consiguen aprobar la parte práctica, y ni tan siquiera pueden optar a presentarse al examen final.

Una buena solución para esto sería una herramienta sencilla que permitiera sustituir en parte esa labor que el profesor no siempre puede realizar, ya que estaría operativa el 100% del tiempo, y entregaría resultados al alumno de forma instantánea.

Una de las principales tareas del profesor, es conseguir mejorar el autoaprendizaje del alumno, con la utilización de herramientas como el autotest ^[3], que permiten conocer al alumno conocer de forma rápida y sencilla, cuáles son sus conocimientos sobre un determinado tema de la asignatura.

1.2 Objetivos del proyecto

Los objetivos principales de este Trabajo Fin de Grado son:

1. Desarrollar una aplicación utilizando la herramienta Microsoft Visual Studio .NET que permita analizar programas escritos en lenguaje C, para hallar así errores lógicos y de estilo, completando así la información dada por el compilador.
2. La herramienta puede ser utilizada tanto por profesores como por alumnos, permitiendo a los primeros analizar de forma rápida y sencilla una gran cantidad de programas, y obteniendo los resultados del análisis en un cómodo archivo para su estudio. Y permitirá a los alumnos corregir esos errores de forma rápida y personal, dotando de mayor autonomía al alumno para fomentar el autoaprendizaje.
3. Además, permite según la necesidad del profesor, centrar el análisis en un determinado campo u otro, pudiéndose centrar en la modularidad del código, la legibilidad, o el uso de las variables en el mismo.

La aplicación permite al alumno realizar una pequeña prueba sobre su programa, que aunque no le asegurará el aprobado en caso de ser pasada, sí que puede ser un indicativo claro de si se va o no por el buen camino.

La aplicación además devuelve información clara del código del alumno y sencilla de entender, a diferencia de otras herramientas similares existentes que se comentarán más adelante, que aunque realizan tareas semejantes, son bastante más complicadas de utilizar, devuelven resultados mucho más simples y complejos de analizar.

Por otro lado, el desarrollo de este proyecto pretende cubrir los siguientes objetivos personales o secundarios:

1. El desarrollo de una aplicación en Microsoft Visual Studio .NET, un lenguaje de programación el cual no se había utilizado en ninguna de las asignaturas cursadas en el Grado de Ingeniería Informática.
2. Un estudio más complejo y detallado del lenguaje C.
3. La realización de una aplicación de esta complejidad y de forma individual, ya que aunque durante la carrera se habían realizado ya proyecto de una índole similar, estos eran realizados entre varios componentes, por lo tanto su dificultad era relativamente menor.

1.3 Contenidos de la memoria

Los contenidos de los que se compone la presente memoria están organizados en forma de capítulos. Cada uno de ellos centra en un aspecto concreto perteneciente a este proyecto. A continuación se muestra un breve resumen del contenido de cada capítulo.

Capítulo 1. Introducción. Se desarrolla una pequeña idea sobre la cual gira el proyecto, una aproximación sobre el problema de la carrera de Informática. Además, incluye una descripción de los objetivos que han motivado la realización de este proyecto, y una breve descripción de cada punto de la memoria.

Capítulo 2. Estado del arte. Se realizará un pequeño repaso sobre el lenguaje de programación C. Se tratarán los tipos de errores existentes en la programación, y se realizará un análisis de las herramientas semejantes a la desarrollada en el proyecto.

Capítulo 3. Fase de Análisis, Diseño e Implementación. Se sitúa el proyecto dentro del marco regulador, el alcance y los requisitos del proyecto, los diferentes casos de uso de la aplicación, además de los distintos módulos con su respectiva funcionalidad.

Capítulo 4. Resultados obtenidos. En este apartado se representan las pruebas realizadas para la comprobación del correcto funcionamiento del programa, y los resultados obtenidos a partir de las mismas.

Capítulo 5. Planificación. Representación de la planificación del proyecto de forma tanto estimada como real.

Capítulo 6. Presupuesto. Inclusión de un presupuesto estimado del coste del proyecto.

Capítulo 7. Conclusiones. Expone las reflexiones obtenidas tras el desarrollo del proyecto, analizando además si se han alcanzado los objetivos marcados al inicio del desarrollo del mismo.

Capítulo 8. Líneas Futuras. Hace alusión a las ideas e intenciones futuras que han surgido al realizar este proyecto.

Anexo. Incluye el Manual de Usuario, para comprender la correcta utilización de la aplicación.

Referencias. Presenta las fuentes y materiales utilizados para la realización del proyecto.

Capítulo 2. Estado del arte

En este capítulo se hará una breve descripción del lenguaje de programación C, incluyendo su historia y sus características más importantes, herramientas existentes que permiten conocer los errores en el lenguaje C, y los tipos de errores existentes en la programación.

2.1 Introducción a los lenguajes de la programación

Un lenguaje de programación es un lenguaje formal inventado para crear procesos que puedan ser ejecutados por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación ^[4].

Si se habla en especial sobre el lenguaje C, es un lenguaje de programación de nivel medio ya que combina los elementos del lenguaje de alto nivel con la funcionalidad del ensamblador.

Un lenguaje de programación de alto nivel se caracteriza por expresar el algoritmo de una manera mucho más cercana al lenguaje humano que al lenguaje máquina perteneciente a las computadoras. Los lenguajes de alto nivel se crearon para que el usuario pudiese comprender los programas de forma más rápida, y desarrollar los mismos de una manera mucho más sencilla.

En el lenguaje de programación de alto nivel, en lugar de tratar con registros, direcciones de memoria y las pilas de llamadas, estos utilizan las variables, matrices, objetos, aritmética compleja o expresiones booleanas, subrutinas y funciones, bucles, hilos, cierres y otros conceptos, haciendo mucho más sencillo el desarrollo y comprensión de los programas creados por el usuario ^[5].

El lenguaje ensamblador por contra, es un lenguaje de programación que es una traducción directa del código de máquina (Éste código es interpretado por el microprocesador), para que pueda ser entendible por los seres humanos, por lo tanto es un lenguaje de bajo nivel. Utiliza registros, y direcciones de memoria que complican su seguimiento, y ralentizan su desarrollo. Aunque no todo son desventajas, el lenguaje ensamblador permite una optimización que no se consigue con lenguajes de medio y alto nivel, además consigue la creación de programas muy rápidos y muy pequeños.

2.2 Introducción al lenguaje C

En 1967 Martin Richard creó el lenguaje BCPL (Lenguaje de Programación Básico Combinado) que fue la base para la creación del lenguaje B escrito por Ken Thompson en 1970 con la intención de recodificar el UNIX, que en su fase de arranque estaba codificado en lenguaje ensamblador.

Como ocurre con el lenguaje de programación B, que es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C también es muy utilizado

para la creación de software de sistemas debido a la eficiencia del código que produce, aunque también es utilizado para crear aplicaciones.

Tiene las características que suelen ser normales en el lenguaje de alto nivel, pero consta también de elementos que le acercan al lenguaje de muy bajo nivel.

Uno de los principales objetivos en el lenguaje C es que sólo se necesiten unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que haga falta un soporte intenso en tiempo de ejecución. Debido que C puede ser escrito a un nivel muy bajo de programación, fue utilizado durante mucho tiempo como punto intermedio entre distintos lenguajes.

Debido a este bajo nivel del que se compone y de tener unas características realmente sencillas, se pueden desarrollar compiladores de C fácilmente. Por tanto, el lenguaje C está disponible en un amplio abanico de plataformas (más que cualquier otro lenguaje).

El desarrollo inicial de C se llevó a cabo en los Laboratorios Bell de AT&T entre 1969 y 1973; según Ritchie, el periodo más creativo tuvo lugar en 1972. Se le dio el nombre "C" porque muchas de sus características fueron tomadas de un lenguaje anterior llamado "B".

Hay muchas leyendas acerca del origen de C y el sistema operativo con el que está íntimamente relacionado, Unix. La más conocida de ellas es:

Tanto Thompson como Ritchie deseaban jugar al videojuego Space Travel, un videojuego que simulaba un viaje interestelar por el Sistema Solar y que consistía en evitar las colisiones con los distintos asteroides existentes por el espacio. Su idea fue jugar en el mainframe (el ordenador más potente y grande de su compañía), pero debido a su escasa capacidad de procesamiento, no podían manejar la aeronave de la forma correcta y acabaron por desistir. Decidieron cambiar el juego a una máquina distinta que estaba sin usar, pero esta no tenía sistema operativo alguno instalado, por lo que decidieron crear uno para esa máquina.

Su idea inicial era copiar la mayor parte del sistema operativo que contenía otra máquina de potencia similar, pero el problema vino cuando el código de dicho sistema operativo estaba en ensamblador, por lo que optaron por crear uno similar, pero en lenguaje de alto nivel para que fuera portátil y se pudiera instalar en diferentes computadoras de forma sencilla. Inicialmente pensaron utilizar el lenguaje de programación B, pero no sacaba el máximo rendimiento posible de las máquinas de la época, por lo que optaron por crear un nuevo lenguaje, C.

En 1973, el lenguaje C se había vuelto tan potente que la mayor parte del kernel Unix, originalmente escrito en el lenguaje ensamblador, fue reescrita en C. Éste fue uno de los primeros núcleos de sistema operativo implementados en un lenguaje distinto al ensamblador.

2.3 Ventajas y desventajas del lenguaje C

Las ventajas de la utilización del lenguaje C son:

- Un núcleo del lenguaje muy simple, con funcionalidades añadidas importantes, como pueden ser funciones matemáticas, de manejo de archivos o de determinados tipos de variables, proporcionadas por las diferentes bibliotecas existentes.
- La utilización de un sistema de tipos para las variables, lo cual evita operaciones sobre las mismas que puedan producir errores.
- Es un lenguaje preprocesado ^[6], que es una fase que utilizan algunos lenguajes de programación, que se usa antes de la fase de compilación, y que permite, entre otras cosas, incluir varios archivos en el código fuente.
- Es un lenguaje de propósito general, puede ser utilizado para desarrollar desde sistemas operativos a gestores de bases de datos.
- Tiene un pequeño grupo de palabras clave (solo 32), siendo estas muy sencillas de entender y aprender por el usuario.
- Paso de parámetros a las funciones tanto por valor como por referencia, pasándose en este caso la variable, o la dirección de memoria de la misma según corresponda.
- Incluye una serie de librerías para conseguir un gran número de funciones accesibles en cualquier programa.
- A pesar de su bajo nivel es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas conocidos.

Los principales inconvenientes en el uso del lenguaje C son:

- Su principal inconveniente es que programar en C es mucho más lento que programar en otros lenguajes, la razón principal es que el compilador simplemente analiza y traduce el código sin añadir nada.
- Al tratarse de un lenguaje bastante antiguo, inicialmente todos los que utilizaban lenguajes de este tipo eran programadores con experiencia, y se asumía que los usuarios que lo acabarían usando poseían conocimientos de programación también elevados.
- Existe además el problema de la basura nativa ^[7], que consiste en que es el propio usuario el que debe reservar, liberar y tratar la memoria reservada por el programa, utilizando para ello las diferentes librerías existentes, y las funciones que estas poseen.

2.4 Proceso de compilación de un programa en lenguaje C

Las etapas por las que pasa un programa en lenguaje C hasta poder ser ejecutado son:

- **Edición.** Consiste en escribir en lenguaje C, las distintas instrucciones para que realicen la tarea deseada.
- **Preproceso.** Sirve para modificar el código obtenido en la etapa anterior, y que se encarga de analizar si hay que realizar otras tareas antes de realizar la compilación.
- **Compilación.** Se traduce el código fuente obtenido tras las etapas anteriores escrito en lenguaje de alto nivel, en código máquina, siempre y cuando no se produzca algún error de compilación.
- **Enlace.** Se enlaza el código de las funciones “especiales” utilizadas en el código fuente, obteniéndolo de las distintas bibliotecas añadidas al propio código.

Como resultado de estas tareas, si no se ha producido ningún error, se obtendrá un ejecutable listo para utilizar.

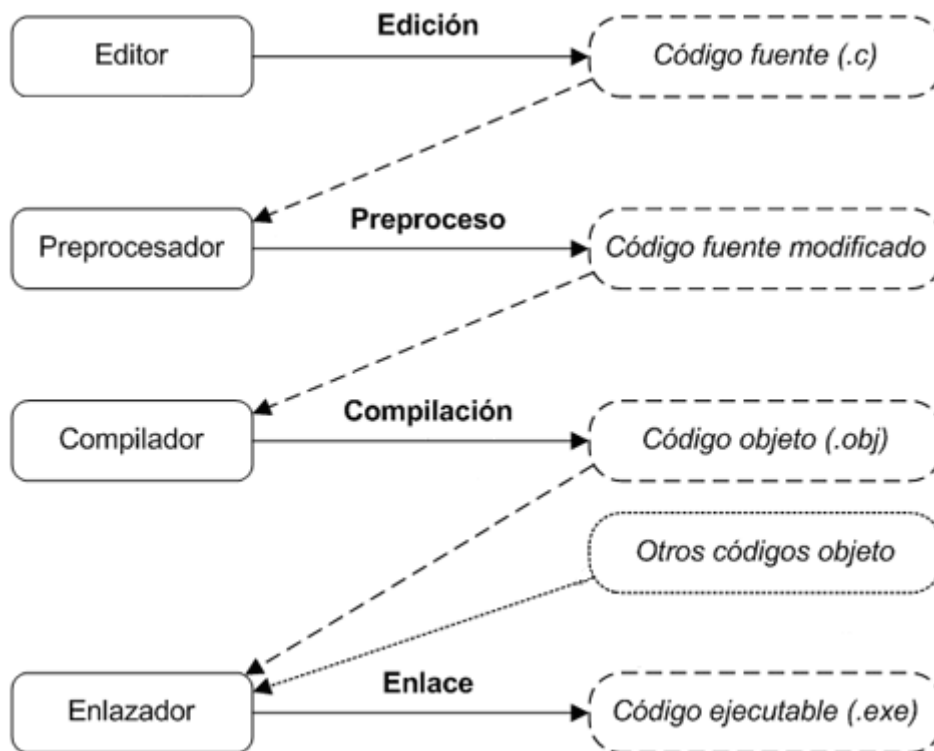


Ilustración 3. Etapas de compilación

2.5 Análisis y detección de errores

2.5.1 Introducción

Debido a que cada lenguaje de programación tiene sus propias reglas, debe existir un compilador específico para cada lenguaje. Si el código fuente está sintácticamente bien definido, el compilador pasará a generar un código objeto, mientras que si por alguna razón no se consigue generar el código objeto, generalmente se mostrará una lista con los distintos errores encontrados, para proceder a la depuración del programa.

En función del momento en el que se produzca el error, este puede ser de dos tipos:

- **Errores en tiempo de compilación.** Son los que se producen antes de la ejecución del programa, en el proceso de compilación del mismo, y conllevan que no se genere el ejecutable correspondiente. Normalmente se deben a algún error en la escritura del código.
- **Errores en tiempo de ejecución.** Son los que se producen en la propia ejecución del programa, son los más complicados de encontrar, ya que no los puede detectar el compilador y puede que solo se de en algunos casos excepcionales en la ejecución del código.

2.5.2 Tipos de errores

Llamamos error a cualquier circunstancia que da lugar a un malfuncionamiento de un programa. Por malfuncionamiento se entiende una respuesta indeseada: desde meros problemas estéticos hasta graves fallos o bloqueos ^[8]. Hay distintos tipos de errores:

- **Errores de Sintaxis.** Son aquellos errores que se producen debido a un fallo en las normas de escritura de un determinado lenguaje de programación, pueden deberse a olvidos o desconocimiento, ausencia o mal uso de elementos separadores de sentencias (punto y coma, llaves de inicio y fin de sentencias,...), o errores en la escritura de palabras clave. Son fáciles de detectar ya que la mayoría de los compiladores los encuentran con facilidad.
- **Errores por procesos no válidos.** Estos errores se producen cuando aunque a simple vista la construcción de la sentencia es correcta, se produce un fallo en alguno de las reglas que define el lenguaje de programación utilizado. Estos errores incluyen el uso de llamadas a funciones no declaradas en el código, usar variables no declaradas, o asignar un valor a una variable de un determinado tipo, que no coincide con el tipo de dicha variable (es decir, asignarle un valor tipo char, (a) a una variable definida como integer (que solo acepta números)).
- **Errores Lógicos.** Son los que se producen debido a un mal diseño de los algoritmos del código. El principal problema de estos errores es que son muy difíciles de detectar, ya que dependen de la experiencia del programador, y además tienen el añadido de ser imprevisibles, ya que pueden producirse solo en determinadas ejecuciones del programa. Estos errores pueden ser de dos tipos:
 - **Tipo bucle infinito.** Aquellos que devuelven como resultado un bloqueo del programa, al acceder a una determinada sentencia o bucle de sentencias, y no poder salir de él debido a alguna condición. También están dentro de esta categoría aquellos bucles que consumen un tiempo desmedido respecto a lo esperado.
 - **Tipo resultado incorrecto.** El funcionamiento del programa es el correcto, pero el resultado no tiene nada que ver con el esperado.
- **Errores Gestionados.** Un error es gestionado cuando se produce cualquiera de los errores anteriores, este es detectado por el compilador, y el programador pasa a subsanar dicho error. El problema ocurre llega cuando el compilador no devuelve un mensaje alertando del error, por lo tanto la gestión del mismo se vuelve algo más complicada y depende únicamente del programador.
- **Errores no Gestionados.** Son aquellos errores que no son subsanados por el usuario, lo cual no tiene por qué indicar un incorrecto funcionamiento del programa. Un ejemplo de esto puede ser la declaración de una variable sin usar, lo cual el compilador reconocerá como un error, pero es algo que no influirá en el desarrollo del programa.
- **Errores sutiles.** Son aquellos errores casi despreciables, que no influyen en la ejecución del programa, como puede ser la declaración de una variable que no se usa.
- **Errores intermedios.** Son aquellos errores que permiten la ejecución del programa, pero pueden alterar el resultado del mismo.
- **Errores groseros.** Son aquellos errores que imposibilitan la correcta ejecución del programa.

2.5.3 Clasificación de los errores

Según en qué nos basemos para organizar los errores, tenemos diferentes clasificaciones:

- Clasificación según los efectos que generan:

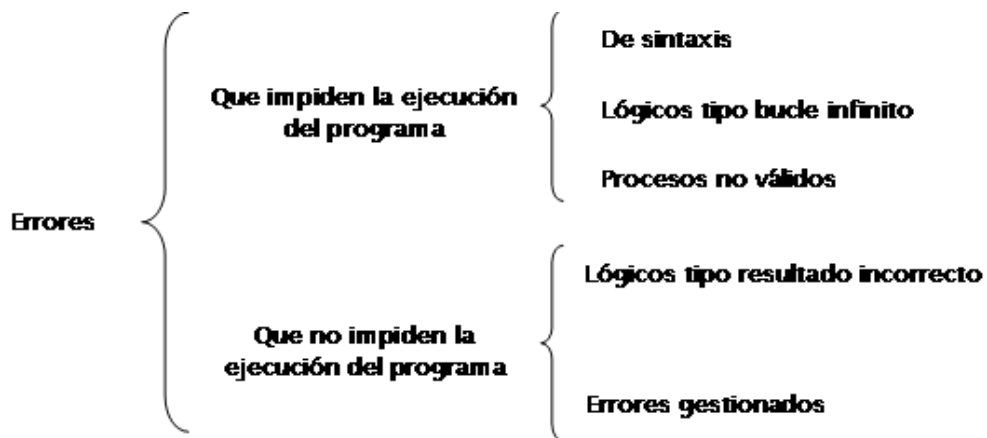


Ilustración 4. Clasificación de errores según efectos generados

- Clasificación según el momento en que se producen:

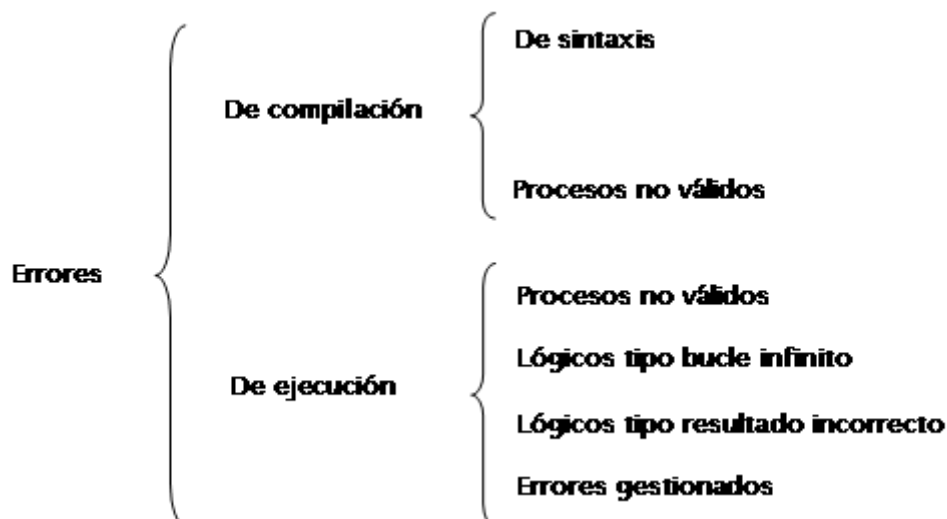


Ilustración 5. Clasificación de errores según el momento en que se producen

- Clasificación atendiendo a la naturaleza del error:



Ilustración 6. Clasificación de errores según su naturaleza

- Clasificación según el tratamiento que reciben:



Ilustración 7. Clasificación de errores según su tratamiento

- Clasificación según su importancia:

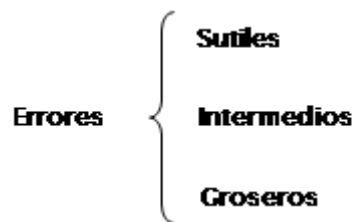


Ilustración 8. Clasificación de errores según su importancia

2.6 Herramientas de compilación

Al programar en C, puede ser recomendable utilizar herramientas de compilación y depuración, debido a que una gran parte del tiempo que un programador utiliza para realizar un programa se basa en la búsqueda de errores.

2.6.1 Compilador gcc

El compilador de GNU gcc^[9] tiene como objetivo mejorar el compilador usado en todos los sistemas GNU, incluyendo la variante GNU/Linux. Usa un entorno de desarrollo abierto y soporta múltiples plataformas para aumentar su uso, y así atraer a la mayor cantidad de desarrolladores posible.

Es una aplicación que dado un fichero, o conjunto de ficheros de código en C genera un programa ejecutable. Al igual que cualquier herramienta de Linux, puede ser invocada desde el terminal de comandos. Además mediante un comando sencillo, permite compilar cualquier programa en lenguaje C de forma sencilla ^[10].

```
daniel@daniel-VirtualBox:~/Escritorio$ gcc hola.c -o hola
daniel@daniel-VirtualBox:~/Escritorio$ ./hola
Hola mundo
daniel@daniel-VirtualBox:~/Escritorio$
```

Ilustración 9. Ejecución sencilla del gcc

El uso de la herramienta gcc no solo permite compilar cualquier programa en lenguaje C, si no que reconoce los errores cometidos, siempre y cuando estos no permitan completar la compilación del código.


```
daniel@daniel-VirtualBox:~/Escritorio$ gcc hola.c -o hola
hola.c: En la función 'main':
hola.c:6:1: error: expected ';' before '}' token
daniel@daniel-VirtualBox:~/Escritorio$
```

Ilustración 10. Ejecución gcc con error

Al contrario que ocurre en los compiladores de otros lenguajes de programación, el de C no realiza una comprobación exhaustiva de que el programa cumple estrictamente con la definición del lenguaje. Cuando en un programa en C se pasan por alto pequeños detalles, el compilador asume un comportamiento por defecto y genera igualmente un ejecutable. Es decir, si la compilación del programa devuelve únicamente warnings (avisos) pero se completa dicha compilación, no mostrará mensaje alguno por pantalla. El problema de esto es que la definición del programa no tiene por qué ser correcta, y por tanto los resultados obtenidos podrían no ser los esperados, desconociendo de esta forma el programador cual es la causa del error.

Debido a esto, gcc tiene la opción `-W` que se permite incluir de forma específica qué tipo de comprobaciones queremos que haga el compilador.

```
daniel@daniel-VirtualBox:~/Escritorio$ gcc hola.c -o hola
daniel@daniel-VirtualBox:~/Escritorio$ ./hola
Hola mundo
daniel@daniel-VirtualBox:~/Escritorio$ gcc -Wall hola.c -o hola
hola.c: En la función 'main':
hola.c:5:5: aviso: variable 'a' sin usar [-Wunused-variable]
daniel@daniel-VirtualBox:~/Escritorio$ ./hola
Hola mundo
daniel@daniel-VirtualBox:~/Escritorio$
```

Ilustración 11. Ejecución gcc con warning

En general el uso de la herramienta gcc sólo tiene ventajas, ya que indica todos los errores cometidos de forma clara, permite depurar el código de forma sencilla ya que señala la línea en la cual se comete el error, y mediante un simple comando se pueden conocer los avisos de nuestro programa, que son pequeños errores que pueden alterar el funcionamiento del mismo.

El problema de esta herramienta es que no indica los errores de estilo recogidos en la aplicación desarrollada en este proyecto. No comprueba el uso de comentarios, que facilitan la comprensión de cualquier programa, ni comprueba la correcta indentación del código, o la coherente definición de variables, el uso de variables globales, o el tamaño excesivo de las subrutinas o funciones, entre otras cosas.

En resumen, el uso de la herramienta gcc es recomendable, aunque no trata todos los aspectos necesarios para el desarrollo correcto de un programa.

2.6.2 Herramienta de depuración GDB

GDB^[11] es uno de los depuradores más utilizados dentro de la plataforma Linux. Es una herramienta que permite ejecutar un programa de forma controlada, avanzando instrucción a instrucción hasta alcanzar una condición deseada, observar el valor de las variables en un determinado momento, o cambiar el valor de las variables. Es una herramienta que permite

encontrar errores de difícil acceso, debido a la posibilidad de tiene de ejecutar un programa instrucción a instrucción.

Mediante un sencillo comando, se puede ejecutar la herramienta sobre cualquier programa, para realizar un análisis sobre el mismo.

```
a0065124@guernika:~/Desktop/dssoo$ gdb ./valgrind_test
GNU gdb (GDB) 7.0.1-debian
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /users/alumnos-11-12/a0065124/Desktop/dssoo/valgrind_test...done.
(gdb)
```

Ilustración 12. Ejecución sencilla del GDB

Mediante distintos comandos, la herramienta permite saber en cualquier momento que desee el usuario, entre otras tareas:

- Observar el valor de los argumentos de una determinada función.
- Observar el valor de las variables de entorno.
- Hacer una parada en un determinado punto de la ejecución.
- Avanzar instrucción a instrucción, o continuar con la ejecución normal del programa.
- Permite detener el programa en una determinada línea del código.
- imprimir el valor de una variable en un determinado momento.
- Mostrar el valor contenido de una variable cada vez que esta cambie de valor.

```
(gdb) break 93
Breakpoint 1 at 0x80487c9: file valgrind_test.c, line 93.
(gdb) run
Starting program: /users/alumnos-11-12/a0065124/Desktop/dssoo/valgrind_test
function 1 - a: 0, b: 9
function 2 - a: 9
function 3 - value: 1
cccccccccccccccccccc - this is the string

Breakpoint 1, function_persons () at valgrind_test.c:93
93         for (i=0; i < persons; i++){
(gdb) n
94             registered[i].name = (char *)malloc(50);
(gdb) n
95             sprintf(registered[i].name, "name %i", i);
(gdb) n
96             registered[i].number = 20549;
(gdb) n
93         for (i=0; i < persons; i++){
(gdb) print i
$1 = 0
(gdb) print registered[i].number
$2 = 20549
```

Ilustración 13. Ejecución normal del GDB

En general, la herramienta GDB es un gran depurador para ejecutar programas instrucción a instrucción, y ayuda a encontrar errores cuya única alternativa es la inclusión de los printf (mensaje por pantalla) para pintar las variables después de cada iteración e intentar encontrar la solución de forma manual. El problema es que para programas de gran tamaño, la dificultad para realizar la depuración con GDB también es bastante elevada, añadiendo que su única función es esa, y que la búsqueda del error tiene que ser realizada por el propio usuario, ya que la herramienta no devuelve ningún indicativo de donde está el error.

2.6.3 Herramienta de depuración Valgrind

La herramienta de depuración Valgrind [\[12\]](#), permite ejecutar programas dentro de un entorno controlado, permitiendo analizar la ejecución desde distintos puntos de vista. Mediante su uso, se pueden detectar problemas de la siguiente índole:

- Utilización de variables no inicializadas.
- Hacer uso de memoria que posteriormente no es liberada.
- Utilizar variables fuera de su rango (acceso a la posición 11 de un array que solo tiene 10 posiciones declaradas).

Uno de los principales problemas del uso de la herramienta Valgrind, es la manera que tiene de representar los errores obtenidos de su análisis, ya que sus resultados no son demasiado intuitivos para el usuario.

```
==16770== Invalid write of size 4
==16770==   at 0x804851D: function1 (in /users/alumnos-11-12/a0065124/Desktop/dssoo/valgrind_test)
==16770==   by 0x8048840: main (in /users/alumnos-11-12/a0065124/Desktop/dssoo/valgrind_test)
==16770== Address 0x4190050 is 0 bytes after a block of size 40 alloc'd
==16770==   at 0x4023F50: malloc (vg_replace_malloc.c:236)
==16770==   by 0x80484EC: function1 (in /users/alumnos-11-12/a0065124/Desktop/dssoo/valgrind_test)
==16770==   by 0x8048840: main (in /users/alumnos-11-12/a0065124/Desktop/dssoo/valgrind_test)
```

```
16 void function1(){
17     int elements = 10;
18     int a = 0;
19     int b = 0;
20     int * values = (int *) malloc(elements * sizeof(int));
21     int * aux = (int *) malloc(elements * sizeof(int));
22     int i = 0;
23     for (i = 0; i <= elements; i++){
```

Ilustración 14. Ejecución de Valgrind

En el caso de la Ilustración 14, Valgrind indica que en la función1 se ha declarado un array de 10 posiciones, pero luego en la línea 23 se recorren 11 posiciones del mismo, alcanzando así una posición fuera de rango. Como se comentaba antes, además que los errores encontrados por la herramienta no son demasiado extensos, la forma que tiene de indicárselos al usuario tampoco ayudan demasiado a su uso.

Otro de los graves problemas de la herramienta es la pérdida de rendimiento del computador durante su uso, ya que los programas se ejecutan entre cinco y veinte veces más despacio durante su utilización, y además el uso de memoria del ordenador es mucho mayor. Es por ello que lo recomendable es usar únicamente Valgrind para encontrar aquellos errores que antes no se hayan podido subsanar de ninguna otra manera.

2.6.4 Conclusiones finales

Haciendo un análisis entre las distintas herramientas existentes en el mercado que ayudan a la compilación y depuración de programas en lenguaje C, aunque cada una aporta unos resultados distintos, ninguna abarca los puntos tratados en la aplicación desarrollada en este proyecto.

Si bien el compilador gcc permite dar a conocer al usuario cualquier error cometido en el código, si dicho error no evita la generación del ejecutable, este no siempre es mostrado al usuario. Centrándose además únicamente en el análisis teórico del programa, sin entrar en ningún momento en el estilo del mismo.

Si se analizan las herramientas GDB o Valgrind, ambas aportan resultados bastante escasos al usuario. Mientras Valgrind se centra, de forma más o menos intuitiva, en realizar un seguimiento del uso de la memoria del programa, lo cual permitiría al usuario arreglar algunos de los errores lógicos existentes, GDB ni tan siquiera le devuelve al usuario algún resultado, sino que es él mismo quien debe ir instrucción a instrucción hasta encontrar el error cometido, imprimiendo el valor de las variables en el momento deseado, y con el resto de recursos de la herramienta.

El resumen es que ninguna de las herramientas descritas entra a tratar el estilo del programa, y tampoco alcanza el análisis que en la aplicación de este proyecto se desarrolla. Lo recomendable para un usuario sería compilar su programa mediante la herramienta gcc, y una vez esta no le devuelva ningún error, pasársela a esta aplicación. Una vez el resultado obtenido por la aplicaciones el correcto, comprobar que la solución obtenida por el programa es correcta, y en caso contrario, buscar el error con una de las dos herramientas descritas anteriormente.

Capítulo 3. Fase de Análisis, Diseño e Implementación

Este capítulo se centra más en el desarrollo de la propia aplicación. Se tendrá en cuenta el marco regulador donde se incluye la herramienta, el alcance y los requisitos de la misma todo ello dentro de la fase de análisis. Se explicará el diseño de la aplicación justificando de forma clara cada elección dentro de la fase de diseño, y por último se comentará el proceso de codificación de la aplicación dentro de la fase de implementación.

3.1 Fase de Análisis

El objetivo de la fase de análisis es obtener una especificación detallada del sistema a construir, lo cual servirá como punto de partida para el posterior diseño de la aplicación. En esta fase se define el problema del que surge la necesidad y se captan las necesidades a resolver, para posteriormente modelar el problema y resolverlo finalmente en el diseño, y llevarlo a cabo mediante su implementación.

3.1.1 Marco regulador

La calidad es un conjunto de propiedades pertenecientes a una cosa que permite que se le categorice y compare con otras de su misma especie. Podría definirse también, como la aptitud que tiene un producto o servicio para satisfacer unas determinadas necesidades de un usuario.

Centrándose más en la calidad de software, es un conjunto de características que definen su utilidad y función.

Según la normativa ISO: 9001 hay 8 principios existentes para lograr el objetivo de la calidad en un desarrollo software:

- **Enfoque al cliente o usuario.** Una organización depende de los clientes que solicitan su servicio, por ende se deben comprender sus necesidades y satisfacer sus requisitos.
- **Liderazgo.** Es el líder el que debe conseguir que cada integrante del grupo alcance sus objetivos, incluyendo al propio cliente.
- **Participación.** Debe asegurarse que todos los participantes en el desarrollo del software están comprometidos al 100% con el proyecto.
- **Enfoque basado en procesos.** Es más fácil alcanzar el objetivo cuando todos los recursos son utilizados como un todo.
- **Enfoque del sistema hacia la gestión.** Tratar los distintos procesos consiguen mejorar tanto la eficiencia como la eficacia del proyecto.
- **Mejora continua.** Este debe ser el objetivo principal de cualquier organización.
- **Enfoque basado en hechos.** Las decisiones a tomar se deben basar en la información y los datos conseguidos, ya sea a partir del cliente o mediante investigación.
- **Interrelaciones de mutuo beneficio.** Una organización y cualquiera que se comunique con ella componen una relación, la mejora de esta también mejora la calidad del servicio.

La norma ISO 9001 ha derivado en la normativa ISO/IEC 9126 ^[13], según la cual para medir la calidad del software se pueden utilizar el siguiente conjunto de características:

- **Funcionalidad.** El software debe contener un conjunto de funciones y sus respectivas propiedades, para satisfacer los distintos requisitos del proyecto.
- **Fiabilidad.** El software debe cumplir su función bajo ciertas condiciones durante un tiempo establecido.
- **Usabilidad.** Se refiere a la facilidad o dificultad que el usuario que el usuario se encontrará a la hora de utilizar el software.
- **Eficiencia.** Se refiere al funcionamiento que desempeña un software, en función de los recursos invertidos en su uso.
- **Mantenibilidad.** Se refiere a la cantidad de esfuerzo requerido para conseguir el correcto funcionamiento del software, en el momento en el que se produce un fallo en el mismo.
- **Portabilidad.** Facilidad o no del software, para ser utilizado en diferentes plataformas.

3.1.2 Definición del sistema

En este apartado se realizará una definición del sistema, a partir del alcance del mismo, se indicaran algunas restricciones generales, se comentará el entorno de desarrollo y se detallaran los diferentes requisitos obtenidos.

3.1.3 Alcance del sistema

La herramienta consiste en una aplicación de escritorio a ejecutar en cualquier ordenador con el sistema operativo Windows, que puede ser utilizada tanto por profesores como por alumnos, y que permite analizar y detectar posibles errores en programas escritos en lenguaje C. Dando a los profesores la posibilidad de analizar un gran cantidad de ficheros de forma rápida y fácil, y guardar una determinada configuración para ser cargada después, o ser entregada a sus alumnos. Y permite a éstos últimos analizar sus propios programas, para que sean ellos quienes busquen y arreglen sus propios errores, fomentando el autoaprendizaje.

3.1.4 Restricciones generales

La primera restricción existente en la herramienta, es que generalmente, en función de donde se encuentre la base de datos con los usuarios y sus contraseñas correspondientes, el usuario que desee hacer uso de la aplicación deberá estar conectado a internet.

El usuario además deberá tener en su ordenador instalado Microsoft Office, debido a que los resultados obtenidos del análisis son escritos en un fichero .doc, es condición indispensable tener instalado Microsoft Word.

Añadir que, siempre que el programa quiera ser ejecutado, deberá haber un servidor u ordenador que contendrá la base de datos, y que sirve para que los usuarios se identifiquen en la aplicación, y que además deberá estar operativo el mayor tiempo posible, para dar mayor cobertura de uso a los usuarios.

3.1.5 Entorno de desarrollo

La herramienta ha sido desarrollada completamente en lenguaje Visual Basic .NET, utilizando el entorno de desarrollo integrado Microsoft Visual Studio .NET.

Se ha utilizado el MySQL Community Server como servidor de datos que contiene las tablas de alumnos y profesores. Y además mediante el uso de MySQL WorkBench se han creado las distintas bases de datos.

3.1.6 Requisitos de la herramienta

En este apartado se identificarán los distintos requisitos divididos en dos grupos, los requisitos funcionales y los no funcionales.

Para su representación se utilizará una tabla descriptiva como la siguiente:

ID: REQ_XX_NN	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	
Descripción:	

Tabla 1. Plantilla de requerimientos

- **ID:** Identificador unívoco de cada requisito que tendrá el siguiente formato: REQ_XX_NN, donde REQ identifica que es un requisito, XX son las dos primeras iniciales de la categoría del requisito, se representará como FU a los requisitos funcionales, y como NF a los requisitos no funcionales, y NN es un número de dos cifras diferenciador para cada requisito.
- **Prioridad:** Es la necesidad de que un requisito se realice frente a otro, cuanta más prioridad, antes debe ser realizado.
- **Fuente:** Es el origen de cada requisito.
- **Riesgo:** Es la necesidad del requisito dentro del sistema acompañado de la dificultad de su desarrollo. Es decir, un requisito con riesgo bajo tiene poco impacto sobre el resto del sistema y es fácil de desarrollar mientras que un requisito con riesgo alto tiene un gran impacto sobre el funcionamiento del sistema y puede ser complejo en su desarrollo.
- **Verificabilidad:** Indica si es posible comprobar que el requisito ha sido añadido al diseño.
- **Necesidad:** Es el interés de que un requisito sea realizado. Si es necesario tendrá que realizarse en cualquier caso, si es deseable se intentará realizar pero podrá ser eliminado por falta de tiempo y si es opcional se realizará solo en caso de que el resto de requisitos se cumplan por completo.
- **Nombre:** Hace referencia a un pequeño título aclarativo y sencillo de lo que trata el requisito.
- **Descripción:** Representa la definición completa y detallada de cada requisito.

3.1.6.1 Requisitos funcionales

Los requisitos funcionales son la explicación de los servicios que contendrá el sistema, de la manera en que éste reaccionará a entradas particulares. En general describe que es lo que debe hacer el sistema.

ID: REQ_FU_01	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Comprobar conexión
Descripción:	La herramienta debe permitir al usuario comprobar si está o no conectado a la base de datos.

Tabla 2. REQ_FU_01

ID: REQ_FU_02	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Iniciar Sesión
Descripción:	La herramienta debe permitir al usuario iniciar sesión.

Tabla 3. REQ_FU_02

ID: REQ_FU_03	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Perfiles
Descripción:	La herramienta debe permitir tener varios perfiles (alumno y profesor), y en función de si se es uno u otro, podrá o no modificar las distintas opciones de la aplicación.

Tabla 4. REQ_FU_03

ID: REQ_FU_04	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Cerrar Sesión
Descripción:	La herramienta debe permitir al usuario cerrar sesión una vez este dentro de la aplicación.

Tabla 5. REQ_FU_04

ID: REQ_FU_05	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Salir
Descripción:	La herramienta debe permitir al usuario salir de la aplicación.

Tabla 6. REQ_FU_05

ID: REQ_FU_06	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input checked="" type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Cargar configuración
Descripción:	La herramienta debe permitir poder cargar un archivo de configuración, que contenga una determinada configuración guardada anteriormente por el profesor. En caso de no existir ninguna configuración anterior, se cargará una por defecto.

Tabla 7. REQ_FU_06

ID: REQ_FU_07	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input checked="" type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Guardar configuración
Descripción:	La herramienta debe permitir a un usuario que acceda como profesor guardar una configuración elegida.

Tabla 8. REQ_FU_07

ID: REQ_FU_08	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Seleccionar archivo
Descripción:	La herramienta debe permitir a un usuario que acceda como alumno seleccionar un determinado archivo para ser analizado.

Tabla 9. REQ_FU_08

ID: REQ_FU_09	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Seleccionar ruta archivos
Descripción:	La herramienta debe permitir a un usuario que acceda como profesor seleccionar una determinada carpeta, que contenga uno o varios archivos para ser analizados.

Tabla 10. REQ_FU_09

ID: REQ_FU_10	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Seleccionar ruta resultados
Descripción:	La herramienta debe permitir a un usuario seleccionar una determinada ruta donde se creará el archivo con el análisis.

Tabla 11. REQ_FU_10

ID: REQ_FU_11	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar rutas
Descripción:	La herramienta debe comprobar que tanto la ruta del fichero o carpeta de ficheros a analizar, como la ruta donde se creará el fichero con los resultados están seleccionadas.

Tabla 12. REQ_FU_11

ID: REQ_FU_12	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Crear archivo resultados
Descripción:	La herramienta debe generar un archivo que contenga el resultado del análisis realizado.

Tabla 13. REQ_FU_12

ID: REQ_FU_13	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Mostrar ayuda
Descripción:	La herramienta mostrará una ayuda al usuario, con una breve explicación del funcionamiento de la aplicación, desde los pasos a seguir, hasta una pequeña información de los distintos errores buscados en el código.

Tabla 14. REQ_FU_13

ID: REQ_FU_14	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar formato
Descripción:	La herramienta debe comprobar que las funciones, procedimientos, variables y constantes del código están creadas todas con el mismo formato.

Tabla 15. REQ_FU_14

ID: REQ_FU_15	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar indentación
Descripción:	La herramienta debe comprobar que la correcta indentación del código.

Tabla 16. REQ_FU_15

ID: REQ_FU_16	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar comentarios
Descripción:	La herramienta debe comprobar que las funciones y los procedimientos, además del código, están correctamente comentados.

Tabla 17. REQ_FU_16

ID: REQ_FU_17	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar saltos
Descripción:	La herramienta debe comprobar la existencia o no en el código de saltos incondicionales (incluyendo el BREAK, CONTINUE, GOTO y RETURN).

Tabla 18. REQ_FU_17

ID: REQ_FU_18	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar break
Descripción:	La herramienta debe comprobar la existencia o no en el código de la sentencia break.

Tabla 19. REQ_FU_18

ID: REQ_FU_19	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar return
Descripción:	La herramienta debe comprobar la existencia o no en el código de la sentencia return.

Tabla 20. REQ_FU_19

ID: REQ_FU_20	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar continue
Descripción:	La herramienta debe comprobar la existencia o no en el código de la sentencia continue.

Tabla 21. REQ_FU_20

ID: REQ_FU_21	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar goto
Descripción:	La herramienta debe comprobar la existencia o no en el código de la sentencia goto.

Tabla 22. REQ_FU_21

ID: REQ_FU_22	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar tamaño funciones
Descripción:	La herramienta debe comprobar el tamaño máximo de las funciones y los procedimientos.

Tabla 23. REQ_FU_22

ID: REQ_FU_23	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Indicar tamaño funciones
Descripción:	La herramienta debe permitir indicar cuál es el tamaño máximo permitido para las funciones y procedimientos en el código (valor permitido comprendido entre 5 y 99).

Tabla 24. REQ_FU_23

ID: REQ_FU_24	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Comprobar valor tamaño
Descripción:	La herramienta debe comprobar que el valor introducido para el tamaño de las funciones está entre 5 y 99.

Tabla 25. REQ_FU_24

ID: REQ_FU_25	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar definición
Descripción:	La herramienta debe comprobar si las funciones y procedimientos están correctamente definidas en el código.

Tabla 26. REQ_FU_25

ID: REQ_FU_26	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar devolución funciones
Descripción:	La herramienta debe comprobar que tanto las funciones como los procedimientos devuelven las variables de forma correcta.

Tabla 27. REQ_FU_26

ID: REQ_FU_27	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar interfaz de usuario
Descripción:	La herramienta debe comprobar que la interfaz de usuario definida en el código este realizada de forma correcta. Incluyendo: <ul style="list-style-type: none"> - Un subprograma que modifique datos no muestre nada por pantalla. - Siempre que se lea un dato, deberá haber antes un mensaje por pantalla pidiendo dicho dato.

Tabla 28. REQ_FU_27

ID: REQ_FU_28	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar función subprograma
Descripción:	La herramienta debe comprobar que no existan funciones en el código que modifiquen datos y a su vez muestren mensajes por pantalla.

Tabla 29. REQ_FU_28

ID: REQ_FU_29	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar lecturas de datos
Descripción:	La herramienta debe comprobar que si se lee un dato, antes se muestre un mensaje por pantalla pidiéndolo.

Tabla 30. REQ_FU_29

ID: REQ_FU_30	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar variable no usada
Descripción:	La herramienta debe comprobar si existen variables sin usar.

Tabla 31. REQ_FU_30

ID: REQ_FU_31	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar asignaciones
Descripción:	La herramienta debe comprobar si existen varias asignaciones de una variable seguidas, sin hacer uso de la misma entre ellas.

Tabla 32. REQ_FU_31

ID: REQ_FU_32	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar variables sin inicializar
Descripción:	La herramienta debe comprobar si existen variables sin inicializar en el código.

Tabla 33. REQ_FU_32

ID: REQ_FU_33	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Comprobar variables globales
Descripción:	La herramienta debe comprobar si existen variables globales en el código.

Tabla 34. REQ_FU_33

ID: REQ_FU_34	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Calcular tiempo
Descripción:	La herramienta debe calcular el tiempo de espera para el usuario hasta finalizar el análisis.

Tabla 35. REQ_FU_34

ID: REQ_FU_35	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Realizar análisis
Descripción:	La herramienta debe permitir ejecutar los siguientes requerimientos para analizar el código: REQ_FU_07, REQ_FU_11, REQ_FU_12, REQ_FU_14, REQ_FU_15, REQ_FU_16, REQ_FU_17, REQ_FU_18, REQ_FU_19, REQ_FU_20, REQ_FU_21, REQ_FU_22, REQ_FU_24, REQ_FU_25, REQ_FU_26, REQ_FU_27, REQ_FU_28, REQ_FU_29, REQ_FU_30, REQ_FU_31, REQ_FU_32, REQ_FU_33, REQ_FU_34

Tabla 36. REQ_FU_35

3.1.6.2 Requisitos no funcionales

Los requisitos no funcionales no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades de éste. También se pueden definir como las restricciones del sistema, la capacidad de los dispositivos de entrada/salida o la representación de datos que se utiliza en la interface del sistema.

ID: REQ_NF_01	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Botón Comprobar conexión
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_01.

Tabla 37- REQ_NF_01

ID: REQ_NF_02	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Botón Iniciar Sesión
Descripción:	La herramienta debe tener un botón que permita realizar los requisitos REQ_FU_02 y REQ_FU_03.

Tabla 38. REQ_NF_02

ID: REQ_NF_03	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Botón Cerrar Sesión
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_04.

Tabla 39. REQ_NF_03

ID: REQ_NF_04	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Botón Salir
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_05.

Tabla 40. REQ_NF_04

ID: REQ_NF_05	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Botón Iniciar Sesión
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_08.

Tabla 41. REQ_NF_05

ID: REQ_NF_06	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Botón Iniciar Sesión
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_09.

Tabla 42. REQ_NF_06

ID: REQ_NF_07	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Botón Iniciar Sesión
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_10.

Tabla 43. REQ_NF_07

ID: REQ_NF_08	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Botón Mostrar ayuda
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_13.

Tabla 44. REQ_NF_08

ID: REQ_NF_09	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar formato
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_14.

Tabla 45. REQ_NF_09

ID: REQ_NF_10	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar indentación
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_15.

Tabla 46. REQ_NF_10

ID: REQ_NF_11	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar comentarios
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_16.

Tabla 47. REQ_NF_11

ID: REQ_NF_12	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar saltos
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_17.

Tabla 48. REQ_NF_12

ID: REQ_NF_13	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar break
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_18.

Tabla 49. REQ_NF_13

ID: REQ_NF_14	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar return
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_19.

Tabla 50. REQ_NF_14

ID: REQ_NF_15	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar continue
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_20.

Tabla 51. REQ_NF_15

ID: REQ_NF_16	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar goto
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_21.

Tabla 52. REQ_NF_16

ID: REQ_NF_17	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar tamaño funciones
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_22.

Tabla 53. REQ_NF_17

ID: REQ_NF_18	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Caja Indicar tamaño funciones
Descripción:	La herramienta debe tener una caja de texto que permita al profesor indicar el requisito REQ_FU_23.

Tabla 54. REQ_NF_18

ID: REQ_NF_19	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar definición
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_25.

Tabla 55. REQ_NF_19

ID: REQ_NF_20	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar devolución funciones
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_26.

Tabla 56. REQ_NF_20

ID: REQ_NF_21	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar interfaz usuario
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_27.

Tabla 57. REQ_NF_21

ID: REQ_NF_22	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar función subprograma
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_28.

Tabla 58. REQ_NF_22

ID: REQ_NF_23	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar lecturas datos
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_29 .

Tabla 59. REQ_NF_23

ID: REQ_NF_24	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar variable no inicializada
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_30 .

Tabla 60. REQ_NF_24

ID: REQ_NF_25	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar asignaciones
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_31 .

Tabla 61. REQ_NF_25

ID: REQ_NF_26	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar variables sin inicializar
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_32 .

Tabla 62. REQ_NF_26

ID: REQ_NF_27	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	CHECK Comprobar variables globales
Descripción:	La herramienta debe tener un CHECK que permita al profesor seleccionar o no seleccionar el análisis del requisito REQ_FU_33 .

Tabla 63. REQ_NF_27

ID: REQ_NF_28	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja	Fuente: <input type="checkbox"/> Profesor <input checked="" type="checkbox"/> Alumno
Riesgo: <input type="checkbox"/> Alto <input type="checkbox"/> Medio <input checked="" type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Nombre:	Barra de carga para indicar tiempo
Descripción:	La herramienta debe tener una barra de carga que represente el tiempo calculado por el requisito REQ_FU_34 .

Tabla 64. REQ_NF_28

ID: REQ_NF_29	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: <input checked="" type="checkbox"/> Profesor <input type="checkbox"/> Alumno
Riesgo: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo	Verificabilidad: <input checked="" type="checkbox"/> Alto <input type="checkbox"/> Medio <input type="checkbox"/> Bajo
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Nombre:	Botón Realizar análisis
Descripción:	La herramienta debe tener un botón que permita realizar el requisito REQ_FU_35 .

Tabla 65. REQ_NF_29

3.1.7 Especificación de casos de uso

Una vez diferenciados y descritos los diferentes requisitos de la aplicación, se deben elaborar los distintos diagramas de casos de uso obtenidos de los requisitos definidos, así como una breve descripción de cada caso de uso para su mejor entendimiento.

En los casos de uso se especifica el funcionamiento del sistema en respuesta a una interacción de un usuario externo, por tanto no se tendrá en cuenta al propio sistema como actor. Estos constan de los siguientes elementos:

- **Actor.** Cual elemento externo ajeno al sistema y que realiza una interacción con él. El actor además puede tener distintos roles dentro del sistema, lo cual le llevará a realizar unas u otras acciones sobre el mismo.
- **Caso de Uso.** Son las distintas acciones que un actor puede realizar sobre un determinado sistema.

- **Relaciones.** Muestra la comunicación entre el actor, y los distintos casos de uso del sistema.

La herramienta desarrollada en este proyecto tiene los siguientes casos de uso:

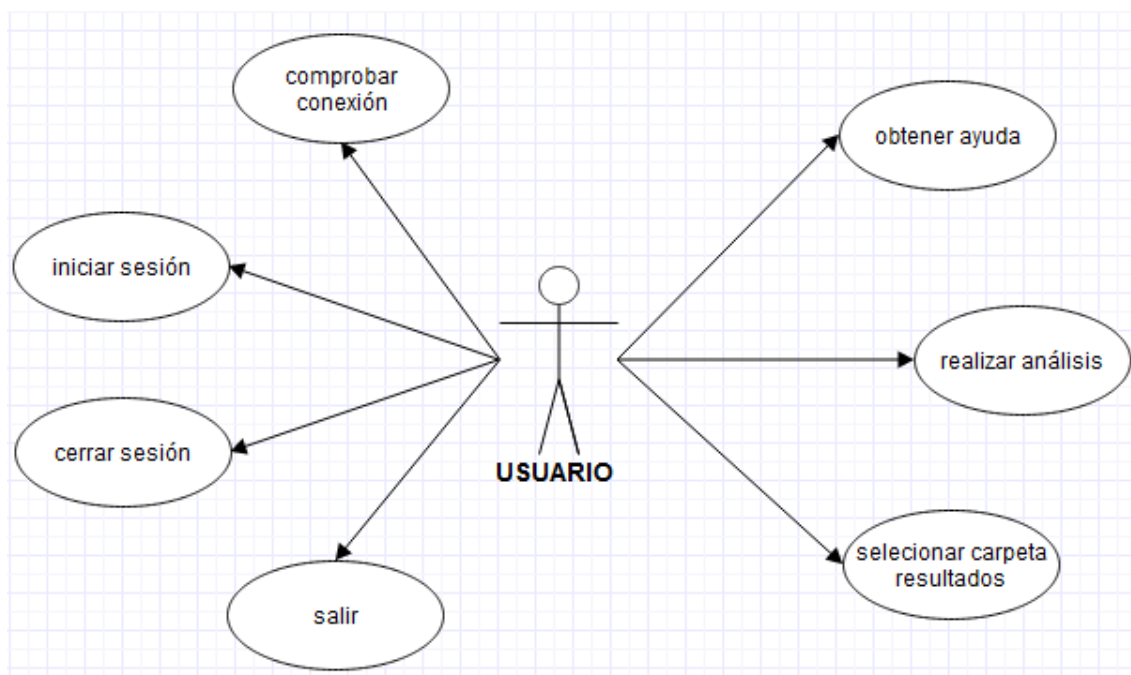


Ilustración 15. Diagrama de casos de uso para rol usuario

En el diagrama de casos de uso anterior se representan aquellas acciones que un usuario puede realizar sobre la herramienta, sin importar su rol dentro del sistema, ya que estas acciones pueden ser realizadas tanto por profesores como por alumnos.

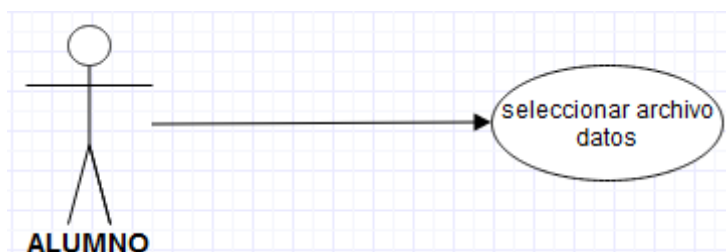


Ilustración 16. Diagrama de casos de Uso para rol Alumno

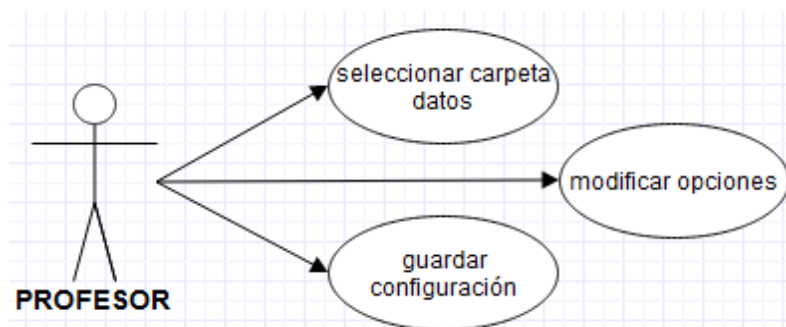


Ilustración 17. Diagrama de casos de Uso para rol Profesor

En las Ilustraciones 16 y 17, aparecen los roles de alumnos y profesor, indicando aquellas acciones que son exclusivas para sus roles dentro del sistema.

A continuación se desarrollaran las distintas tablas de cada caso de uso, para una mayor comprensión. Para su representación se utilizará una tabla descriptiva como la siguiente:

ID: CdU_NN	
Nombre:	
Actores:	
Objetivo:	
Precondiciones:	
Postcondiciones:	

Tabla 66. Plantilla de casos de uso

- **ID:** Identificador unívoco de cada caso de uso que tendrá el siguiente formato: CdU_NN, donde CdU identifica que es un caso de uso, y NN es un número de dos cifras diferenciador para cada caso de uso.
- **Nombre:** Hace referencia a un pequeño título aclarativo y sencillo de lo que trata el caso de uso, en concordancia con los diagramas anteriores.
- **Actores:** Son los usuarios que intervienen directamente en cada caso de uso.
- **Objetivo.** Se indica el objetivo de cada caso de uso.
- **Precondiciones:** Condiciones que deben darse previamente en el sistema para que pueda efectuarse el caso de uso.
- **Postcondiciones.** Representa como queda el sistema después de efectuarse el caso de uso.

ID: CdU_01	
Nombre:	Comprobar conexión
Actores:	Usuario (Profesor o alumno)
Objetivo:	Comprobar si se está conectado a la base de datos para la autenticación.
Precondiciones:	- Haber iniciado la aplicación.
Postcondiciones:	- Se muestra un mensaje indicando si se está o no conectado.

Tabla 67. CdU_01

ID: CdU_02	
Nombre:	Iniciar sesión
Actores:	Usuario (Profesor o alumno)
Objetivo:	Iniciar sesión y acceder a la ventana principal de la aplicación.
Precondiciones:	- Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos.
Postcondiciones:	- Se cargará la ventana principal de la aplicación correspondiente, en función de si el usuario tiene rol de profesor o de alumno.

Tabla 68. CdU_02

ID: CdU_03	
Nombre:	Cerrar sesión
Actores:	Usuario (Profesor o alumno)
Objetivo:	Cerrar sesión y volver a la pantalla de inicio de sesión.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos.
Postcondiciones:	<ul style="list-style-type: none"> - Se cargará la ventana de inicio de sesión de la aplicación.

Tabla 69. CdU_03

ID: CdU_04	
Nombre:	Salir
Actores:	Usuario (Profesor o alumno)
Objetivo:	Salir de la aplicación y cerrar la herramienta.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos.
Postcondiciones:	<ul style="list-style-type: none"> - Se pondrá fin a la ejecución de la aplicación.

Tabla 70. CdU_04

ID: CdU_05	
Nombre:	Obtener ayuda
Actores:	Usuario (Profesor o alumno)
Objetivo:	Acceder a un menú de ayuda o explicación de la herramienta.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos.
Postcondiciones:	<ul style="list-style-type: none"> - Se mostrará un menú con una pequeña ayuda describiendo los pasos a seguir para utilizar la aplicación.

Tabla 71. CdU_05

ID: CdU_06	
Nombre:	Seleccionar carpeta resultados
Actores:	Usuario (Profesor o alumno)
Objetivo:	Seleccionar la carpeta donde se creará el archivo que contendrá los resultados del análisis.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos.
Postcondiciones:	<ul style="list-style-type: none"> - Aparecerá en el cuadro de texto correspondiente la ruta seleccionada por el usuario para guardar el fichero con el resultado del análisis.

Tabla 72. CdU_06

ID: CdU_07	
Nombre:	Seleccionar archivo datos
Actores:	Alumno
Objetivo:	Seleccionar el archivo sobre el que se quiere realizar el análisis.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos. - Haber accedido a la aplicación con rol de alumno.
Postcondiciones:	<ul style="list-style-type: none"> - Aparecerá en el cuadro de texto correspondiente la ruta del archivo seleccionada por el alumno sobre el que se va a realizar el análisis.

Tabla 73. CdU_07

ID: CdU_08	
Nombre:	Seleccionar carpeta datos
Actores:	Profesor
Objetivo:	Seleccionar la carpeta que contiene el archivo o los archivos a analizar.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos. - Haber accedido a la aplicación con rol de profesor.
Postcondiciones:	<ul style="list-style-type: none"> - Aparecerá en el cuadro de texto correspondiente la ruta de la carpeta seleccionada por el profesor que contiene el archivo, o los archivos, sobre el que se va a realizar el análisis.

Tabla 74. CdU_08

ID: CdU_09	
Nombre:	Modificar opciones
Actores:	Profesor
Objetivo:	Permitir marcar o desmarcar los distintos CHECK de la ventana principal, para poder elegir que errores se quiere analizar.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos. - Haber accedido a la aplicación con rol de profesor.
Postcondiciones:	<ul style="list-style-type: none"> - Se marcarán o desmarcarán las distintas opciones de análisis.

Tabla 75. CdU_09

ID: CdU_10	
Nombre:	Guardar configuración
Actores:	Profesor
Objetivo:	Permitir guardar la configuración seleccionada en ese momento.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos. - Haber accedido a la aplicación con rol de profesor. - Haber seleccionado la carpeta que tendrá los ficheros a analizar. - Haber seleccionado la carpeta donde se creará el fichero que contendrá el resultado del análisis.
Postcondiciones:	<ul style="list-style-type: none"> - Se creará un fichero config.txt en la carpeta de la aplicación, que contendrá de forma codificada la configuración guardada.

Tabla 76. CdU_10

ID: CdU_11	
Nombre:	Realizar análisis
Actores:	Usuario (Profesor o alumno)
Objetivo:	Realizar el análisis sobre el o los archivos.
Precondiciones:	<ul style="list-style-type: none"> - Haber iniciado la aplicación. - Haber introducido un usuario y contraseña correctos. - Haber seleccionado la carpeta que tendrá el archivo, o los archivos, a analizar. - Haber seleccionado la carpeta donde se creará el fichero que contendrá el resultado del análisis.
Postcondiciones:	<ul style="list-style-type: none"> - Se realizará el análisis sobre el fichero, o los ficheros indicados, guardando dicho análisis en archivo en la carpeta que se haya indicado.

Tabla 77. CdU_11

3.2 Fase de Diseño e Implementación

La fase de diseño debe partir de la etapa de análisis anterior, y realizar un diseño coherente de los componentes, para que en la posterior etapa de implementación no haya que revisar y volver a plantear ningún punto. Para ello se propondrá un diseño que cumpla las necesidades requeridas para la aplicación y se definirán las diferentes clases necesarias, para representar de forma correcta la herramienta.

Hecho esto, se pasará a implementar dichas clases, partiendo de los requisitos y casos de uso expuestos en la fase de análisis, y se comentarán los distintos métodos para llevar a cabo cada una de las tareas de la aplicación.

3.2.1 Clase ventana_ppal

Esta clase representa la ventana inicial de la aplicación, desde ella se permite comprobar la conexión al usuario, así como iniciar sesión para acceder a la ventana principal de la herramienta. Para realizar su función, hace uso de una base de datos que contiene las diferentes tablas que permiten identificarse en la aplicación, y que además define los distintos roles de cada usuario, y se comentará más adelante en el apartado [3.2.2 Base de datos](#).

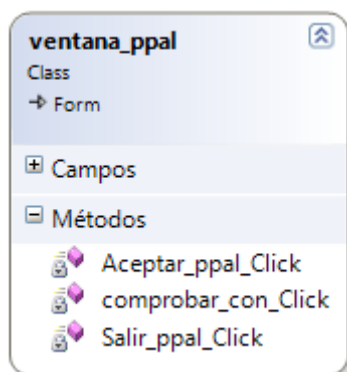


Ilustración 18. Clase ventana_ppal

La clase consta de los siguientes métodos para realizar sus funciones:

- **Aceptar_ppal_Click**. Mediante este método, se comprueba si el nombre de usuario y la contraseña introducidos son correctos. Realiza la función especificada en los requisitos **REQ_FU_02** y **REQ_FU_03** de la fase de análisis.

```
Private Sub Aceptar_ppal_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Aceptar_ppal.Click
    SqlConex = New MySqlConnection
    SqlConex.ConnectionString = "server=localhost;Port=3306;userid=db_user;password=db_password;database=data_pass; pe
    Dim lector_a As MySqlDataReader
    Dim lector_p As MySqlDataReader

    Try
        SqlConex.Open()
        Dim consulta_a As String

        consulta_a = "select * from data_pass.alum_pass where nombre_usuario='" & login_txt.Text & "' and pass='" & pa
```

Ilustración 19. Método aceptar_ppal_Click

Para realizar este método, la función se conectará al servidor indicado (en el caso de la imagen es localhost, ya que la conexión se hace de forma local), puede variar este desde una dirección IP del ordenador que tenga como función ser el servidor y que contendrá la base de datos, una dirección url o local host, para realizar la conexión de forma local.

Hecho esto se obtiene los campos usuario y contraseña introducidos por el usuario, y se comparan con las distintas tablas existentes en la base de datos, si se produce alguna coincidencia entre el par de datos introducido por el usuario y la base de datos, se dará acceso al usuario a la aplicación, en caso contrario se le mostrará un error por pantalla.

- **Comprobar con Click**. Este método permite al usuario comprobar, haciendo click en un botón, si se está o no conectado con la base de datos. Realiza la función especificada en el requisito **REQ_FU_01** de la fase de análisis.

```
Private Sub comprobar_con_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles comprobar_con.Click
    SqlConex = New MySqlConnection
    SqlConex.ConnectionString = "server=localhost;Port=3306;userid=db_user;password=db_password;database=data_pass; pers
    Try
        SqlConex.Open()
        MsgBox("Conexión establecida", MsgBoxStyle.Exclamation Or MsgBoxStyle.OkOnly, _
            "Comprobar conexión")
        SqlConex.Close()
    Catch ex As Exception
        MsgBox("Error en la conexión", MsgBoxStyle.Exclamation Or MsgBoxStyle.OkOnly, _
            "Comprobar conexión")
    Finally
        SqlConex.Dispose()
    End Try
End Sub
```

Ilustración 20. Método comprobar_con_click

La función de este método es conectarse a la base de datos indicada en el campo server, que como se indicó en el método anterior, puede variar desde una dirección IP del servidor, la dirección url del servidor, o localhost si la conexión se realiza de forma local.

En función de si la conexión se consigue o no, se mostrará al usuario un mensaje por pantalla indicándoselo.

- **Salir_ppal Click.** Este método permite al usuario salir de la aplicación, realizando una llamada a una función auxiliar que imprime un mensaje al usuario por pantalla, indicándole si realmente desea salir de la herramienta. Realiza la función especificada en el requisito **REQ_FU_05** de la fase de análisis.

```
Private Sub Salir_ppal_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Salir_ppal.Click
    Mensaje_Salir()
End Sub
```

Ilustración 21. Método salir_ppal_Click

3.2.2 Base de datos

Para realizar la identificación en la aplicación, se ha optado por utilizar una base de datos. Para ello se dispondrá de un servidor que debe ser creado con la herramienta MySQL Community Server, que permite crear un servidor de base de datos, donde se alojará la utilizada en la aplicación.

Además se necesitará de un gestor de base de datos, como puede ser MySQL WorkBench, que permita crear la base de datos, y las correspondientes tablas para realizar la identificación.

En la ilustración 23, aparece la creación de la base de datos correspondiente, la del usuario, la concesión de los permisos necesarios para poder acceder a la base de datos. Además de la creación de las diferentes tablas, una que tendrá aquellos usuarios con rol de profesor, y otra con aquellos usuarios con rol de alumno.

Por último se introducirán algunas tuplas en ambas tablas, para crear algunos usuarios con ambos roles, como puede observarse en la ilustración 22.

```
1 • SELECT * FROM data_pass.alum_pass;
┌──────────┬──────────┬──────────┐
│ nombre_usuario │ pass │
├──────────┬──────────┬──────────┤
│ alum1 │ 1234 │
├──────────┬──────────┬──────────┤
│ alum2 │ 1234 │
└──────────┬──────────┬──────────┘

1 • SELECT * FROM data_pass.prof_pass;
┌──────────┬──────────┬──────────┐
│ nombre_usuario │ pass │
├──────────┬──────────┬──────────┤
│ prof1 │ 1234 │
├──────────┬──────────┬──────────┤
│ prof2 │ 1234 │
└──────────┬──────────┬──────────┘
```

Ilustración 22. Tablas de ejemplo

```

1 • drop database data_pass;
2 • create database data_pass;
3 • drop user db_user;
4 • create user db_user;
5
6 • grant all on data_pass.* to 'db_user'@'localhost' identified by 'db_password';
7
8 • use data_pass;
9
10 • create table prof_pass(
11     nombre_usuario varchar(20) not null,
12     pass            varchar(20) not null);
13
14 • create table alum_pass(
15     nombre_usuario varchar(20) not null,
16     pass            varchar(20) not null);
17
18 • insert into prof_pass values ('prof1','1234');
19 • insert into prof_pass values ('prof2','1234');
20 • insert into alum_pass values ('alum1','1234');
21 • insert into alum_pass values ('alum2','1234');
22
23 • GRANT ALL PRIVILEGES ON data_pass.* TO 'db_user'@'%' IDENTIFIED BY 'db_password';
24 • FLUSH PRIVILEGES;

```

Ilustración 23. Creación base de datos y tablas

3.2.3 Clase ventana_ayuda

Esta clase representa la ventana de ayuda accesible desde la ventana principal de la aplicación, independientemente del rol con el que el usuario haya accedido a la herramienta.

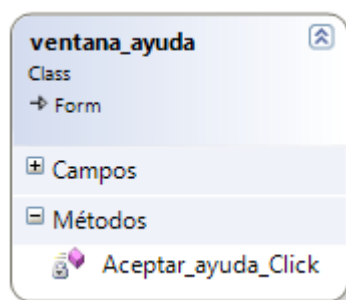


Ilustración 24. Clase ventana_ayuda

La clase consta únicamente del siguiente método:

- **Aceptar ayuda Click**. Mediante este método, se cierra la ventana de ayuda de la aplicación, haciendo visible de nuevo la ventana principal de la herramienta y escondiendo la ventana de ayuda. Realiza la función especificada en el requisito **REQ_FU_13** de la fase de análisis.

```

Private Sub Aceptar_ayuda_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Aceptar_ayuda.Click
    Me.Hide()
End Sub

```

Ilustración 25. Método aceptar_ayuda_click

3.2.4 Clase ventana_prof

Esta clase representa la ventana principal de la herramienta, siempre y cuando el usuario haya iniciado sesión con rol de profesor.

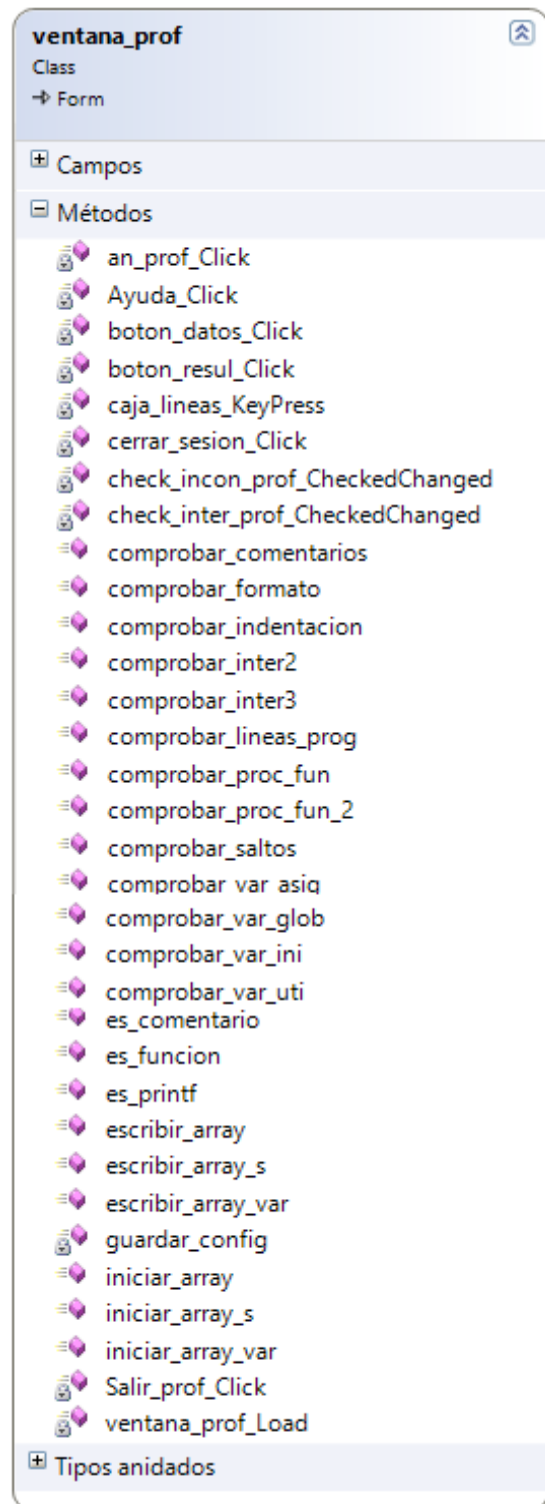


Ilustración 26. Clase ventana_prof

A continuación se describirán los métodos más importantes de esta clase.

- **Salir_prof Click.** Permite al usuario salir de la aplicación pulsando el botón correspondiente. Realiza la función especificada en el requisito **REQ_FU_05** de la fase de análisis.

```
Private Sub Salir_prof_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Salir_prof.Click
    ventana_ppal.Mensaje_Salir()
End Sub
```

Ilustración 27. Método salir_prof_click

Su función es llamar al método que creado en la clase ventana_ppal, que indica al usuario si desea o no salir de la aplicación.

- **Cerrar sesión Click.** Permite al usuario cerrar la sesión abierta en la aplicación pulsando el botón correspondiente. Realiza la función especificada en el requisito **REQ_FU_04** de la fase de análisis.

```
Private Sub cerrar_sesion_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cerrar_sesion.Click
    If MsgBox("¿Está seguro que desea cerrar la sesión?", MsgBoxStyle.Exclamation Or MsgBoxStyle.YesNo, "Cerrar Sesión") = MsgBoxResult.Yes Then
        Me.Close()
        ventana_ppal.Show()
    End If
End Sub
```

Ilustración 28. Método cerrar_sesión_click

Una vez se pulse el botón correspondiente de la aplicación, se mostrará un mensaje al usuario preguntándose si realmente desea cerrar la sesión, en función de la respuesta elegida, se cerrará el mensaje, o se volverá a la pantalla de inicio de sesión, cerrándose la ventana actual y perdiendo la sesión iniciada.

- **Botón datos click.** Método que permite al usuario, al pulsar el botón correspondiente, cargar la carpeta que contendrá los archivos a analizar. Realiza la función especificada en el requisito **REQ_FU_09** de la fase de análisis.

```
Private Sub boton_datos_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles boton_datos.Click
    If carpeta_datos.ShowDialog = DialogResult.OK Then
        carp_datos.Text = carpeta_datos.SelectedPath
    End If
End Sub
```

Ilustración 29. Método botón_datos_click

Una vez el usuario pulse el botón correspondiente, aparecerá un cuadro de diálogo permitiendo al usuario seleccionar la carpeta que contiene los archivos a tratar. Una vez seleccionada la carpeta, aparecerá su ruta en el cuadro de texto creado para ello.

- **Botón resul click.** Método que permite al usuario, al pulsar el botón correspondiente, indicar la carpeta en la cual se creará el archivo donde se escribirán los resultados del análisis. Realiza la función especificada en el requisito **REQ_FU_10** de la fase de análisis.

```
Private Sub boton_resul_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles boton_resul.Click
    If carpeta_resultados.ShowDialog = DialogResult.OK Then
        carp_resul.Text = carpeta_resultados.SelectedPath
    End If
End Sub
```

Ilustración 30. Método botón_resul_click

Una vez el usuario pulse el botón correspondiente, aparecerá un cuadro de diálogo permitiendo al usuario seleccionar la carpeta donde se creará el archivo con los resultados del

análisis. Una vez seleccionada la carpeta, aparecerá su ruta en el cuadro de texto creado para ello.

- **Caja líneas keypress.** Método encargado de controlar que la información introducida en el cuadro de texto existente para indicar la longitud de las funciones es correcta. Realiza la función especificada en el requisito **REQ_FU_23** de la fase de análisis.

```
Private Sub caja_lineas_KeyPress(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles caja_lineas.KeyPress
    If (Char.IsLetter(e.KeyChar)) Then
        e.Handled = True
    ElseIf (Char.IsSymbol(e.KeyChar)) Then
        e.Handled = True
    ElseIf (Char.IsPunctuation(e.KeyChar)) Then
        e.Handled = True
    End If
    caja_lineas.MaxLength = 2
End Sub
```

Ilustración 31. Método caja_lineas_keypress

Comprueba que el valor introducido en el campo correspondiente es un número, y no cualquier otro carácter, y además controla que la longitud del valor sea como mucho de 2 cifras.

- **Ayuda Click.** Mediante este método, al pulsar en el botón correspondiente, se muestra una ventana de ayuda, que indica al usuario los pasos a seguir para usar la herramienta, y la explicación de cada error analizado por la aplicación. Realiza la función especificada en el requisito **REQ_FU_13** de la fase de análisis.

```
Private Sub Ayuda_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Ayuda.Click
    ventana_ayuda.Show()
End Sub
```

Ilustración 32. Método ayuda_click

El método tiene por función invocar la función de la clase ayuda, que muestra la ventana de ayuda en la pantalla al usuario.

- **Ventana prof load.** Este método tiene como función realizar las distintas tareas que la ventana debe hacer cuando es cargada. Realiza la función especificada en el requisito **REQ_FU_06** de la fase de análisis.

```
Private Sub ventana_prof_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Const fichero As String = ".\config.txt"
    Dim cont As Integer = 65
    If My.Computer.FileSystem.FileExists(fichero) Then
        Dim lector As New IO.StreamReader(fichero)
        While lector.Peek() <> -1
            Dim linea As String = lector.ReadLine()
            '---'
        End While
    End If
End Sub
```

Ilustración 33. Método ventana_prof_load

Este método tiene varias tareas que realizar, la primera es buscar si en la carpeta donde se encuentra la aplicación existe un fichero “config.txt”. Este fichero contiene una configuración que haya sido guardada anteriormente, y además está codificado para que no se pueda modificar. Por tanto, el método debe encargarse de decodificar de nuevo el fichero, y cargar la configuración correspondiente, cargando una configuración por defecto en caso de no encontrarse dicho fichero en la carpeta.

Además mostrará en la ventana de la aplicación distintas tooltip ^[14], que son etiquetas aclaratorias que muestran información al usuario al situar el curso sobre algunos elementos gráficos de la propia ventana.

- **Guardar Config.** Este método tiene como función guardar la configuración seleccionada en el momento de pulsar el botón de realizar el análisis. Realiza la función especificada en el requisito **REQ_FU_07** de la fase de análisis.

```
Private Sub guardar_config()
    Const fichero As String = ".\config.txt"
    Dim cont As Integer = 1
    Dim var As Integer = 2
    Dim a As New System.IO.StreamWriter(fichero)
    For i As Integer = 1 To 300
        Dim value As Integer = CInt(Int((2 * Rnd()) + 1))
        a.Write(value)
        If i = 149 Then
```

Ilustración 34. Método guardar_config

La función crea el fichero “config.txt” en la carpeta del programa, y va recorriendo cada CHECK de la herramienta, guardando si está marcado o desmarcado, además de guardar el valor del número máximo de líneas que puede tener una función. Para finalizar, se codificará el fichero para que no pueda ser modificado.

- **Comprobar formato.** Este método tiene como función comprobar que los nombres de variables, constantes, procedimientos y funciones sigan un formato determinado a lo largo de todo el programa. Realiza la función especificada en el requisito **REQ_FU_14** de la fase de análisis.

```
Function comprobar_formato(ByVal archi As String, ByVal palabra As String, ByVal array As Integer())
    Dim gui_bajo_enc As Integer = 0
    Dim gui_enc As Integer = 0
    Dim linea_gui As Integer = 0
    Dim linea_gui_b As Integer = 0
    Dim cont() As Integer = {0, 0, 0}
    Dim form_enc As Integer = 0
    Dim lector As New IO.StreamReader(archi)
```

Ilustración 35. Método comprobar_formato

El método va buscando línea a línea, en el archivo pasado como parámetro en la función, si hay una declaración de una variable, constante, función o procedimiento, en caso de ser así se guarda su formato, según contenga una letra mayúscula, un guion bajo (_), o un guion normal (-). Se va recorriendo el fichero hasta finalizar, devolviendo la función un array indicando si hay o no alguna declaración de cada tipo de formato posible, y un array que contiene la línea o líneas donde aparecen las declaraciones con formato erróneo.

- **Comprobar indentación.** Este método tiene como función comprobar que el código esté correctamente indentado, según la profundidad de cada instrucción dentro del conjunto de bloques, mediante el uso de tabuladores. Realiza la función especificada en el requisito **REQ_FU_15** de la fase de análisis.

```
Function comprobar_indentacion(ByVal archi As String, ByVal array As Integer())
    Dim com, com2 As Integer
    Dim enc_act As Integer = 0
    Dim enc_ant As Integer = 0
    Dim num_tab_1 As Integer = 0
    Dim num_tab_2 As Integer = 0
```

Ilustración 36. Método comprobar_indentación

El método va buscando línea a línea por el fichero pasado por parámetro a la función, la existencia de bloque de instrucciones, y cada vez que se encuentre un bloque de instrucciones nuevo, este deberá estar más tabulado que el anterior. También comprueba que cada vez que se cierre un bloque de instrucciones, deberá estar menos tabulado que la línea anterior. Generalmente los bloques se abrirán con el carácter “{” y se cerrarán con el carácter “}”.

El método devuelve valor indicando si el código está o no bien indentado, y un array que contiene la línea o líneas donde el código no está bien indentado.

- **Comprobar comentarios.** Este método tiene como función comprobar que el código esté comentado. Se deben incluir comentarios la línea antes de cada función, y en caso de no existir función alguna, debe existir algún comentario en el código. Realiza la función especificada en el requisito **REQ_FU_16** de la fase de análisis.

```
Function comprobar_comentarios(ByVal archi As String, ByVal array As Integer())
    Dim porc_com As Integer
    Dim com, com2 As Integer
    Dim lector As New IO.StreamReader(archi)
    Dim fun_enc As Integer = 0
    Dim enc = 0
    Dim func_com As Integer = 0
    Dim cont_lineas As Integer = 0
```

Ilustración 37. Método comprobar_comentarios

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, buscando la existencia de funciones en el mismo, en caso de encontrar una, se comprueba que la línea justamente anterior tenga un comentario. Si no hay ninguna función en el código, debe existir al menos un comentario en el mismo. El método devolverá un valor en función de si el código está o no comentado, y un array que contiene la línea o líneas donde existan funciones sin comentar.

- **Comprobar saltos.** Este método tiene como función comprobar que no se haga uso de sentencias de salto incondicional en el código (BREAK, RETURN, GOTO, CONTINUE). Realiza la función especificada en el requisito **REQ_FU_18**, **REQ_FU_19**, **REQ_FU_20**, **REQ_FU_21** de la fase de análisis.

```
Function comprobar_saltos(ByVal archi As String, ByVal palabra As String, ByVal array As Integer())
    Dim saltos_enc As Integer
    Dim lector As New IO.StreamReader(archi)
    Dim cont_lineas As Integer = 0
    Dim pal_enc As Integer = 0
    saltos_enc = 0
    While lector.Peek() <> -1
```

Ilustración 38. Método comprobar_saltos

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, buscando la existencia de la sentencia de salto incondicional indicada en el parámetro palabra, siempre que esta no esté comentada. El método devolverá un valor en función de si en el

código aparece o no la sentencia de salto condicional correspondiente, y un array que contiene la línea o líneas donde aparezcan dichas sentencias.

- **Comprobar líneas prog.** Este método tiene como función controlar el tamaño máximo de las funciones y procedimientos. Realiza la función especificada en el requisito **REQ_FU_22** de la fase de análisis.

```
Function comprobar_lineas_prog(ByVal archi As String, ByVal array As Integer())
    Dim max_lineas As Integer = 0
    Dim com, com2 As Integer
    Dim lector As New IO.StreamReader(archi)
    Dim fun_enc As Integer = 0
    Dim enc = 1
```

Ilustración 39. Método comprobar_lineas_prog

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, buscando las distintas funciones o procedimientos que aparezcan en el código. Cuenta el número de líneas de cada subprograma encontrado y compara esas líneas con el tamaño máximo de las funciones indicado en el cuadro de texto correspondiente. El método devolverá un valor en función de si en el código hay subprogramas que exceden el tamaño máximo permitido, y un array que contiene la línea o líneas donde tienen la cabecera dichas funciones.

- **Comprobar proc fun.** Este método tiene como función controlar los argumentos de un subprograma, ya que estos pueden ser pasados por valor o referencia, en función de si se va a querer modificar su contenido dentro de la subrutina o no. Se controlarán los siguientes aspectos:
 - o No se debe modificar una variable pasada por valor.
 - o Los argumentos pasados por referencia deben ser modificados.

Realiza la función especificada en el requisito **REQ_FU_25** de la fase de análisis.

```
Function comprobar_proc_fun(ByVal archi As String, ByVal array As Integer())
    Dim comp_f_p As Integer = 0
    Dim llave_a As Integer = 0
    Dim llave_c As Integer = 0
    Dim com, com2 As Integer
    Dim enc = 0
    Dim param As String = ""
    Dim param_2 As String = ""
    ...
```

Ilustración 40. Método comprobar_proc_fun

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, buscando las distintas funciones y procedimientos que aparezcan en el código. Una vez encontrado un subprograma, se guardarán los distintos parámetros que tiene, y en función de si están pasados por valor (definidos normal) o referencia (la variable tendrá un * delante), podrán o no ser modificados. El método devolverá un valor en función de si en el código hay subprogramas con los parámetros mal definidos o no, y un array que contiene la línea o líneas donde tienen la cabecera dichas funciones.

- **Comprobar_proc_fun_2.** Este método tiene como función controlar, partiendo de que aquellos subprogramas que devuelvan un valor serán funciones y el resto serán procedimientos, se cumpla que:
 - o Toda función debe tener un valor de retorno.
 - o Los procedimientos no pueden tener valor de retorno.

Realiza la función especificada en el requisito **REQ_FU_26** de la fase de análisis.

```
Function comprobar_proc_fun_2(ByVal archi As String, ByVal array As Integer())
    Dim comp_f_p As Integer = 0
    Dim llave_a As Integer = 0
    Dim llave_c As Integer = 0
    Dim com, com2 As Integer
    Dim fun_v_enc As Integer = 0
```

Ilustración 41. Método comprobar_proc_fun_2

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, buscando los distintos subprogramas que aparezcan en el código. Una vez encontrado un subprograma, se comprueba si es un procedimiento (void) o una función (cualquier otra definición). Hecho esto se comprueba que los parámetros no devuelven ninguna variable, y que las funciones sí que devuelven algo, para que ambos estén bien definidos. El método devolverá un valor en función de si en el código hay subprogramas que devuelven mal los datos o no, y un array que contiene la línea o líneas donde tienen la cabecera dichas funciones.

- **Comprobar_inter2.** Este método tiene como función comprobar que un subprograma que calcule o modifique un dato nunca deberá imprimir por pantalla, y viceversa. Realiza la función especificada en el requisito **REQ_FU_28** de la fase de análisis.

```
Function comprobar_inter2(ByVal archi As String, ByVal array As Integer())
    Dim sal As Integer = 0
    Dim com, com2 As Integer
    Dim lector As New IO.StreamReader(archi)
    Dim fun_enc As Integer = 0
    Dim enc = 1
```

Ilustración 42. Método comprobar_inter2

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, buscando los distintos subprogramas que aparezcan en el código. Una vez encontrado un subprograma, se comprueba que si modifica un dato, no muestre mensajes por pantalla, y viceversa. El método devolverá un valor en función de si en el código hay subprogramas que tienen tareas que mezclan la lógica del programa con la interfaz de usuario, y un array que contiene la línea o líneas donde tienen la cabecera dichas funciones.

- **Comprobar_inter3.** Este método tiene como función comprobar que siempre que se pida un dato por teclado, deberá haber un mensaje previo de petición del dato. Realiza la función especificada en el requisito **REQ_FU_29** de la fase de análisis.

```
Function comprobar_inter3(ByVal archi As String, ByVal array As Integer())
    Dim com As Integer
    Dim lector As New IO.StreamReader(archi)
    Dim lect_datos() As String = {"scanf", "gets", "fgets"}
    Dim print_enc As Integer = 0
    Dim lect_enc As Integer = 0
    Dim ...
```

Ilustración 43. Método comprobar_inter3

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, buscando peticiones de datos al usuario, siempre que encuentra una comprueba que en la línea anterior se ha mostrado un mensaje por pantalla pidiendo dicho dato. El método devolverá un valor en función de si en el código hay peticiones de datos sin un mensaje por pantalla indicándolas, y un array que contiene la línea o líneas donde aparecen dichas peticiones erróneas en caso de existir.

- **Comprobar var ini.** Este método tiene como función comprobar que no existan variables sin inicializar en el código. Realiza la función especificada en el requisito **REQ_FU_32** de la fase de análisis.

```
Function comprobar_var_ini(ByVal archi As String, ByVal array As Integer())
    Dim comp_f_p As Integer = 0
    Dim llave_a As Integer = 0
    Dim llave_c As Integer = 0
    Dim com, com2, com3, com4 As Integer
    Dim cont_it As Integer = 0
    Dim enc2 As Integer = 0
```

Ilustración 44. Método comprobar_var_ini

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, y va buscando definiciones de variables. Una vez encontrada una variable, se guarda el número de veces que la variable es usada, y el número de veces que la variable es inicializada. Si la variable es usada pero no se ha inicializado anteriormente, esta variable estará sin inicializar.

El método devolverá un valor en función de si en el código hay variables sin inicializar, y un array que contiene la línea o líneas donde aparecen dichas variables en caso de existir.

- **Comprobar var asig.** Este método tiene como función comprobar que no existan dos asignaciones de una variable sin hacer uso de la misma entre ellas. Realiza la función especificada en el requisito **REQ_FU_31** de la fase de análisis.

```
Function comprobar_var_asig(ByVal archi As String, ByVal array As Integer())
    Dim comp_f_p As Integer = 0
    Dim llave_a As Integer = 0
    Dim llave_c As Integer = 0
    Dim com, com2, com3, com4 As Integer
    Dim ...
```

Ilustración 45. Método comprobar_var_asig

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, y va buscando definiciones de variables. Por cada variable, va comprobando las asignaciones y los usos que se hace de ella, y en el momento en el que se encuentra dos asignaciones seguidas de la misma variable, se produce el error. El método devolverá un valor en función de si en el

código variables con asignaciones seguidas sin hacer uso de la misma, y un array que contiene la línea o líneas donde aparecen dichas variables en caso de existir.

- **Comprobar var uti.** Este método tiene como función comprobar que no existan variables que no se usen nunca en el código. Realiza la función especificada en el requisito **REQ_FU_30** de la fase de análisis.

```
Function comprobar_var_uti(ByVal archi As String, ByVal array As Integer())
    Dim comp_f_p As Integer = 0
    Dim llave_a As Integer = 0
    Dim llave_c As Integer = 0
    Dim com, com2, com3, com4 As Integer
    Dim cont_it As Integer = 0
    Dim enc2 As Integer = 0
```

Ilustración 46. Método comprobar_var_uti

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, y va buscando definiciones de variables. Una vez encontrada una variable, se guarda el número de veces que la variable es usada. Si su número de usos es 0, la variable estará sin usar, y por tanto se habrá cometido un error.

El método devolverá un valor en función de si en el código hay variables sin usar, y un array que contiene la línea o líneas donde aparecen dichas variables en caso de existir.

- **Comprobar var glob.** Este método tiene como función comprobar que no existan variables globales en el código. Realiza la función especificada en el requisito **REQ_FU_33** de la fase de análisis.

```
Function comprobar_var_glob(ByVal archi As String, ByVal palabra As String, ByVal array As Integer())
    Dim llave_a As Integer = 0
    Dim llave_c As Integer = 0
    Dim com, com2 As Integer
    Dim glob_enc As Integer = 0
    Dim cont_lineas As Integer = 0
    Dim lector As New IO.StreamReader(archi)
    While lector.Peek() <> -1
```

Ilustración 47. Método comprobar_var_glob

Este método va recorriendo el archivo pasado por parámetro a la función línea a línea, y va buscando definiciones de variables, si dicha definición no se encuentra dentro de un subprograma o del programa principal, significa que es una variable global. El método devolverá un valor en función de si en el código hay variables globales, y un array que contiene la línea o líneas donde aparecen dichas variables en caso de existir.

- **An_prof_Click.** Este método es la principal de la clase, y realiza llamadas a la mayor parte de las funciones descritas anteriormente. Realiza la función especificada en el requisito **REQ_FU_35** de la fase de análisis.

```
Private Sub an_prof_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles an_prof.Click
    If carp_datos.Text = "" Or carp_resul.Text = "" Then
        MsgBox("Debe seleccionar la carpeta de donde se obtendrán los datos, y la carpeta donde se guardarán",
            MsgBoxStyle.Exclamation, "Atención")
    ElseIf caja_lineas.Text < 5 Or caja_lineas.Text > 99 Then
        MsgBox("El número de líneas solo permite valores comprendidos entre 5 y 99", MsgBoxStyle.Critical, "E")
    ElseIf carp_datos.Text <> "" And carp_resul.Text <> "" And caja_lineas.Text > 4 And caja_lineas.Text < 99
        guardar_config()
    Dim lista As New List(Of String)
```

Ilustración 48. Método An_prof_Click

Este método se acciona cuando se pulsa el botón “Analizar” de la ventana, y lo primero que hace es comprobar que se han seleccionado tanto la ruta de los archivos a analizar, como la ruta donde del archivo donde se escribirá el análisis. De no ser así se mostrará por pantalla un mensaje de error.

Después comprueba que el valor introducido como tamaño de las funciones y los procedimientos está comprendido entre 5 y 99, en caso contrario mostrará un error.

Hecho esto, hace el cálculo total de lo que tardará en ejecutarse el análisis para mostrar el avance en la correspondiente barra de carga.

Por último, se tomarán aquellos archivos “*.c” existentes en la ruta de archivos a analizar, y en función de si los CHECK del programa están marcados o no, se enviará cada archivo a cada función, escribiendo los resultados del análisis en el archivo correspondiente. Y para finalizar se mostrará por pantalla un mensaje indicando la finalización del análisis.

3.2.5 Clase ventana_alum

Esta clase representa la ventana principal de la herramienta, siempre y cuando el usuario haya iniciado sesión con rol de alumno.

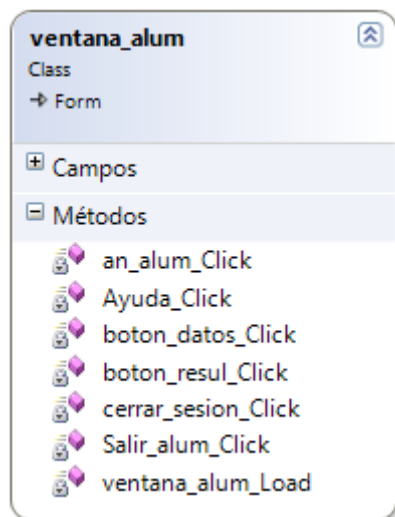


Ilustración 49. Clase ventana_alum

De esta clase no se comentará ningún método en especial, ya que tiene los mismos que los recogidos en el punto [3.2.4 Clase ventana_prof](#).

Capítulo 4. Resultados Obtenidos

En este apartado se recogerán las diferentes pruebas realizadas para comprobar el correcto funcionamiento de la aplicación. Para la representación de dichas pruebas se utilizarán Casos de prueba mediante una tabla descriptiva como la siguiente:

ID: CdP_NN	
Nombre:	
Descripción:	

Tabla 78. Plantilla de Casos de Prueba

- **ID:** Identificador único de cada caso de prueba que tendrá el siguiente formato: CdP_NN, donde CdP identifica que es un caso de prueba, y NN es un número de dos cifras diferenciador para cada caso de prueba.
- **Nombre:** Hace referencia a un pequeño título aclarativo y sencillo de lo que trata el caso de prueba.
- **Descripción:** Un pequeño comentario sobre lo que trata el caso de prueba.
- Un apartado extra para indicar la salida del programa.

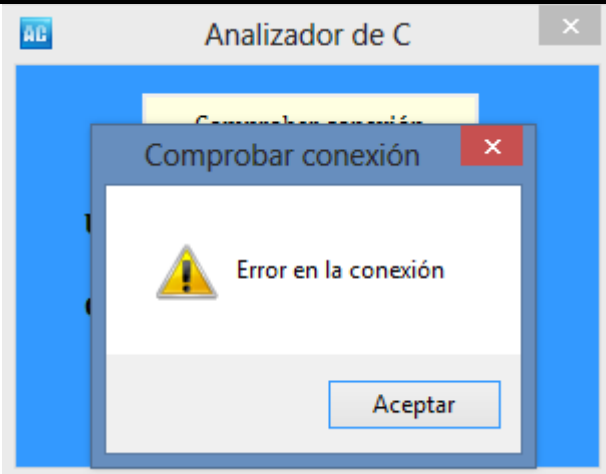
ID: CdP_01	
Nombre:	Comprobar conexión errónea
Descripción:	Se comprobará si se está conectado al servidor sin estar este activado.
	

Tabla 79. CdP_01

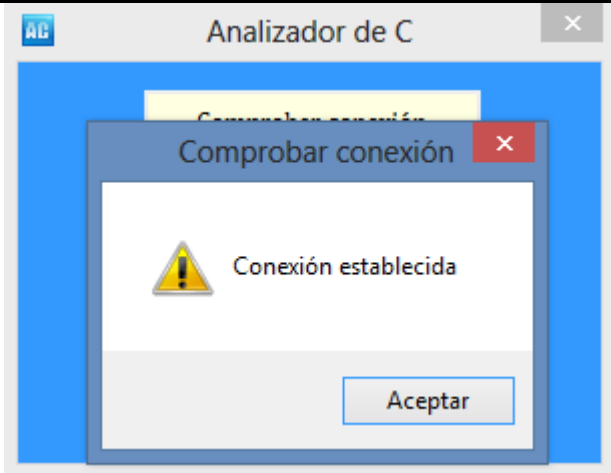
ID: CdP_02	
Nombre:	Comprobar conexión correcta
Descripción:	Se comprobará si se está conectado al servidor estando este activado.
	

Tabla 80. CdP_02

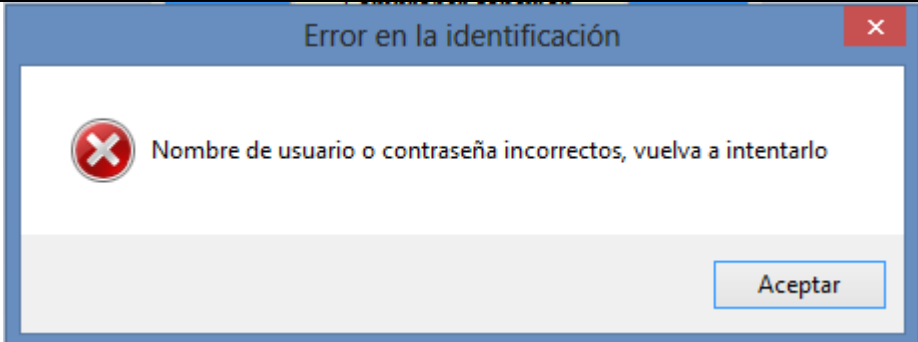
ID: CdP_03	
Nombre:	Identificación errónea
Descripción:	Intentar acceder a la aplicación con un nombre de usuario o contraseña incorrectos.
	

Tabla 81. CdP_03

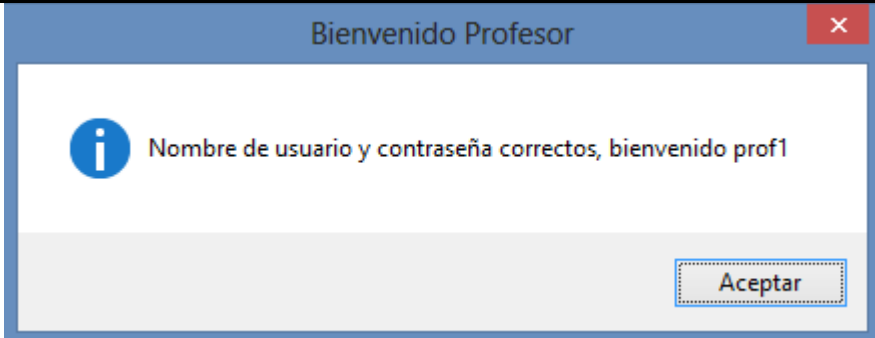
ID: CdP_04	
Nombre:	Identificación correcta profesor
Descripción:	Acceder a la aplicación con un usuario con rol de profesor.
	

Tabla 82. CdP_04

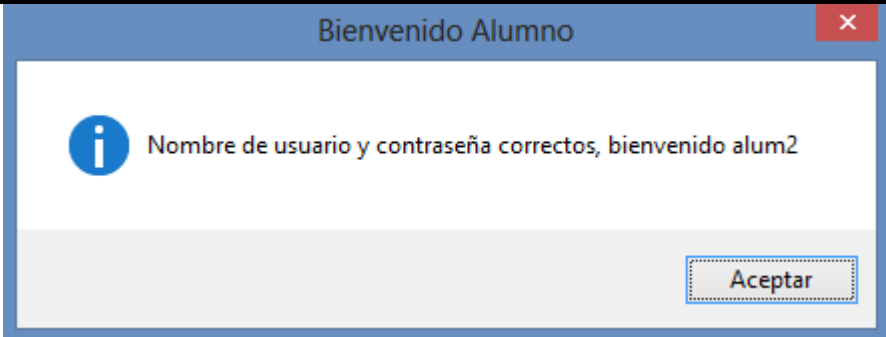
ID: CdP_05	
Nombre:	Identificación correcta alumno
Descripción:	Acceder a la aplicación con un usuario con rol de alumno.
	

Tabla 83. CdP_05

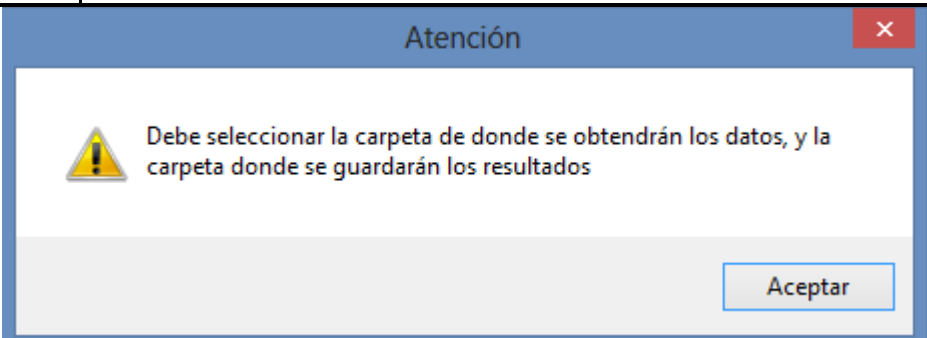
ID: CdP_06	
Nombre:	Ruta de archivos o de resultados vacía
Descripción:	Intentar realizar el análisis sin haber escogido la carpeta de datos, o la carpeta donde se guardará el archivo de resultados.
	

Tabla 84. CdP_06

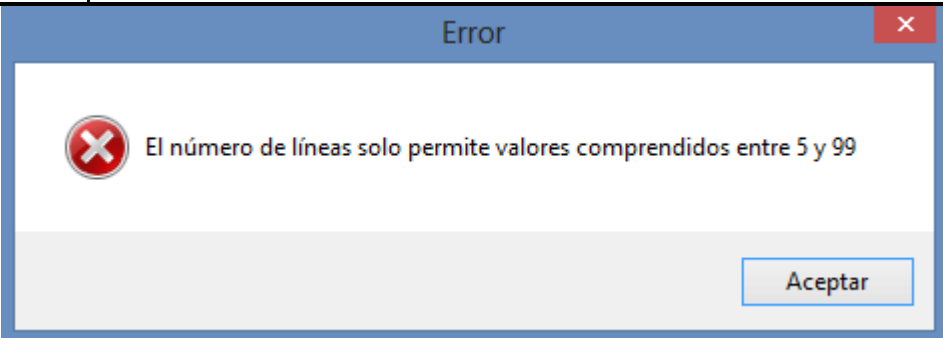
ID: CdP_07	
Nombre:	Tamaño fuera de rango
Descripción:	Introducir un tamaño para las funciones fuera de rango.
	

Tabla 85. CdP_07

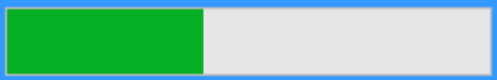
ID: CdP_08	
Nombre:	Barra de carga
Descripción:	Funcionamiento barra de carga.
	

Tabla 86. CdP_08

ID: CdP_09	
Nombre:	Ejemplo de formato incorrecto en el código
Descripción:	<pre> 3 int main () { 4 int num-1; 5 int num_2; 6 int Num3 = 0; </pre> <p>El código contiene variables, constantes o funciones declaradas con distinto formato (guion, guion bajo y palabras mayúsculas).</p> <p>Línea: 4</p> <p>Línea: 5</p> <p>Línea: 6</p>

Tabla 87. CdP_09

ID: CdP_10	
Nombre:	Ejemplo de formato correcto en el código
Descripción:	<pre> 3 int main () { 4 int num_1; 5 int num_2; </pre> <p>El código tiene funciones, procedimientos y variables declaradas con formato correcto.</p>

Tabla 88. CdP_10

ID: CdP_11	
Nombre:	Ejemplo de no seleccionar CHECK formato correcto en el código
Descripción:	<p>Legibilidad</p> <p><input type="checkbox"/> Nombres de variables, constantes, procedimientos y funciones, deben permanecer invariables</p> <p>No se ha comprobado el formato de las variables, constantes, funciones y procedimientos.</p>

Tabla 89. CdP_11

ID: CdP_12	
Nombre:	Ejemplo de indentación incorrecta en el código
Descripción:	<pre> 4 for (x = 1;x<N;x++) { 5 per[x] = pedir_datos(); 6 }</pre> <p>El código está mal tabulado o indentado.</p> <p>Línea: 5</p>

Tabla 90. CdP_12

ID: CdP_13	
Nombre:	Ejemplo de indentación correcta en el código
Descripción:	<pre> for (x = 1;x<N;x++) { per[x] = pedir_datos(); }</pre> <p>El código está bien tabulado o indentado.</p>

Tabla 91. CdP_13

ID: CdP_14	
Nombre:	Ejemplo de no seleccionar CHECK indentación
Descripción:	<input type="checkbox"/> Comprobar la indentación del código (tabuladores)
No se ha comprobado la indentación del código.	

Tabla 92. CdP_14

ID: CdP_15	
Nombre:	Ejemplo de código mal comentado
Descripción:	<pre> 3 #define N 3 4 5 void fun (int *n){ 6 printf("Introduzca un numero: ");</pre> <p>El código está mal comentado.</p> <p>Línea: 5</p>

Tabla 93. CdP_15

ID: CdP_16	
Nombre:	Ejemplo de código bien comentado
Descripción:	<pre> 5 //Esta función lee un número por teclado 6 void fun (int *n){ 7 printf("Introduzca un numero: ");</pre> <p>El código está bien comentado.</p>

Tabla 94. CdP_16

ID: CdP_17	
Nombre:	Ejemplo de no seleccionar CHECK de comentarios
Descripción:	<input type="checkbox"/> Comprobar el uso de comentarios en el código
No se ha comprobado si el código está comentado.	

Tabla 95. CdP_17

ID: CdP_18	
Nombre:	Ejemplo de no seleccionar CHECK de saltos incondicionales
Descripción:	<input type="checkbox"/> Comprobar el uso de sentencias incondicionales <input checked="" type="checkbox"/> break <input checked="" type="checkbox"/> return <input checked="" type="checkbox"/> continue <input checked="" type="checkbox"/> goto
No se ha comprobado el uso de saltos incondicionales.	

Tabla 96. CdP_18

ID: CdP_19	
Nombre:	Ejemplo de código con sentencia RETURN
Descripción:	<pre> 14 printf("El valor del numero introducido es %d.\n",x); 15 return 0; 16 }</pre>
<p>Se ha encontrado la instrucción <u>return</u> en código. Apariciones: 1</p> <p>Línea: 15</p>	

Tabla 97. CdP_19

ID: CdP_20	
Nombre:	Ejemplo de código sin sentencia RETURN
Descripción:	<pre> 14 printf("El valor del numero introducido es %d.\n",x); 15 //return 0; 16 }</pre>
<p>No se ha encontrado la instrucción <u>return</u> en código.</p>	

Tabla 98. CdP_20

ID: CdP_21	
Nombre:	Ejemplo de no seleccionar CHECK de return
Descripción:	<input checked="" type="checkbox"/> Comprobar el uso de sentencias incondicionales <input checked="" type="checkbox"/> break <input type="checkbox"/> return
<p>No se ha encontrado la instrucción break en código.</p> <p>No se ha comprobado la aparición de la sentencia <u>return</u> en el código.</p>	

Tabla 99. CdP_21

ID: CdP_22	
Nombre:	Ejemplo de código con función muy extensa
Descripción:	<pre> 6 void fun (int *n, int *m, int *p){ 7 printf("Introduzca el numero 1: "); 8 scanf("%d",n); 9 printf("Introduzca el numero 2: "); 10 scanf("%d",m); 11 printf("Introduzca el numero 3: "); 12 scanf("%d",p); 13 }</pre>
<p>Hay funciones o procedimientos en el código que sobrepasan el límite de líneas. Límite: 05</p> <p>Línea: 6</p>	

Tabla 100. CdP_22

ID: CdP_23	
Nombre:	Ejemplo de código con función correcta
Descripción:	<pre> 6 void fun (int *n, int *m){ 7 printf("Introduzca el numero 1: "); 8 scanf("%d",n); 9 printf("Introduzca el numero 2: "); 10 scanf("%d",m); 11 }</pre>
<p>Las funciones y procedimientos tienen la longitud idónea.</p>	

Tabla 101. CdP_23

ID: CdP_24	
Nombre:	Ejemplo de no seleccionar CHECK comprobar tamaño funciones
Descripción:	<input type="checkbox"/> Comprobar el tamaño de funciones y procedimientos (número de líneas) 05
<p>No se ha comprobado la longitud de las funciones y procedimientos en el código.</p>	

Tabla 102. CdP_24

ID: CdP_25	
Nombre:	Ejemplo de código con función mal definida
Descripción:	<pre> 6 void fun (int *n, int *m) { 7 printf("Introduzca el numero 1: "); 8 scanf("%d",n); 9 }</pre>
<p style="text-align: center;">Hay funciones o procedimientos mal declarados en el código.</p> <p style="text-align: center;">Línea: 6</p>	

Tabla 103. CdP_25

ID: CdP_26	
Nombre:	Ejemplo de código con función mal definida
Descripción:	<pre> 6 void fun (int *n, int *m) { 7 printf("Introduzca el numero 1: "); 8 scanf("%d",n); 9 printf("Introduzca el numero 2: "); 10 scanf("%d",m); 11 }</pre>
<p style="text-align: center;">Las funciones y los procedimientos están bien declarados.</p>	

Tabla 104. CdP_26

ID: CdP_27	
Nombre:	Ejemplo de no seleccionar CHECK definición funciones
Descripción:	<input type="checkbox"/> Comprobar la correcta definición de funciones y procedimientos
<p style="text-align: center;">No se ha comprobado la definición de funciones y procedimientos.</p>	

Tabla 105. CdP_27

ID: CdP_28	
Nombre:	Ejemplo de procedimiento con devolución errónea
Descripción:	<pre> 6 void fun (int *n) { 7 int a = 0; 8 printf("Introduzca el numero 1: "); 9 scanf("%d",n); 10 a = n + 1; 11 return a; 12 }</pre>
<p style="text-align: center;">Hay funciones o procedimientos que devuelven valores de forma incorrecta.</p> <p style="text-align: center;">Línea: 6</p>	

Tabla 106. CdP_28

ID: CdP_29	
Nombre:	Ejemplo de función con devolución errónea
Descripción:	<pre> 4 int fun (int n){ 5 int a = 0; 6 printf("Introduzca el numero 1: "); 7 scanf("%d",n); 8 a = n + 1; 9 }</pre>
<p style="color: red;">Hay funciones o procedimientos que devuelven valores de forma incorrecta.</p> <p style="color: orange;">Línea: 4</p>	

Tabla 107. CdP_29

ID: CdP_30	
Nombre:	Ejemplo de función con devolución correcta
Descripción:	<pre> 4 int fun (int n){ 5 int a = 0; 6 printf("Introduzca el numero 1: "); 7 scanf("%d",n); 8 a = n + 1; 9 return a; 10 }</pre>
<p style="color: green;">Las funciones y los procedimientos devuelven valores de forma correcta.</p>	

Tabla 108. CdP_30

ID: CdP_31	
Nombre:	Ejemplo de procedimiento con devolución correcta
Descripción:	<pre> 4 void fun (void){ 5 int a = 0; 6 printf("Introduzca el numero: "); 7 scanf("%d",a); 8 a = a * a; 9 printf("El cuadrado del numero introducido es %d.\n",a); 10 }</pre>
<p style="color: green;">Las funciones y los procedimientos devuelven valores de forma correcta.</p>	

Tabla 109. CdP_31

ID: CdP_32	
Nombre:	Ejemplo de no seleccionar CHECK devolución de subprogramas
Descripción:	<div style="border: 1px solid black; background-color: #007bff; color: white; padding: 5px; display: inline-block;"> <input type="checkbox"/> Comprobar la correcta devolución de funciones y procedimientos </div>
<p>No se ha comprobado la devolución de valores en funciones y procedimientos.</p>	

Tabla 110. CdP_32

ID: CdP_33	
Nombre:	Ejemplo de no seleccionar CHECK de interfaz de usuario
Descripción:	<div style="border: 1px solid black; padding: 5px;"> <input type="checkbox"/> Comprobar la interfaz de usuario <input checked="" type="checkbox"/> Comprobar que un subprograma que modifica datos no muestre mensajes por pantalla, y viceversa <input checked="" type="checkbox"/> Comprobar que si se lee un dato, antes se pida </div>
No se han comprobado las tareas en base a la interfaz de usuario.	

Tabla 111. CdP_33

ID: CdP_34	
Nombre:	Ejemplo de subprograma que calcule y además muestre mensajes
Descripción:	<pre> 4 int fun (int *num1){ 5 int a = 0; 6 printf("Introduzca un numero: "); 7 scanf("%d",a); 8 num1 = num1 + a; </pre>
<p style="color: red;">Hay funciones o procedimientos que imprimen por pantalla y además hacen cálculos con datos.</p> <p style="color: orange;">Línea: 4</p>	

Tabla 112. CdP_34

ID: CdP_35	
Nombre:	Ejemplo de subprograma con su función bien definida
Descripción:	<pre> 4 int fun (int *num1){ 5 printf("Introduzca un numero: "); 6 scanf("%d",num1); 7 return num1; 8 } </pre>
<p style="color: green;">Las funciones y los procedimientos tienen las tareas correctamente definidas.</p>	

Tabla 113. CdP_35

ID: CdP_36	
Nombre:	Ejemplo de no seleccionar CHECK de función de subprogramas
Descripción:	<div style="border: 1px solid black; padding: 5px;"> <input type="checkbox"/> Comprobar que un subprograma que modifica datos no muestre mensajes por pantalla, y viceversa </div>
No se han comprobado las tareas en funciones y procedimientos.	

Tabla 114. CdP_36

ID: CdP_37	
Nombre:	Ejemplo de programa con las lecturas mal definidas
Descripción:	<pre> 10 int main() { 11 int num1,num2; 12 scanf("%d",num1); 13 scanf("%d",num1); 14 }</pre>
<p>Hay lecturas de datos en el código, que no están precedidas de un mensaje pidiéndolo.</p> <p>Línea: 12</p> <p>Línea: 13</p>	

Tabla 115. CdP_37

ID: CdP_38	
Nombre:	Ejemplo de programa con las lecturas bien definidas
Descripción:	<pre> 4 int fun (int *num1){ 5 printf("Introduzca un numero: "); 6 scanf("%d",num1); 7 return num1; 8 }</pre>
<p>Las lecturas de datos están definidas correctamente.</p>	

Tabla 116. CdP_38

ID: CdP_39	
Nombre:	Ejemplo de no seleccionar CHECK de lecturas en el código
Descripción:	<input type="checkbox"/> Comprobar que si se lee un dato, antes se pida
<p>No se han comprobado las lecturas en el código.</p>	

Tabla 117. CdP_39

ID: CdP_40	
Nombre:	Ejemplo de variable sin inicializar
Descripción:	<pre> 3 int main() { 4 int num1; 5 printf("El valor del numero introducido es %d.\n",num1);</pre>
<p>Hay variables sin inicializar en el código.</p> <p>Línea: 4</p>	

Tabla 118. CdP_40

ID: CdP_41	
Nombre:	Ejemplo de código sin variables sin inicializar
Descripción:	<pre> 3 int main() { 4 int num1 = 0; 5 printf("El valor del numero introducido es %d.\n", num1); </pre> <p>No hay variables sin inicializar en el código.</p>

Tabla 119. CdP_41

ID: CdP_42	
Nombre:	Ejemplo de no seleccionar CHECK de variables sin inicializar
Descripción:	<input type="checkbox"/> Comprobar el acceso al valor de una variable no inicializada
No se ha comprobado si hay variables sin inicializar en el código.	

Tabla 120. CdP_42

ID: CdP_43	
Nombre:	Ejemplo de asignaciones seguidas
Descripción:	<pre> 3 int main() { 4 int num1; 5 num1 = 0; 6 num1 = 1; </pre> <p>Hay asignaciones de variables seguidas sin hacer uso de la misma entre ellas.</p> <p>Línea: 4</p>

Tabla 121. CdP_43

ID: CdP_44	
Nombre:	Ejemplo de asignaciones correctas
Descripción:	<pre> 5 num1 = 0; 6 printf("El valor del numero es %d.\n", num1); 7 num1 = 1; 8 printf("El valor del numero es %d.\n", num1); </pre> <p>No hay asignaciones de variables erróneas en el código.</p>

Tabla 122. CdP_44

ID: CdP_45	
Nombre:	Ejemplo de no seleccionar CHECK de asignaciones erróneas
Descripción:	<input type="checkbox"/> Comprobar la existencia de dos asignaciones, sin hacer uso de la variable entre ambas
No se ha comprobado si hay asignaciones de variables erróneas en el código.	

Tabla 123. CdP_45

ID: CdP_46	
Nombre:	Ejemplo de código con variables sin usar
Descripción:	<pre> 3 int main() { 4 int num1; 5 int num2; 6 num1 = 0; 7 printf("El valor del numero es %d.\n",num1); 8 return 0; 9 }</pre>
<p style="text-align: center;">Hay variables sin usar en el código.</p> <p style="text-align: center;">Línea: 5</p>	

Tabla 124. CdP_46

ID: CdP_47	
Nombre:	Ejemplo de código sin variables sin usar
Descripción:	<pre> 3 int main() { 4 int num1; 5 num1 = 0; 6 printf("El valor del numero es %d.\n",num1); 7 return 0; 8 }</pre>
<p style="text-align: center;">No hay variables sin usar en el código.</p>	

Tabla 125. CdP_47

ID: CdP_48	
Nombre:	Ejemplo de no seleccionar CHECK de comprobar variables sin usar
Descripción:	<input type="checkbox"/> Comprobar la definición de una variable sin utilizar
<p style="text-align: center;">No se ha comprobado si hay variables sin usar en el código.</p>	

Tabla 126. CdP_48

ID: CdP_49	
Nombre:	Ejemplo de código con variables globales
Descripción:	<pre> 1 #include <stdio.h> 2 3 int num2; 4 int main() { 5 int num1; 6 num1 = 0; 7 printf("El valor del numero es %d.\n",num1); 8 return 0; 9 }</pre>
<p style="text-align: center;">Hay variables globales en el código.</p> <p style="text-align: center;">Línea: 3</p>	

Tabla 127. CdP_49

ID: CdP_50	
Nombre:	Ejemplo de código sin variables globales
Descripción:	<pre> 1 #include <stdio.h> 2 3 int main() { 4 int num1; 5 num1 = 0; 6 printf("El valor del numero es %d.\n", num1); 7 return 0; 8 }</pre>
No hay variables globales en el código.	

Tabla 128. CdP_50

ID: CdP_51	
Nombre:	Ejemplo de no seleccionar CHECK de comprobar variables globales
Descripción:	<input type="checkbox"/> Comprobar la declaración de variables globales
No se ha comprobado el uso de variables globales en el código.	

Tabla 129. CdP_51

Capítulo 5. Planificación

A continuación se da una planificación estimada de lo que debía ser el desarrollo del proyecto, y la que acabó siendo la estimación real, debido a que en los meses de Febrero a Mayo se tuvo que cursar una asignatura, quedando el desarrollo de la herramienta en un segundo plano.

En el caso del desarrollo de esta herramienta, se ha utilizado un Modelo de desarrollo evolutivo, que consiste en desarrollar distintas versiones del mismo proyecto, añadiendo nuevas funciones en cada una. En el modelo evolutivo, se tiene en cuenta que los requisitos conocidos al inicio del proyecto, no tienen por qué cumplir con la funcionalidad final del proyecto, pudiendo aparecer nuevos requisitos en cualquier fase del desarrollo.

5.1 Planificación estimada

En este apartado se especifica mediante el uso de un diagrama de Gantt la planificación que se esperaba cumplir antes del inicio del proyecto. Como se ha comentado antes, esta quedó descartada debido a que se tuvo que cursar una asignatura en el segundo cuatrimestre, que le quitó tiempo de desarrollo al proyecto.

ta ▾	Nombre de tarea ▾	Duración ▾	Comienzo ▾	Fin ▾
★	Primera Reunión Tutor	1 día	mar 07/10/14	mar 07/10/14
★	▾ Fase Inicial	3 días	mié 08/10/14	vie 10/10/14
★	Análisis de las propuestas	3 días	mié 08/10/14	vie 10/10/14
★	▾ Primera Fase de Desarrollo	47 días	sáb 11/10/14	dom 14/12/14
★	Planteamiento del problema	7 días	sáb 11/10/14	dom 19/10/14
★	Primera Fase de Análisis	6 días	lun 20/10/14	dom 26/10/14
★	Primera Fase de Implementación	36 días	lun 27/10/14	dom 14/12/14
★	Segunda Reunión Tutor	1 día	lun 15/12/14	lun 15/12/14
★	▾ Segunda Fase del Desarrollo	32 días	lun 05/01/15	mar 17/02/15
★	Segunda Fase de Análisis	5 días	mar 06/01/15	dom 11/01/15
★	Segunda Fase de Implementación	27 días	lun 12/01/15	mar 17/02/15
★	Tercera Reunión Tutor	1 día	mié 18/02/15	mié 18/02/15
★	▾ Tercera Fase de Desarrollo	58 días	jue 19/02/15	dom 10/05/15
★	Tercera Fase de Análisis	3 días	jue 19/02/15	dom 22/02/15
★	Tercera Fase de Implementación	16 días	lun 23/02/15	dom 15/03/15
★	Primera Fase de Documentación	41 días	lun 16/03/15	dom 10/05/15
★	Cuarta Reunión Tutor	1 día	lun 11/05/15	lun 11/05/15
★	▾ Cuarta Fase de Desarrollo	30 días	mar 12/05/15	lun 22/06/15
★	Segunda Fase de Documentación	30 días	mar 12/05/15	lun 22/06/15
★	Entrega Memoria	1 día	lun 22/06/15	lun 22/06/15

Ilustración 50. Tabla Diagrama de Gantt – Planificación Estimada

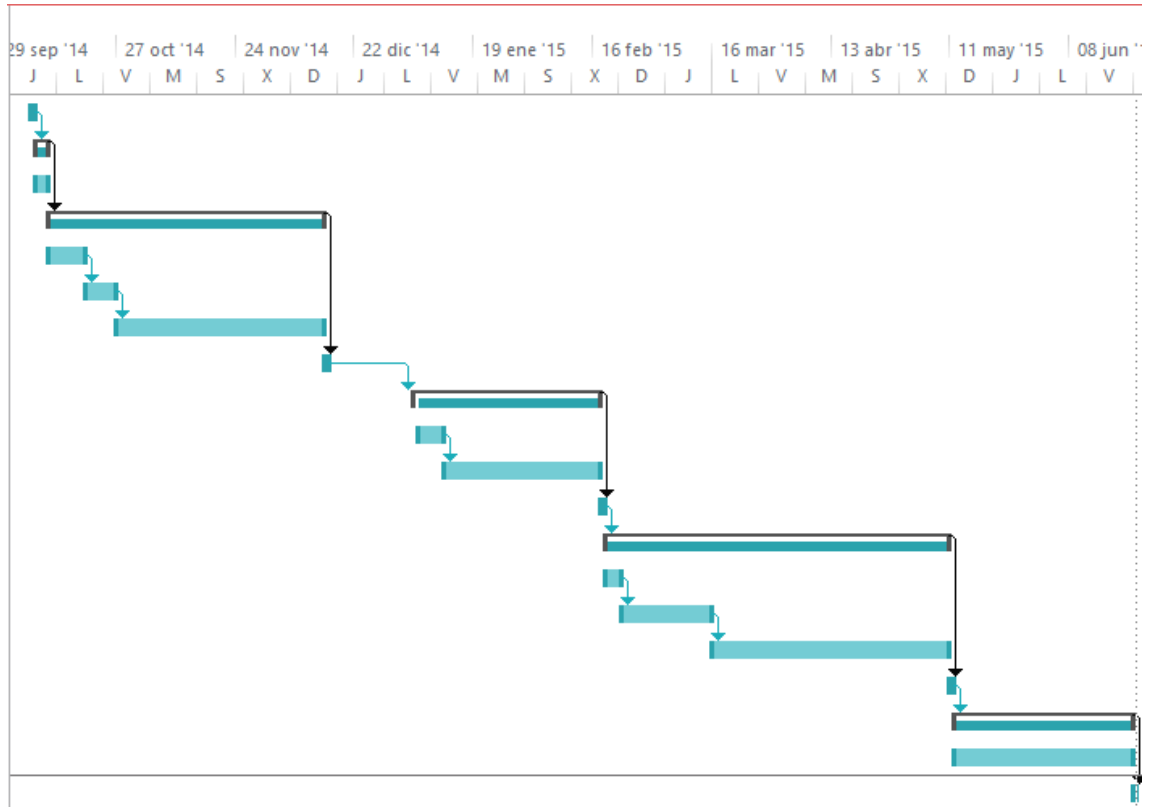


Ilustración 51. Diagrama de Gantt – Planificación Estimada

5.2 Planificación Real

ta ▾	Nombre de tarea ▾	Duración ▾	Comienzo ▾	Fin ▾
★	Primera Reunión Tutor	1 día	mar 07/10/14	mar 07/10/14
★	▾ Fase Inicial	3 días	mié 08/10/14	vie 10/10/14
★	Análisis de las propuestas	3 días	mié 08/10/14	vie 10/10/14
★	▾ Primera Fase de Desarrollo	47 días	sáb 11/10/14	dom 14/12/14
★	Planteamiento del problema	7 días	sáb 11/10/14	dom 19/10/14
★	Primera Fase de Análisis	6 días	lun 20/10/14	dom 26/10/14
★	Primera Fase de Implementación	36 días	lun 27/10/14	dom 14/12/14
★	Segunda Reunión Tutor	1 día	lun 15/12/14	lun 15/12/14
★	▾ Segunda Fase del Desarrollo	32 días	lun 05/01/15	mar 17/02/15
★	Segunda Fase de Análisis	5 días	mar 06/01/15	dom 11/01/15
★	Segunda Fase de Implementación	27 días	lun 12/01/15	mar 17/02/15
★	Tercera Reunión Tutor	1 día	mié 18/02/15	mié 18/02/15
★	▾ Tercera Fase de Desarrollo	58 días	jue 19/02/15	dom 10/05/15
★	Tercera Fase de Análisis	3 días	jue 19/02/15	dom 22/02/15
★	Tercera Fase de Implementación	56 días	lun 23/02/15	dom 10/05/15
★	Cuarta Reunión Tutor	1 día	lun 11/05/15	lun 11/05/15
★	▾ Cuarta Fase de Desarrollo	86 días	mar 12/05/15	mar 08/09/15
★	Cuarta Fase de Análisis	15 días	mar 12/05/15	lun 01/06/15
★	Cuarta Fase de Implementación	21 días	mar 02/06/15	mar 30/06/15
★	Primera Fase de Documentación	50 días	mié 01/07/15	mar 08/09/15
★	Quinta Reunión Tutor	1 día	mié 09/09/15	mié 09/09/15
★	▾ Quinta Fase de Desarrollo	13 días	jue 10/09/15	dom 27/09/15
★	Segunda Fase de Documentación	12 días	vie 11/09/15	dom 27/09/15
★	Entrega Memoria	1 día	dom 27/09/15	dom 27/09/15

Ilustración 52. Tabla Diagrama de Gantt – Planificación Real

El desarrollo del trabajo ha abarcado finalmente 236 días, que se pueden agrupar de la siguiente forma:

FASE	DÍAS
Planteamiento del problema	7
Fase de Análisis	29
Fase de Implementación	140
Fase de Documentación	62

Tabla 130. Desarrollo del trabajo

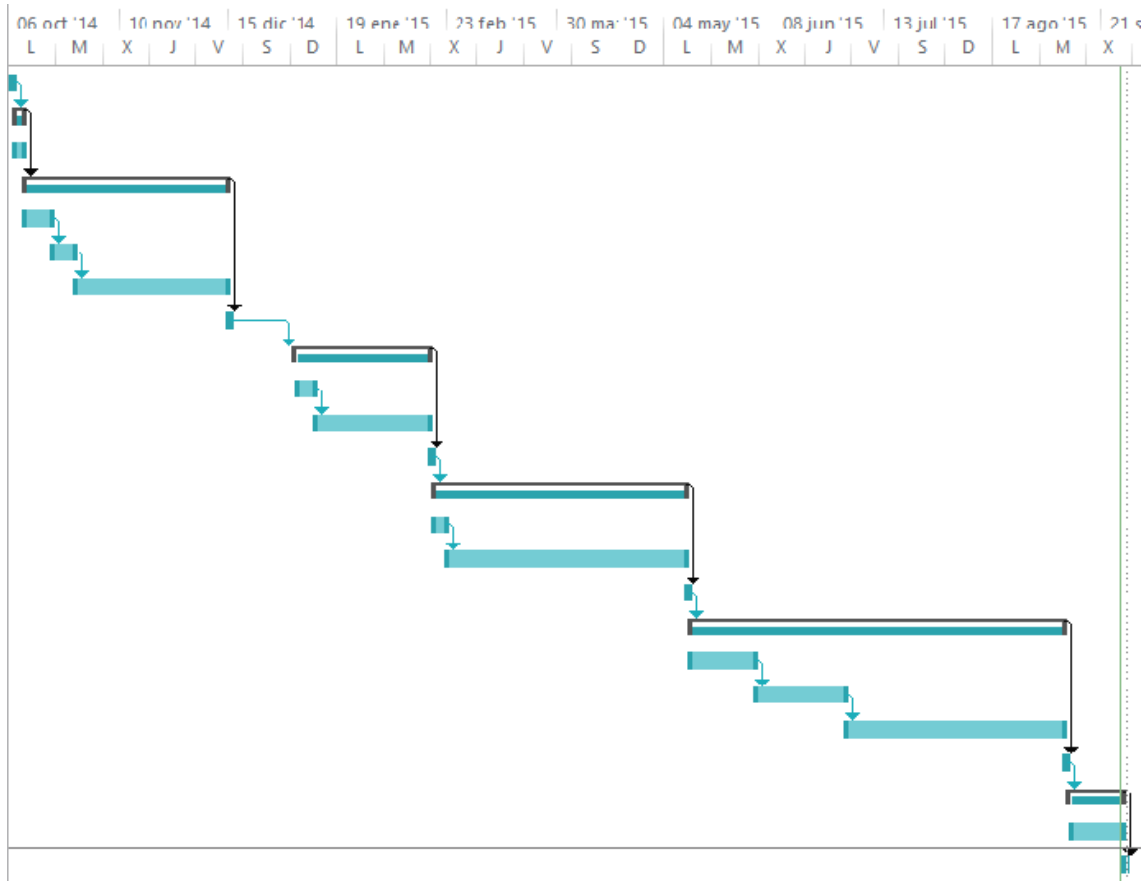


Ilustración 53. Diagrama de Gantt – Planificación Real

La planificación Real ha tenido las siguientes etapas:

- **Primera Reunión Tutor.** En ésta primera reunión con el tutor, se explicaron los distintos proyectos existentes al alumno para su elección.
- **Fase Inicial.**
 - o **Análisis de las Propuestas.** Se estudiaron las distintas propuestas a realizar, hasta elegir la desarrollada en éste proyecto.
- **Primera Fase de Desarrollo.**
 - o **Planteamiento del problema.** En esta etapa concluyó que la herramienta sería desarrollada completamente en lenguaje Visual Basic .NET. Además se pensó una idea inicial de la aplicación.
 - o **Primera Fase de Análisis.** En ésta primera etapa de análisis, se hallaron los siguientes requisitos: REQ_FUN_08, REQ_FUN_14, REQ_FUN_15, REQ_FUN_16, REQ_FUN_17, REQ_FUN_18, REQ_FUN_19, REQ_FUN_20, REQ_FUN_21, REQ_FUN_23, REQ_FUN_24, REQ_FUN_25, REQ_FUN_26, REQ_FUN_27, REQ_FUN_28, REQ_FUN_29, REQ_FUN_30, REQ_FUN_31, REQ_FUN_32, REQ_FUN_33 y REQ_FUN_35.
 - o **Primera Fase de Implementación.** En esta fase se realizó una primera versión de la aplicación, que cumplía con los siguientes requisitos: REQ_FUN_08, REQ_FUN_14, REQ_FUN_15, REQ_FUN_16, REQ_FUN_17, REQ_FUN_18, REQ_FUN_19, REQ_FUN_20, REQ_FUN_21 y REQ_FUN_35.
- **Segunda Reunión Tutor.** Muestra al tutor de la aplicación desarrollada hasta el momento.

- **Segunda Fase de Desarrollo.**
 - **Segunda Fase de Análisis.** Tras la segunda reunión con el tutor, se hallaron los siguientes requisitos: REQ_FUN_09, REQ_FUN_10, REQ_FUN_11, REQ_FUN_12 y REQ_FUN_22.
 - **Segunda Fase de Implementación.** En esta fase, se cambió completamente el diseño gráfico de la aplicación, y además se añadieron a la misma los siguientes requisitos: REQ_FUN_09, REQ_FUN_10, REQ_FUN_11, REQ_FUN_12 y REQ_FUN_22.
- **Tercera Reunión con el Tutor.** En esta reunión, se mostró la aplicación desarrollada hasta el momento, y se propuso alguna novedad para la misma.
- **Tercera Fase de desarrollo.**
 - **Tercera Fase de Análisis.** Tras la tercera reunión con el tutor, se añadieron los siguientes requisitos al sistema: REQ_FUN_01, REQ_FUN_02, REQ_FUN_03, REQ_FUN_04, REQ_FUN_05.
 - **Tercera Fase de Implementación.** En esta etapa, se implementaron los siguientes requisitos: REQ_FUN_01, REQ_FUN_02, REQ_FUN_03, REQ_FUN_04, REQ_FUN_05, REQ_FUN_23, REQ_FUN_24, REQ_FUN_33. Además, debido a la inclusión en la aplicación de distintos perfiles, se cambió de nuevo el diseño gráfico de la aplicación.
- **Cuarta Reunión con el tutor.** Se muestra los resultados obtenidos hasta el momento al tutor.
- **Cuarta Fase de desarrollo.**
 - **Cuarta Fase de Análisis.** En esta etapa, se obtuvieron la totalidad de los requisitos de los que consta el sistema, añadiendo a los existentes hasta el momento los siguientes: REQ_FUN_06, REQ_FUN_07, REQ_FUN_13 y REQ_FUN_34.
 - **Cuarta Fase de Implementación.** En esta etapa, se completó el desarrollo de la aplicación, implementándose los siguientes requisitos: REQ_FUN_06, REQ_FUN_07, REQ_FUN_13, REQ_FUN_25, REQ_FUN_26, REQ_FUN_27, REQ_FUN_28, REQ_FUN_29, REQ_FUN_30, REQ_FUN_31, REQ_FUN_32 y REQ_FUN_34.
 - **Primera Fase de Documentación.** En esta primera fase de la documentación, se redactó en forma de borrador la mayor parte de la memoria final, incluyendo los puntos de Introducción, Estado del arte, Fase de Análisis, Diseño e Implementación y el apartado de Pruebas.
- **Quinta Reunión con el tutor.** En esta etapa se muestra la aplicación en su versión final al tutor, y se hace una revisión del estado de la documentación.
- **Quinta Fase de desarrollo.**
 - **Segunda Fase de la documentación.** Se desarrollan los puntos restantes de la memoria, incluyendo la Planificación, el Presupuesto, Conclusiones, Líneas Futuras y demás apartados.

Capítulo 6. Presupuesto

A continuación se hará una estimación de presupuesto del proyecto en base a los recursos humanos y materiales para llevarlo a cabo.

6.1 Recursos Materiales

NOMBRE	CANTIDAD	PRECIO UNITARIO	IMPORTE TOTAL
Ordenador Portatil Hp Pavilion i3	1	479 €	479 €
Memoria USB 4 GB	1	4 €	4 €
TOTAL	*	*	483 €

Tabla 131. Presupuesto recursos Hardware

NOMBRE	CANTIDAD	PRECIO UNITARIO	IMPORTE TOTAL
Microsoft Visual Studio	1	0 €	0 €
MySQL Comunnity Server	1	0 €	0 €
MySQL Workbench	1	0 €	0 €
Microsoft Office University Suscription	1	79 €	79 €
Microsoft Project (versión prueba 1 mes)	1	0 €	0 €
TOTAL	*	*	79 €

Tabla 132. Presupuesto recursos Software

6.2 Recursos Humanos

NOMBRE	SUELDO MENSUAL	SUELDO ANUAL
Grado en Ingeniería Informática Analista Programador	1.300 €	24 000 € - 26 000 €

Tabla 133. Presupuesto recursos humanos

6.3 Presupuesto final

NOMBRE	IMPORTE TOTAL
Recursos Hardware	483 €
Recursos Software	79 €
Recursos humanos	24.000 €
TOTAL	24.562 €

Tabla 134. Presupuesto final

Capítulo 7. Conclusiones

Una vez finalizado el proyecto, la realización de este Trabajo Fin de Grado ha cumplido con solvencia los objetivos marcados antes de su desarrollo. La utilización de la herramienta Visual Studio con el lenguaje de programación Visual Basic .NET ha sido más que acertada. Gracias a los conocimientos previos del lenguaje, y del agradable entorno gráfico de la herramienta permiten realizar el desarrollo de aplicaciones de forma cómoda. Además ha permitido solucionar cualquier función extra requerida aun basándose en otras herramientas, como era el uso de una base de datos externa mediante la utilización de MySQL, o la creación del archivo de resultados mediante el programa Microsoft Office Word.

En lo que respecta a la aplicación desarrollada en el proyecto, se han alcanzado todos los requisitos requeridos por el cliente, el tutor en este caso, y se ha añadido alguno extra para darle mayor funcionalidad a la aplicación, y producir mejoras en la misma, como pueden ser el tiempo de espera para el análisis, la ventana de ayuda al usuario, o el uso de tooltip que proporcionan información al usuario sin que este la solicite de forma explícita.

Pese a que existen herramientas para el lenguaje C con funciones similares a la desarrollada en el proyecto, como se ha demostrado durante esta memoria, ninguna alcanza la funcionalidad cubierta por esta herramienta. Ya que todas se centran en errores más cerca de la compilación del programa que de cualquier otra cosa, mientras que esta aplicación se basa más en la búsqueda de errores de estilo.

La herramienta no solo permite al alumno arreglar esos errores de estilo cometidos, sino que le ayuda a evitar esos mismos errores en próximos programas, ya sean en lenguaje C como en cualquier otro lenguaje. Además dota al alumno de una autonomía a la hora de revisar su código, sin tener que depender de la revisión del profesor.

Permite además a los profesores realizar pruebas inexistentes en cualquier otra herramienta, de manera rápida y sencilla sobre una gran cantidad de archivos, y obtener los resultados en un archivo de fácil revisión.

En lo que se refiere a motivación y objetivos personales, la aplicación ha cumplido completamente las expectativas fijadas. Se ha conseguido desarrollar una aplicación de un tamaño nunca antes realizado de forma individual, basada además en un lenguaje no utilizado en ninguna asignatura del Grado de Ingeniería Informática.

Capítulo 8. Líneas Futuras

Debido a que para la realización del proyecto hay un tiempo limitado, no todos los objetivos marcados o ideas planteadas pueden ser desarrollados al final del proyecto. A continuación se muestran algunas posibles mejoras a realizar en la aplicación:

- **Añadir un compilador a la funcionalidad.** Como se ha realizado en la asignatura Procesadores del Lenguaje del Grado en Ingeniería Informática, se podría introducir un compilador para que la herramienta no solo revisara y buscara posibles errores de estilo en el código, sino que además fuera capaz de compilar dicho código, para comprobar que el programa funciona correctamente y realiza su función esperada.
- **Reducir los tiempos de espera.** Aunque para el alumno el tiempo de espera es casi imperceptible (suele ser menor de 5 segundos dependiendo de la longitud del archivo), cuando es el profesor quien realiza el análisis, y este se realiza sobre una gran cantidad de archivos, el tiempo de espera para el usuario es bastante elevado. Esto se debe a que el tiempo de espera es proporcional al número de archivos a tratar y a su longitud. Una solución para esto sería mostrar resultados parciales al profesor sin que tuviera que esperar al análisis completo, o crear un archivo de análisis para cada archivo tratado.
- **Crear aplicación web.** Una muy buena opción sería utilizar la aplicación como una aplicación web, consiguiendo ventajas como poder utilizarla sin necesidad de instalar ningún programa, conseguir que la herramienta fuera compatible con cualquier sistema operativo o permitir una disponibilidad mucho más alta, entre otras.
- **Tratamiento de otros lenguajes.** Se podría realizar el análisis de la herramienta en lenguajes de programación, para ello podría optarse por varias ideas:
 - **Reutilización de código.** Podría reutilizarse el código, cambiando las palabras clave en función del lenguaje que se quiera tratar, existiendo el inconveniente de tener una aplicación por cada lenguaje tratado.
 - **Añadir lenguaje.** Añadir un campo a la aplicación para elegir el lenguaje de programación a tratar. Sería una opción más recomendable, ya que se tendría una sola aplicación para tratar todos los lenguajes, y se podría hacer de manera gradual.
- **Otros errores a tratar.** Otros errores que podrían tratarse en futuras versiones sería:
 - **Comprobar comentarios coherentes.** Uno de los errores revisados por la aplicación es comprobar que la línea anterior a una función o procedimiento tiene un comentario. Sería recomendable analizar dicho comentario, y que este tuviera relación con la función realizada por el subprograma.
 - **Comprobar longitud línea.** Limitar, al igual que ocurre con la longitud de los subprogramas, la longitud máxima que pueden tener las líneas en el código.
 - **Nombres de variables y subprogramas.** Añadir una funcionalidad que comprobara que los nombre de las variables y subprogramas declarados en el código fueran representativos.

- **Comprobar copias.** Permitir al profesor comparar los distintos archivos a analizar, buscando posibles copias entre los mismos, o en una base de datos de archivos guardados de otros años.
- **Realizar informe.** Añadir una funcionalidad al sistema, que permitiera al profesor realizar un informe sobre cada alumno, detallándose la evolución del mismo según avance el curso.

Anexo. Manual de Usuario

a.1 Instalación

La aplicación no necesita instalarse para su funcionamiento, pero es necesario instalar una serie de componentes para su funcionamiento.

- **MySQL Community Server.** Deberá ser el profesor quien creé la base de datos que permita identificarse en la aplicación. Para ello deberá descargarse esta herramienta desde [AQUÍ](#). Hecho esto se descomprimirá e instalará el servidor de forma sencilla siguiendo los pasos indicados por la aplicación.
- **MySQL Workbench.** No es necesario instalar esta herramienta para el funcionamiento de la aplicación, pero para simplificar se creará y ejecutará el script de MySQL con ella. Para ello se descargará desde [AQUÍ](#), se descomprimirá e instalará la herramienta de forma sencilla siguiendo los pasos indicados por la aplicación.

Una vez instalada la herramienta MySQL Workbench, se abrirá y se pulsará en el siguiente apartado para conectarse al servidor.

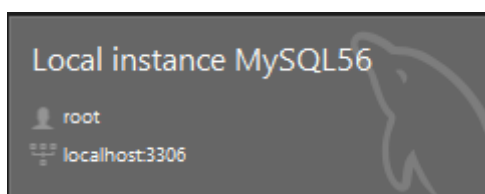


Ilustración 54. Conectar Servidor MySQL

Se abrirá el siguiente cuadro de diálogo, en el que se introducirá la contraseña del propio ordenador:

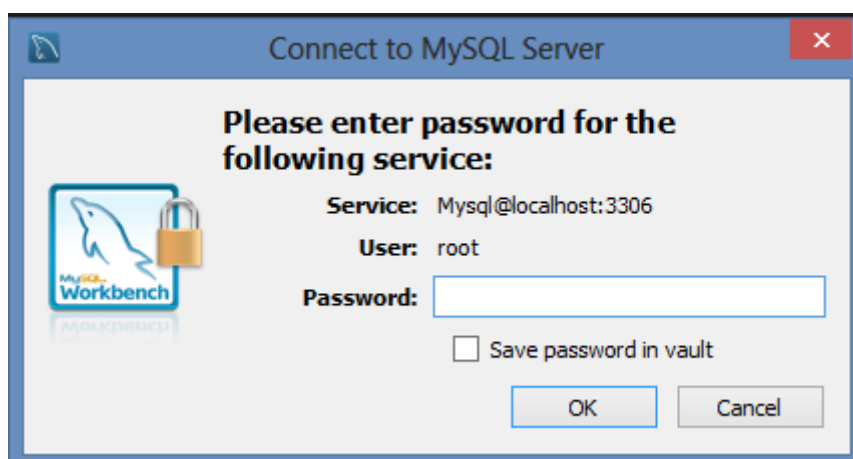


Ilustración 55. Mensaje MySQL


Hecho esto, se creará un script en MySQL creando la base de datos y las diferentes tablas de las que hará uso la aplicación.

```

1 • drop database data_pass;
2 • create database data_pass;
3 • drop user db_user;
4 • create user db_user;
5
6 • grant all on data_pass.* to 'db_user'@'localhost' identified by 'db_password';
7
8 • use data_pass;
9
10 • create table prof_pass(
11     nombre_usuario varchar(20) not null,
12     pass            varchar(20) not null);
13
14 • create table alum_pass(
15     nombre_usuario varchar(20) not null,
16     pass            varchar(20) not null);
17
18 • insert into prof_pass values ('prof1','1234');
19 • insert into prof_pass values ('prof2','1234');
20 • insert into alum_pass values ('alum1','1234');
21 • insert into alum_pass values ('alum2','1234');
22
23 • GRANT ALL PRIVILEGES ON data_pass.* TO 'db_user'@'%' IDENTIFIED BY 'db_password';
24 • FLUSH PRIVILEGES;

```

Ilustración 56. Creación base de datos y tablas

Se ejecutará el script pulsado en el siguiente icono  lo cual permitirá crear las tablas y la base de datos, y por último se guardará el archivo.

- **Microsoft Office Word.** Como el archivo de resultados se escribe en un *.doc, hace falta tener instalada esta herramienta.

La aplicación se encuentra en la carpeta: C:\...\definitivo\Analizador-C\Analizador-C\bin\Debug. Para ejecutarla, simplemente es necesario hacer click en el ejecutable de la aplicación:







Nombre	Tipo	Tamaño
 Analizador-C	Aplicación	685 KB
 Analizador-C	Program Debug Database	164 KB
 Analizador-C.vshost	Aplicación	14 KB
 Analizador-C.vshost.exe.manifest	Archivo MANIFEST	1 KB
 Analizador-C	Archivo XML	1 KB
 MySql.Data.dll	Extensión de la aplicación	434 KB

Ilustración 57. Ejecutable de la aplicación

a.2 Pantallas

Una vez se ha accedido a la aplicación, aparecerá la ventana de inicio de sesión, que contiene las siguientes opciones:

- **Comprobar conexión.** Permite comprobar al usuario si está conectado a la base de datos.
- **Aceptar.** Permite acceder a la ventana principal de la aplicación siempre que el usuario y la contraseña introducidos sean correctos.
- **Salir.** Permite al usuario salir de la aplicación.

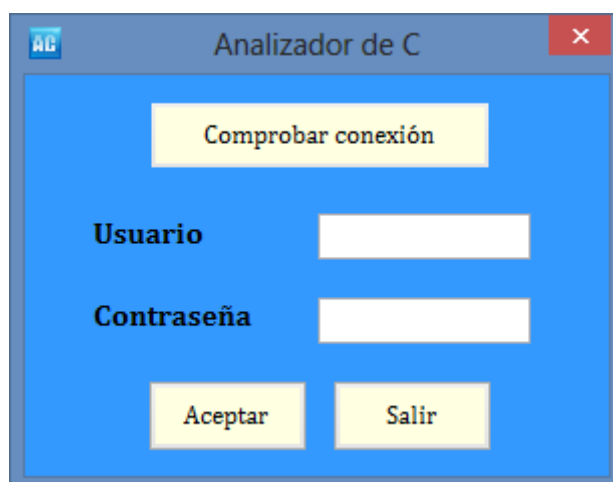


Ilustración 58. Ventana principal

Si se accede con rol de profesor a la aplicación, el usuario podrá marcar y desmarcar los distintos CHECK de la ventana, podrá modificar el valor del número de líneas máximo de los subprogramas, podrá guardar la configuración, y podrá cargar una carpeta con varios archivos a analizar. La ventana que se le presentará será la siguiente:

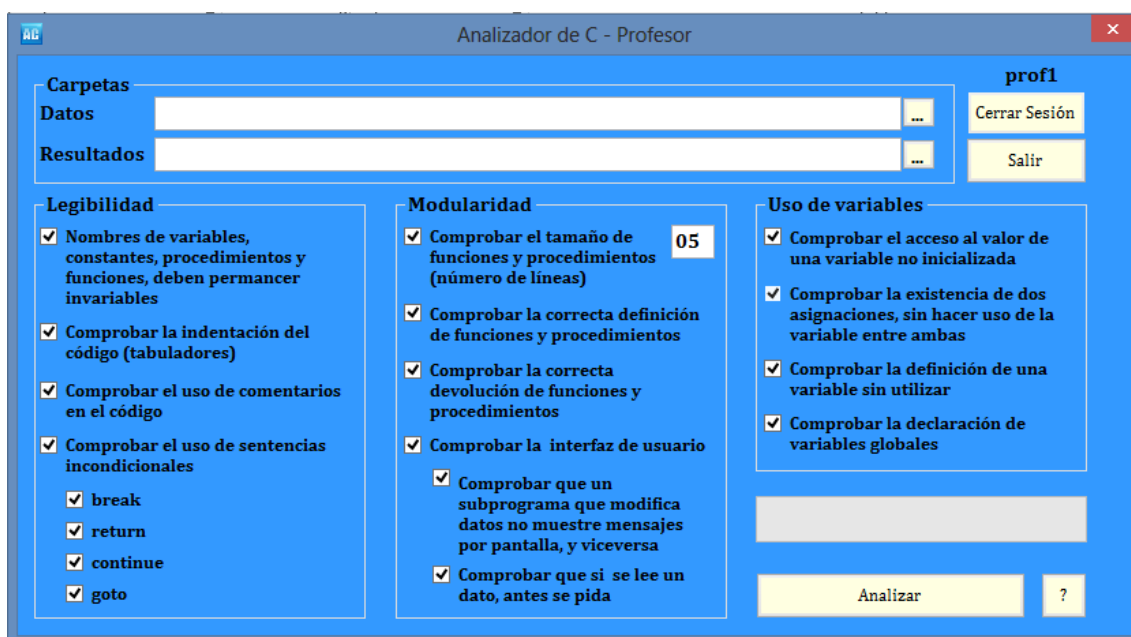


Ilustración 59. Ventana principal profesor

Mientras que si se accede a la aplicación con rol de alumno, la ventana que se le presentará será la siguiente, con las restricciones de no poder cambiar el valor de ninguno de los CHECK contenidos en la ventana, ni tampoco modificar el campo del tamaño máximo de los subprogramas, no podrá guardar la configuración, y sólo podrá cargar un archivo a analizar. Su ventana tendrá el siguiente aspecto:

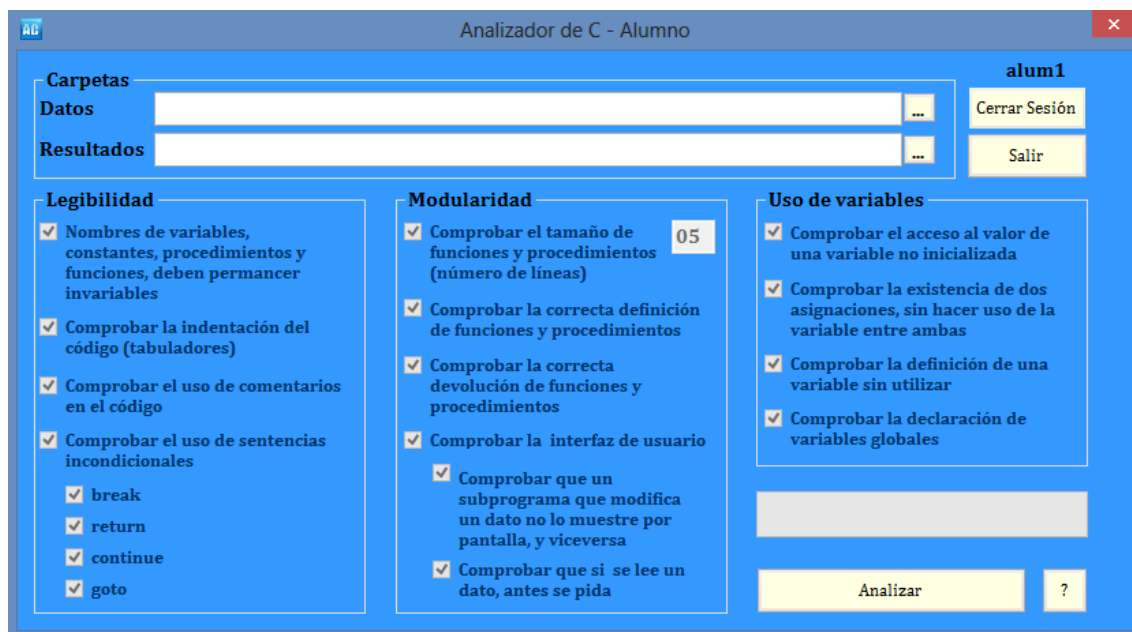


Ilustración 60. Ventana Principal alumno

En la siguiente ilustración aparecen los diferentes botones de la aplicación, cuyas funciones son las siguientes:

1. **Botón Datos.** Permite al usuario cargar los datos sobre los que se realizará el análisis. Un archivo en caso de ser un alumno, o una carpeta con un conjunto de archivos en caso de ser profesor.
2. **Botón Resultados.** Permite al usuario elegir la carpeta donde se guardará el archivo con los resultados del análisis.
3. **Botón Cerrar Sesión.** Permite al usuario cerrar la sesión actual, y volver a la ventana de inicio de sesión.
4. **Botón Salir.** Permite al usuario salir de la aplicación.
5. **Barra de carga.** Permite al usuario observar el progreso del análisis.
6. **Botón de ayuda.** Muestra al usuario un menú de ayuda sobre las distintas funcionalidades del programa, y pequeñas explicaciones de su funcionamiento.
7. **Botón analizar.** Permite a los usuarios realizar el análisis del archivo o archivos, siempre y cuando tanto la ruta de datos como la de resultados estén seleccionadas.

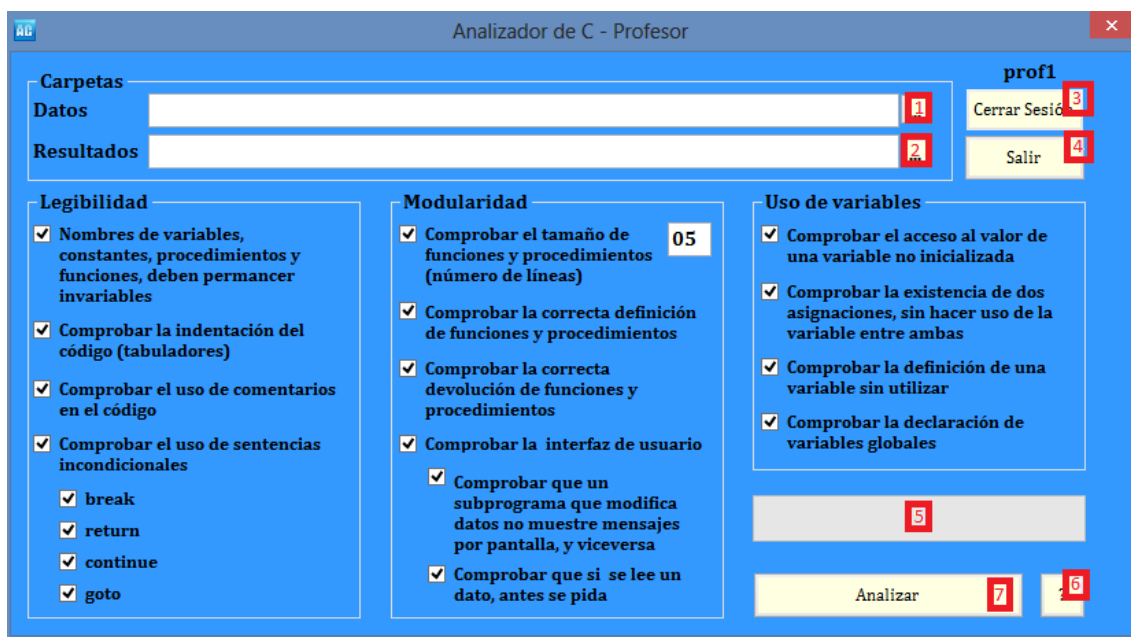


Ilustración 61. Ventana principal botones

La ventana de ayuda que aparece al pulsar en el botón de ayuda (6) es la siguiente:

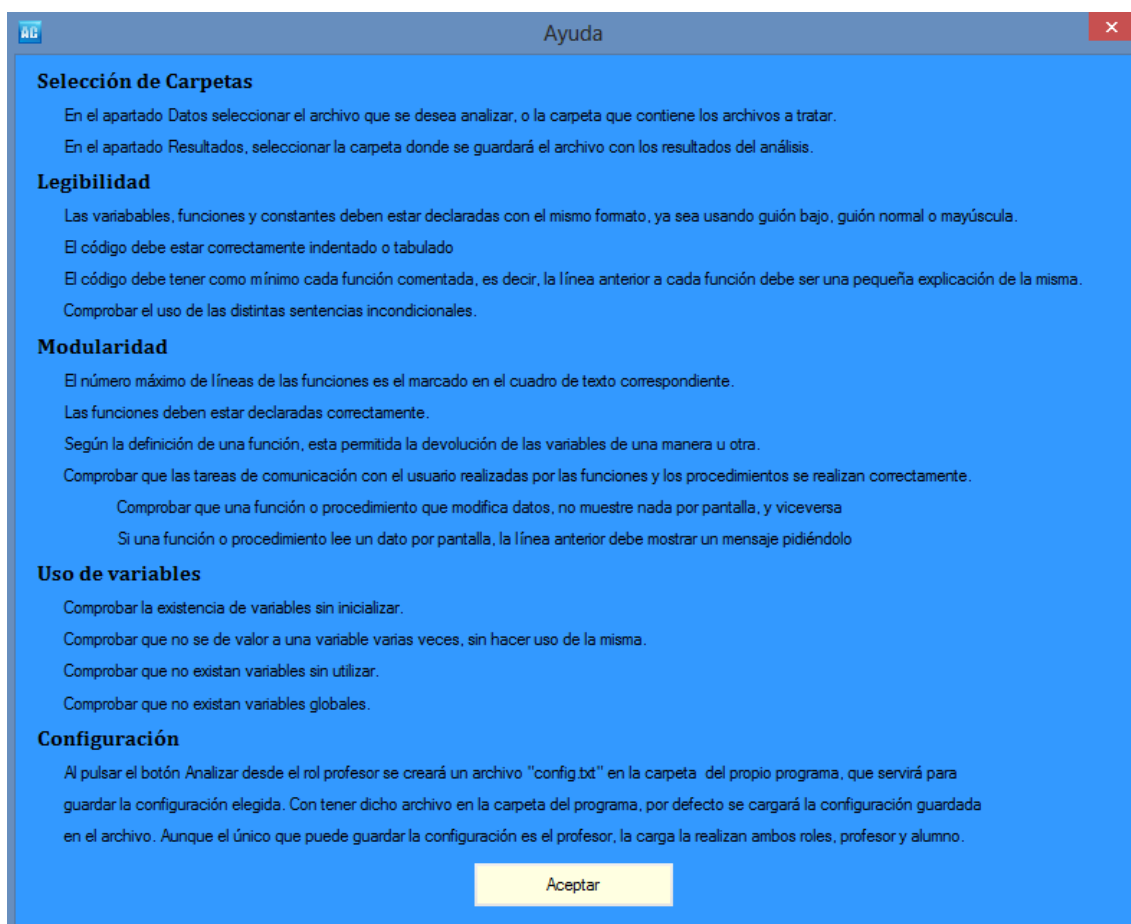


Ilustración 62. Ventana de ayuda

Si se accede al programa como rol de profesor, al pulsar en el botón analizar (7), además de crear el correspondiente archivo de análisis, se creará en la carpeta de la aplicación un archivo de configuración codificado con las opciones seleccionadas en ese momento.

Dicho archivo será cargado tanto por alumnos como por profesores. A continuación se muestra una ventana con todos los CHECK desmarcados y con el valor máximo de líneas permitido en 99:

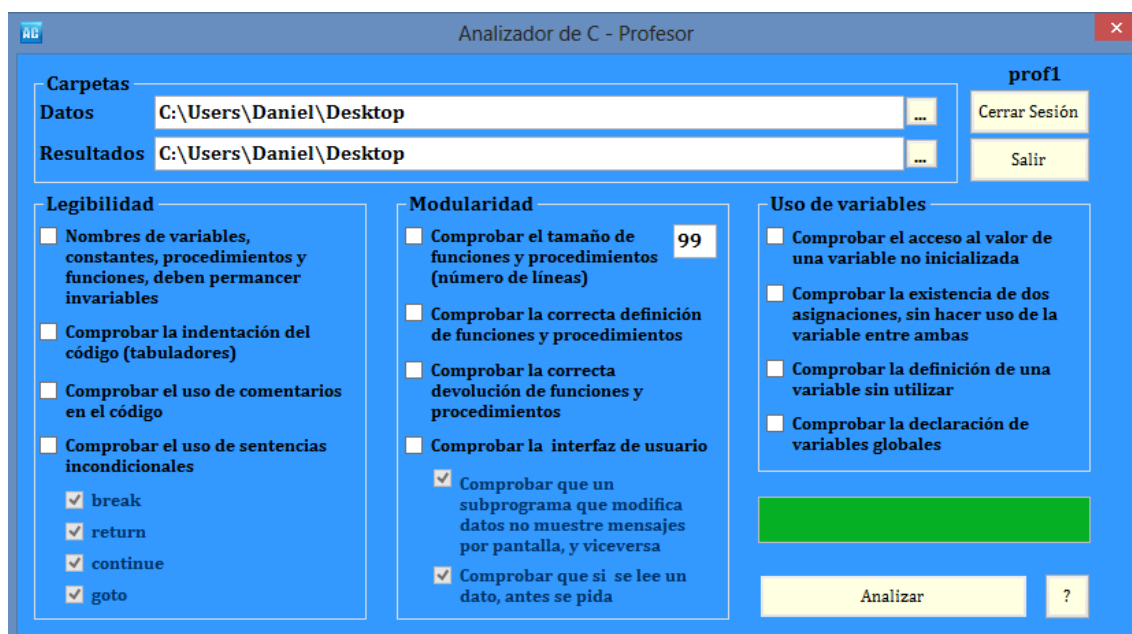


Ilustración 63. Ventana Ejemplo configuración

En la carpeta de la aplicación se creará un archivo “config.txt” que contiene la configuración deseada. Con pasar al alumno la carpeta de la aplicación con este archivo dentro, cargará la configuración guardada:

Nombre	Tipo	Tamaño
Analizador-C	Aplicación	689 KB
Analizador-C	Program Debug D...	164 KB
Analizador-C.vshost	Aplicación	14 KB
Analizador-C	Archivo XML	1 KB
config	Documento de tex...	6 KB
MySql.Data.dll	Extensión de la apl...	434 KB

Ilustración 64. Carpeta configuración

A continuación se muestra el archivo codificado, y una muestra de la ventana del alumno con la configuración guardada anteriormente, es decir, todos los CHECK desmarcados, y el valor máximo de líneas de los subprogramas a 99:

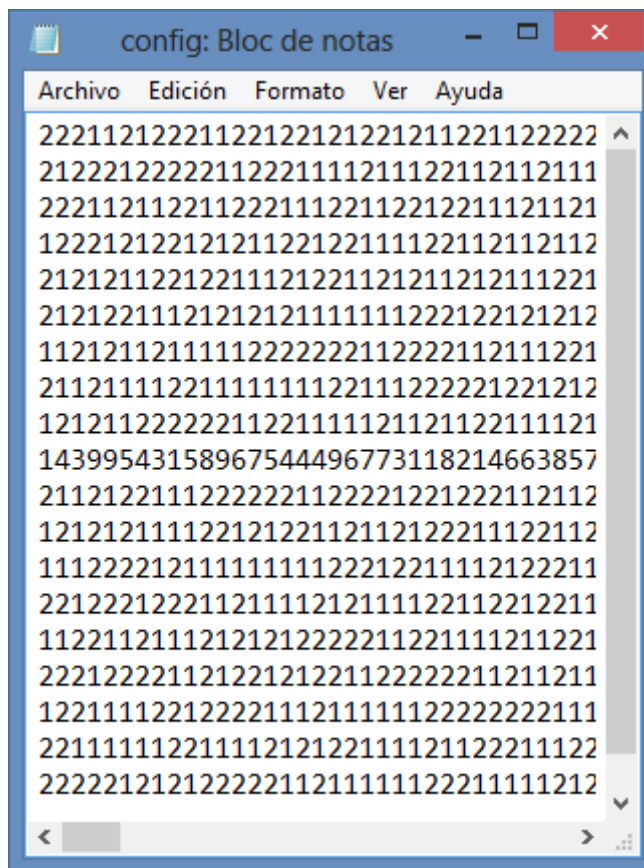


Ilustración 65. Ejemplo config.txt

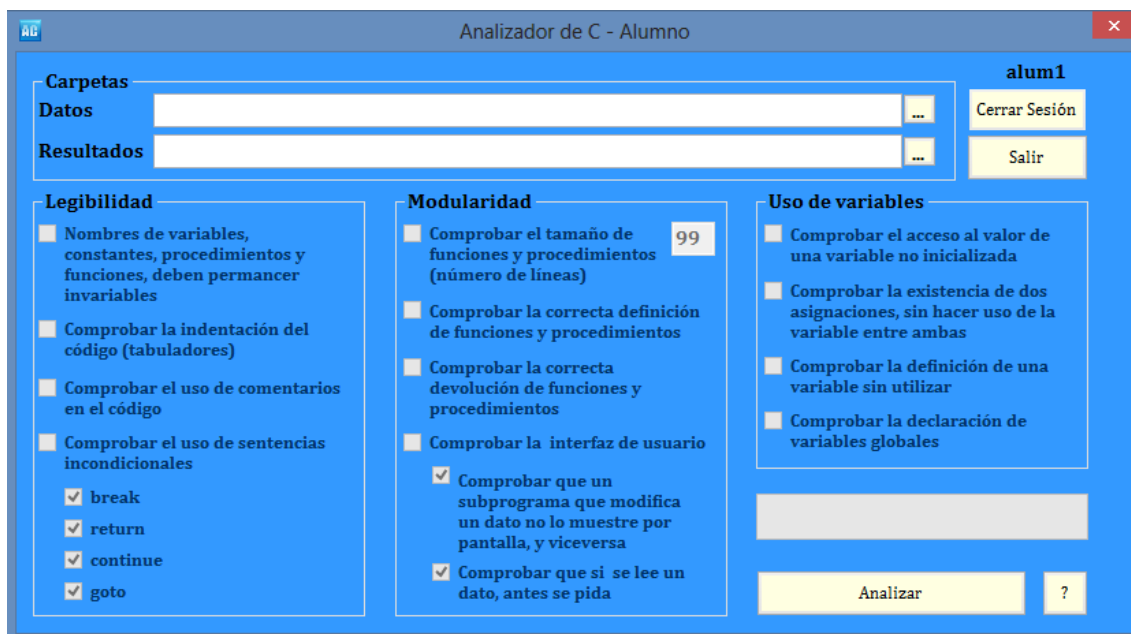


Ilustración 66. Ventana alumno configuración

Por último indicar que el fichero solución tendrá inicialmente en color azul la ruta y el nombre del archivo tratado. Se incluirá un comentario en color verde si la prueba ha sido pasada con éxito, o en color rojo si la prueba no ha sido pasada, indicándose en color amarillo además la línea donde se produce el error. Esto se repetirá para cada uno de los archivos a analizar siempre que se acceda con rol de profesor, mientras que si se accede con rol de alumno, saldrá la información para el único archivo seleccionado. Su aspecto es el siguiente:

```

C:\Users\Daniel\Desktop\pruebas_aaa.c
=====
El código tiene funciones, procedimientos y variables declaradas con formato
correcto.
El código está bien tabulado o indentado.
El código está mal comentado.
No se ha encontrado la instrucción break en código.
Se ha encontrado la instrucción return en código. Apariciones: 1
    Línea: 8
No se ha encontrado la instrucción continue en código.
No se ha encontrado la instrucción goto en código.
Las funciones y procedimientos tienen la longitud idónea.
Las funciones y los procedimientos están bien declarados.
Las funciones y los procedimientos devuelven valores de forma correcta.
Las funciones y los procedimientos tienen las tareas correctamente definidas.
Las lecturas de datos están definidas correctamente.
No hay variables sin inicializar en el código.
No hay asignaciones de variables erróneas en el código.
Hay variables sin usar en el código.
    Línea: 3
Hay variables globales en el código.
    Línea: 3
=====

```

Ilustración 67. Archivo solución

Referencias

- [1] - eldiario.es - [Junio 2014]
< http://www.eldiario.es/turing/Ninos-programadores-ensenanza-programacion-escuelas_0_293970921.html >
- [2] - blogspot.com.es - [Diciembre 2009]
< <http://jesusgonzalezfonseca.blogspot.com.es/2009/12/cuales-son-las-carreras-con-menos.html> >
- [3] - Autotest de la UC3M - [Septiembre 2015]
< <http://carbonero.arcos.inf.uc3m.es/login.aspx> >
- [4] - Wikipedia-Lenguajes de programación- [Septiembre 2015]
< https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n >
- [5] - Wikipedia-lenguajes de alto nivel- [Septiembre 2015]
< https://es.wikipedia.org/wiki/Lenguaje_de_alto_nivel >
- [6] - Wikipedia-lenguaje preprocesado- [Octubre 2014]
< <https://es.wikipedia.org/wiki/Preprocesado> >
- [7] - Wikipedia-recolector de basura- [Mayo 2015]
< https://es.wikipedia.org/wiki/Recolector_de_basura >
- [8] - Prevención, gestión y tipos de errores- [Mayo 2014]
< http://aprenderaprogramar.es/index.php?option=com_attachments&task=download&id=285 >
- [9] - Wikipedia-GNU Compiler Collection- [Agosto 2015]
< https://es.wikipedia.org/wiki/GNU_Compiler_Collection >
- [10] - Universidad Carlos III de Madrid- [Noviembre 2011]
< http://www.it.uc3m.es/abel/as/DSP/L2/GCC_es.html >
- [11] - GDB - [Octubre 2014]
< <http://sourceware.org/gdb/documentation/> >
- [12] - Valgrind - [Noviembre 2014]
< <http://valgrind.org/> >
- [13] - ISO/IEC 9126 - [Noviembre 2000]
< <http://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-1%20Standard.pdf> >
- [14] - Wikipedia-información sobre herramientas - [Noviembre 2013]
< https://es.wikipedia.org/wiki/Informaci%C3%B3n_sobre_herramientas >