Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Doctorado en Software, Sistemas y Computación

Tesis doctoral

**Programadores con dislexia:
programación visual frente a la programación
textual**

Autor: Luis Fernando González Alvarán

Directores: José Luis Fuertes Castro
Doctor en Informática

Loïc Antonio Martínez Normand
Doctor en Informática

Madrid, octubre 2019

A Dios que me ha dado la fortaleza para sacar adelante mis proyectos y, que, en los momentos de éxito, felicidad, angustia y soledad, ha sido el verdadero compañero y aliciente de mi existir.

A la memoria de mi madre Lucila quien, con sus esfuerzos y enseñanzas, hizo de mí un hombre honesto, responsable, alegre y soñador.

A mi hermosa familia, que siempre han creído en mí y a pesar de las diferencias, me han apoyado incondicionalmente para salir victorioso ante las adversidades de la vida.

A Óscar que es mi soporte, fortaleza e inspiración.

AGRADECIMIENTOS

Quiero expresar mis más sinceros agradecimientos a:

D. José Luis Fuertes y D. Loïc Martínez por ser los orientadores en este proceso de formación investigativa, y por sus grandes aportes para el desarrollo de mi tesis.

A Claudia Colorado, la más noble y hermosa persona que he conocido y que apareció en mi vida en el momento que más la necesitaba, convirtiéndose en mi amiga y mi ángel protector en España.

Al Politécnico Colombiano Jaime Isaza Cadavid, por el apoyo brindado, no solo en este proceso, si no durante los últimos 15 años, ayudando a mi crecimiento personal, académico y profesional.

A los amigos que creyeron en mí. En especial a Jorge Vallejo e Iván Martínez quienes, con su apoyo y compañía, hicieron más agradable mi estadía en España.

A mi familia, en especial a mi hermana Marta quien, con sus palabras de motivación, siempre alivianó todo este proceso, ayudándome a llevarlo a feliz término.

A Óscar Bedoya, por su tiempo, paciencia y enseñanzas, por su valiosa asesoría en los análisis estadísticos.

A Alexánder Grajales Vanegas, estudiante de trabajo de grado del Politécnico Colombiano Jaime Isaza Cadavid de la ciudad de Medellín (Colombia), quien gracias a su aporte hizo posible el diseño del prototipo que demuestra la posibilidad de transformación de código entre los lenguajes de programación seleccionados.

RESUMEN

Cuando los programadores de ordenadores con dislexia trabajan con un lenguaje de programación textual, presentan múltiples dificultades, que se reflejan en posibles diferencias de desempeño con relación a otros programadores. Éstas se acentúan cuando el desarrollo de software se hace en equipos, convirtiéndose en un problema de eficiencia y calidad.

El objetivo de esta Tesis Doctoral es establecer las diferencias entre programadores con y sin dislexia en el desarrollo de software bajo un lenguaje de programación textual y otro visual, y para ello se ha realizado un estudio analítico en 315 programadores hispanoparlantes de computadores, 2 experimentos entre programadores colombianos con y sin dislexia, y diversos análisis exploratorios que permitieron establecer las diferencias de desempeño, de comportamiento y de satisfacción de los programadores y de las medidas de calidad de software al usar el lenguaje de programación visual Alice, comparado con el lenguaje Java.

Los hallazgos de la investigación han permitido corroborar que la programación visual puede facilitar los procesos de desarrollo de software en los programadores de computadores con dislexia y a la vez, justifican la propuesta de inclusión de un lenguaje de programación con las características de Alice, como adecuado para este grupo poblacional y para ser tenido en cuenta en futuros trabajos que ayuden a mejorar los procesos de integración entre programadores con y sin dislexia, aumentando los índices de eficiencia y calidad del software desarrollado.

PALABRAS CLAVE

Dislexia, lenguajes de programación visual, integración de programadores, lenguaje Alice, lenguajes de programación textual

ABSTRACT

Programmers with dyslexia face several difficulties when using textual programming languages. These difficulties could reduce their performance when compared to the performance of programmers without dyslexia. And these differences can be accentuated when software development is a team-effort, implying efficiency and quality problems.

The goal of this PhD Thesis is to identify the differences between programmers with or without dyslexia when they use a textual programming language or a visual programming language. To this end, several studies have been performed. First, an analytical study was conducted on 315 Spanish-speaking computer programmers. Then, two experiments were conducted, where programmes solved programming exercises with two selected programming languages (Java as textual language and Alice as visual language). These experiments measured the programmers' performance and behaviour, the quality of the resulting programs, and the satisfaction of the programmers using each of the programming languages.

The findings of the research have corroborated that visual programming can facilitate software development processes for programmers with dyslexia. At the same time, they justify the proposal to include a visual programming language with the characteristics of Alice, as appropriate for this population group. Such a visual language should be considered in future work to help improve the integration process between programmers with and without dyslexia, increasing programmers' performance and the quality of the software developed.

KEYWORDS

Dyslexia, Visual programming languages, Integration of programmers, Alice language, textual programming languages

Tabla de contenidos

Pág.

1	Introducción.....	1
2	Estado de la cuestión.....	3
2.1	Dislexia	3
2.1.1	Prevalencia de la dislexia	3
2.1.2	Características de los adultos con dislexia	4
2.1.3	Ayudas informáticas para las personas con dislexia.....	5
2.1.4	Dytective: herramienta para detección de la dislexia	6
2.1.5	La dislexia en los desarrolladores de software	7
2.2	Programación visual	8
2.2.1	El concepto de la programación visual	8
2.2.1	Características relevantes en los lenguajes visuales	9
3	Planteamiento del problema	13
3.1	Pregunta de investigación.....	13
3.2	Hipótesis	13
3.3	Objetivos	14
3.3.1	Objetivo general	14
3.3.2	Objetivos específicos	14
3.4	Metodología	14
4	Características de los programadores con dislexia	17
4.1	Variables Estudiadas.....	17
4.1.1	Variables específicas	17
4.1.2	Variables sociodemográficas.....	17
4.1.3	Variables de comportamiento	17
4.1.4	Variables de programación.....	20
4.2	Resultado de la encuesta.....	21
4.2.1	Muestra	21
4.2.2	Análisis de variables sociodemográficas	22
4.2.3	Análisis de variables específicas	23
4.2.4	Análisis de variables de comportamiento.....	25
4.2.5	Análisis de variables de programación.....	28

4.3	Resultado: caracterización de PCD	33
5	Lenguaje visual para programadores con dislexia.....	35
5.1	Clasificación de lenguajes de programación visual.....	35
5.1.1	Alice.....	35
5.1.2	AppInventor	36
5.1.3	Ark.....	36
5.1.4	Blockly.....	36
5.1.5	Code	37
5.1.6	Cube.....	37
5.1.7	Drakon.....	37
5.1.8	Fabrik	37
5.1.9	Forms/3.....	38
5.1.10	G.....	38
5.1.11	GameSalad.....	38
5.1.12	Max.....	39
5.1.13	Netlogo.....	39
5.1.14	Nxt-g.....	39
5.1.15	Kodu.....	39
5.1.16	Lava.....	40
5.1.17	Limnor	40
5.1.18	Prograph	40
5.1.19	Pure data	41
5.1.20	Quartz Composer	41
5.1.21	Robolab.....	41
5.1.22	Scratch	42
5.1.23	Simulink	42
5.1.24	Snap!	42
5.1.25	Stencyl.....	43
5.1.26	Subtext	43
5.1.27	ToonTalk.....	43
5.1.28	Agilent Vee Pro.....	44
5.1.29	VIPR.....	44
5.1.30	VISSIM.....	44
5.1.31	VVVV	44
5.1.32	Resultado de la clasificación de los 31 lenguajes	45

5.2	Caracterización de los principales lenguajes de programación visual.....	46
5.2.1	Descripción de los lenguajes seleccionados	47
5.2.2	Características Técnicas.....	52
5.2.3	Características facilitadoras del lenguaje de programación visual	52
5.3	Selección de lenguaje visual	53
6	Desempeño, satisfacción y efectividad de los PCD y PSD	54
6.1	Clasificación de los lenguajes de programación textual más usados	54
6.2	Experimento 1. Experiencias de desempeño, satisfacción y efectividad.....	54
6.2.1	Diseño del experimento	55
6.2.2	Desarrollo del experimento.....	57
6.2.3	Análisis estadístico.....	59
6.3	Experimento 2. Experiencias de desempeño, satisfacción, efectividad y medidas de calidad	63
6.3.1	Diseño experimental	63
6.3.2	Desarrollo del experimento.....	65
6.3.3	Análisis estadístico.....	69
6.4	Comparación de desempeño de los programadores.....	73
7	Estrategia para la integración de programadores con y sin dislexia.....	75
7.1	Análisis de particularidades y relaciones de Alice y Java	75
7.1.1	Alice	75
7.1.2	Java	77
7.1.3	Equivalencias entre Alice y Java	80
7.2	Propuesta de integración	81
7.2.1	Algoritmo para la conversión de Alice a Java.....	86
7.2.2	Desarrollo del prototipo.....	87
8	Conclusiones y trabajos futuros.....	89
8.1	Conclusiones.....	89
8.1.1	OE1 “Determinar las principales características y dificultades de los programadores con dislexia, cuando desarrollan software, a partir del análisis de su comportamiento físico, emocional y conductual”	89
8.1.2	OE2 “Construir una taxonomía de la programación visual que permita la selección de un lenguaje apropiado para los programadores con dislexia”	90
8.1.3	OE3 “Determinar los niveles de desempeño, satisfacción y efectividad de programadores con y sin dislexia en el uso de un lenguaje visual y uno textual”	90
8.1.4	OE4 “Proponer una estrategia que facilite la integración de programadores con dislexia”	90
8.2	Trabajo futuro	91

8.2.1	Caracterización de PCD angloparlantes	91
8.2.2	Estudio de desempeño, satisfacción y calidad	91
8.2.3	Lenguaje de programación Visual para PCD.....	91
9	Referencias bibliográficas	93
10	Anexo A. Encuesta experimento 1.	98
11	Anexo B. Encuesta experimento 2.	104
12	Anexo C. Compendio de resultados arrojados por el software SPSS, estudio para la definición del perfil de los programadores con dislexia.....	110
12.1	Variables de comportamiento (Emocionales).....	110
12.2	Variables de comportamiento (conductuales).....	115
12.3	Variables de comportamiento (Físicas).....	124
13	Anexo D. Compendio de resultados arrojados por el software SPSS, Experimento 1.....	126
13.1	Análisis descriptivo.....	126
13.2	Análisis cruzado	129
14	Anexo E. Compendio de resultados arrojados por el software SPSS, Experimento 2.....	131
14.1	Tablas cruzadas	131
14.2	Pruebas no paramétricas. Tablas cruzadas.....	132

Índice de Tablas

Pág.

Tabla 1. Resumen de actividades y resultados	15
Tabla 2. Variables específicas.....	17
Tabla 3. Variables emocionales	18
Tabla 4. Variables conductuales	18
Tabla 5. Variables físicas.....	20
Tabla 6. Variables de programación	20
Tabla 7. Frecuencia nacionalidad por Género.....	22
Tabla 8. Frecuencia de Edades.....	23
Tabla 9. Prevalencia de dislexia.....	23
Tabla 10. Frecuencia presencia de dislexia.....	24
Tabla 11. Diagnóstico de dislexia	24
Tabla 12. Frecuencia Tratamiento de dislexia	24
Tabla 13. Frecuencia afección de la dislexia en el desempeño	24
Tabla 14. Variables emocionales: porcentajes más relevantes.	26
Tabla 15. Variables conductuales: porcentajes más relevantes.	26
Tabla 16. Variables físicas: porcentajes más relevantes.....	27
Tabla 17. Asociación estadísticamente significativa.....	27
Tabla 18. Frecuencia de uso de los LPV más relevantes	29
Tabla 19. Frecuencia de uso de los LPT reportados	29
Tabla 20. Porcentajes más relevantes variables de programación.....	30
Tabla 21. Principales dificultades de los PCD a la hora de programar	32
Tabla 22. Clasificación de los lenguajes de programación visual	45
Tabla 23. Resumen de clasificación de los LPV.....	46
Tabla 24. Resumen. Características técnicas de los LPV seleccionados	52
Tabla 25. Resume Características facilitadoras para PCD en los LPV	52
Tabla 26. Frecuencia de uso de los lenguajes de programación textual	54
Tabla 27. Variables generales.....	55
Tabla 28. Variables de eficiencia	56
Tabla 29. Variables de satisfacción	57
Tabla 30. Medidas de tendencia central MTC. Tiempo de desarrollo.....	60
Tabla 31. Resumen de programas finalizados y no finalizados	60

Tabla 32. Medidas de tendencia central. Número de Errores	60
Tabla 33. Porcentaje de precisión	61
Tabla 34. Variables de Calidad	63
Tabla 35. Etapas para el desarrollo de la fase dos	65
Tabla 36. Actividades para el desarrollo de la fase 3.....	65
Tabla 37. Comparativo de satisfacción. Experimento 1 y 2	73
Tabla 38. Prueba de T pareada. Satisfacción Java y Alice PCD	73
Tabla 39. Prueba de Wilcoxon Satisfacción Java Alice. PSD	74
Tabla 40. Operadores Java.....	77
Tabla 41. Correspondencia estructuras de asignación y ramificación.....	80
Tabla 42. Correspondencia Tipos de datos	80
Tabla 43. Correspondencia de estructuras	80
Tabla 44. Correspondencia de control de flujo.....	81
Tabla 45. Comportamiento Emocional (Ansiedad_P)	110
Tabla 46. Comportamiento Emocional (Angustia)	110
Tabla 47. Comportamiento Emocional (Rabia)	111
Tabla 48. Comportamiento Emocional (Concentración)	111
Tabla 49. Comportamiento Emocional (Dispersión)	112
Tabla 50. Comportamiento Emocional (Humor)	112
Tabla 51. Comportamiento Emocional (Depresión)	113
Tabla 52. Comportamiento Emocional (Tristeza)	113
Tabla 53. Comportamiento Emocional (Irritabilidad)	114
Tabla 54. Comportamiento Emocional (Indefenso).....	114
Tabla 55. Comportamiento Emocional (Autoestima)	115
Tabla 56. Comportamiento Emocional (Desconfianza)	115
Tabla 57. Comportamiento Conductual (Excusa_E).....	116
Tabla 58. Comportamiento Conductual (Excusa_T)	116
Tabla 59. Comportamiento Conductual (Atención)	116
Tabla 60. Comportamiento Conductual (Hipoactividad).....	117
Tabla 61. Comportamiento Conductual (Hipoactividad_P)	118
Tabla 62. Comportamiento Conductual (Agresividad)	118
Tabla 63. Comportamiento Conductual (Agresividad_P)	119
Tabla 64. Comportamiento Conductual (Retraído).....	119
Tabla 65. Comportamiento Conductual (Retraído_P)	119
Tabla 66. Comportamiento Conductual (Compulsivo).....	120

Tabla 67. Comportamiento Conductual (Compulsivo_P)	120
Tabla 68. Comportamiento Conductual (Aislamiento).....	121
Tabla 69. Comportamiento Conductual (Aislamiento_P)	121
Tabla 70. Comportamiento Conductual (Hiperactividad)	122
Tabla 71. Comportamiento Conductual (Hiperactividad_P).....	122
Tabla 72. Comportamiento Conductual (Ansiedad_G)	123
Tabla 73. Comportamiento Conductual (Ansiedad_H)	123
Tabla 74. Comportamiento Física (Insomnio)	124
Tabla 75. Comportamiento Física (Madrugar).....	124
Tabla 76. Comportamiento Física (Migrañas)	124
Tabla 77. Comportamiento Física (Visión_D)	125
Tabla 78. Comportamiento Física (Fatiga)	125
Tabla 79. Frecuencias. Género	126
Tabla 80. Frecuencias. Nivel_Educación.....	126
Tabla 81. Frecuencias. Dytective	126
Tabla 82. Frecuencias. Nivel_J	126
Tabla 83. Frecuencias. Termina_J.....	126
Tabla 84. Frecuencias. Conocimiento_J	127
Tabla 85. Frecuencias. Termino_A	127
Tabla 86. Frecuencias. Precisión_J	127
Tabla 87. Frecuencias. Nro_Errores_Java_1	127
Tabla 88. Frecuencias. Precisión_Alice	128
Tabla 89. Frecuencias. Nro_Errores_Alice_1	128
Tabla 90. Medidas de tendencia central. Edad	128
Tabla 91. Medida de tendencia central. Nro_Errores_Alice	128
Tabla 92. Medidas de tendencia central. Tiempo_Desarrollo_J	128
Tabla 93. Medida de tendencia central. T_Desarrollo_Alice.....	129
Tabla 94. Medida de tendencia central. Satisfacción_J.....	129
Tabla 95. Medida de tendencia central. Satisfacción_A.....	129
Tabla 96. Resultados. Angustia_J	129
Tabla 97. Resultados. Concentración_J.....	130
Tabla 98. Resultados. Sintaxis_J.....	130
Tabla 99. Resultados. Angustia_A	130
Tabla 100. Resultados. Concentración_A	130
Tabla 101. Resultados. Sintaxis_A	130

Tabla 102. Tabla cruzada Dislexia - Angustia_J	131
Tabla 103. Tabla cruzada Dislexia - Angustia_A.....	131
Tabla 104. Tabla cruzada Dislexia - Concentración Java.....	131
Tabla 105. Tabla cruzada Dislexia – Resumen Concentración Java.....	131
Tabla 106. Tabla cruzada Dislexia - Concentración Alice	132
Tabla 107. Tabla cruzada Dislexia - Sintaxis_J	132
Tabla 108. Tabla cruzada Dislexia - Sintaxis_A.....	132
Tabla 109. Prueba no paramétrica Angustia_J	132
Tabla 110. Recuento prueba no paramétrica Angustia_J.....	133
Tabla 111. Pruebas de chi-cuadrado Angustia_J.....	133
Tabla 112. Prueba no paramétrica Angustia_A.....	133
Tabla 113. Recuento prueba no paramétrica Angustia_A	133
Tabla 114. Pruebas de chi-cuadrado Angustia_A.....	133
Tabla 115. Prueba no paramétrica Concentracion_J	134
Tabla 116. Resumen prueba no paramétrica Concentración_J.....	134
Tabla 117. Pruebas de chi-cuadrado Concentración_J	134
Tabla 118. Prueba no paramétrica Concentracion_A	134
Tabla 119. Pruebas de chi-cuadrado Concetración_A.....	135
Tabla 120. Resumen prueba no paramétrica Concentración_A.....	135
Tabla 121. Pruebas de chi-cuadrado Sintaxis_J	135
Tabla 122.....	135
Tabla 123. Prueba no paramétrica Sintaxis_A.....	135
Tabla 124. Pruebas de chi-cuadrado Sintaxis_A.....	135

Índice de Figuras

Pág.

Figura 1. Relación Dificultades / Problemas de la Dislexia. Fuente: (Alsobhi, 2014)	5
Figura 2. Informe Dytective. Parte 1	7
Figura 3. Informe Dytective. Parte 2	7
Figura 4. Captura de pantalla de Pygmalion. Fuente: (Smith, 1975).....	9
Figura 5. Frecuencia según género.....	22
Figura 6. Distribución por género PCD.....	25
Figura 7. Variables comportamentales relevantes en la caracterización de PCD. Grado desacuerdo	28
Figura 8. Variables comportamentales relevantes en la caracterización de PCD. Grado de acuerdo	28
Figura 9. Frecuencia de uso de Lenguajes de Programación Textual	30
Figura 10. Análisis de frecuencia, variables de programación.....	31
Figura 11. Distribución demográfica. Principales dificultades de los PCD	32
Figura 12. Interfaz de Scratch	48
Figura 13. Interfaz principal de Kodu	49
Figura 14. Interfaz de Snap!.....	50
Figura 15. Interfaz de AppInventor.....	51
Figura 16. Interfaz Alice	51
Figura 17. Nivel de satisfacción lenguaje Alice. Experimento 1.....	62
Figura 18. Nivel de satisfacción lenguaje Java. Experimento 1	62
Figura 19. Nivel de satisfacción lenguaje Java. Experimento 2	71
Figura 20. Nivel de satisfacción lenguaje Alice. Experimento 2	71
Figura 21. Diagrama de componentes de Alice.	75
Figura 22. Diagrama de relaciones de Alice.	77
Figura 23. Diagrama de relaciones de Java	79
Figura 24. Diagrama de componentes de Java	79
Figura 25. Diagrama de casos de uso.....	82
Figura 26: Estructura inicial de un programa en Alice	82
Figura 27. Apariencia del XML	83
Figura 28. Modelo del XML	84
Figura 29. Modelo del JSON.....	85
Figura 30. Primer análisis con GitHub	85

Figura 31. Segundo análisis con GitHub	86
Figura 32. Diagrama de para la conversión	86
Figura 33. Diagrama de flujo conversión	87
Figura 34. Interfaz inicial del prototipo	88
Figura 35. Interfaz subir archivo	88

1 Introducción

Según la Organización Mundial de Salud (OMS), más de mil millones de personas en todo el mundo viven con algún grado de “discapacidad¹” (en la presente Tesis se hará uso principalmente del término diversidad funcional, que fue introducido en España en sustitución de otros con semántica peyorativa como “discapacidad”, “minusvalía”, etc. (Romañach, 2005)). En los años futuros, el número de personas con diversidad funcional será un motivo de preocupación aún mayor, pues su prevalencia va en aumento a medida que va envejeciendo la población (OMS, 2011).

La “Convención sobre los Derechos de las Personas con Discapacidad”, aprobada el 13 de diciembre de 2006 en la sede de las Naciones Unidas en Nueva York, tiene como propósito: “promover, proteger y asegurar el goce pleno y en condiciones de igualdad de todos los derechos humanos y libertades fundamentales por todas las personas con discapacidad, y promover el respeto de su dignidad inherente” (UN, 2006). Organismos internacionales están promoviendo y ven con buenos ojos, los procesos de ayuda digital que brindan las Tecnologías de la Información y la Comunicación (TIC) como una herramienta para disminuir la brecha de exclusión social que dicha situación genera en una comunidad particular.

Esta Tesis Doctoral se centra en un tipo concreto de diversidad funcional, la dislexia, que se puede describir como:

“Una dificultad del aprendizaje que tiene un origen neurológico. Se caracteriza por dificultades en un correcto o fluido reconocimiento de palabras y por un delecteo y habilidades decodificadoras pobres [...]. Algunas consecuencias secundarias pueden incluir problemas en la comprensión lectora y una experiencia en lectura reducida, que puede impedir el crecimiento del vocabulario y el aprendizaje de otros conocimientos.” (International Dyslexia Association, 2002).

Alsobhi afirma que la dislexia es la discapacidad con mayor número creciente de pacientes diagnosticados por año y que datos estadísticos comparativos entre los periodos 1994/1995 y 2000/2001, demuestran que éstos aumentaron en un 100%. Por otro lado, afirma que el porcentaje más elevado por grupo poblacional es el de los estudiantes con dislexia y que el sistema educativo, en especial la educación superior, no está preparado para atender las necesidades específicas de éstos (Alsobhi, 2013).

La prevalencia de la dislexia a nivel mundial va en ascenso. Su falta de diagnóstico o intervención oportuna y permanente se ve reflejada en el impacto negativo que ésta puede generar en diversas áreas del conocimiento. Específicamente, en el desarrollo de software, ahora que está tomando fuerza la tendencia de trabajo conocida como “*pair programming*” (Salge, 2016), la cual requiere de dos personas trabajando en conjunto por un objetivo común, es posible encontrar que uno de los programadores del equipo presente síntomas de dislexia.

Los efectos de la dislexia en los desarrolladores de software se reflejan muchas veces en aspectos positivos como la capacidad de análisis que pueden llegar a tener estos individuos. Sin embargo, a menudo, los programadores con dislexia (PCD a partir de ahora) se ven opacados por estados emocionales y motivacionales originados ante la

¹ Limitación física o limitación cognitiva ya sea por enfermedad, accidente o congénito (Organización Mundial de la Salud, 2011).

incapacidad de poder realizar correctamente ciertas tareas como cumplir normas de sintaxis o la definición de identificadores. Estos aspectos están relacionados con las características específicas de la dislexia relacionadas con la dificultad para el correcto deletreo y limitan a los PCD a la hora de codificar un programa en un lenguaje de programación tradicional. Dado que investigaciones anteriores han planteado que el uso de lenguajes de programación visual podría ayudar a minimizar los efectos negativos de la dislexia, en la presente Tesis se hace un análisis de las características de los PCD, asociadas a las peculiaridades de la programación visual, con el fin de afianzar la idea de que ésta pueda facilitar los procesos de desarrollo de software en este grupo poblacional.

Por todo lo anterior, y teniendo en cuenta que los estudios científicos relacionados con la presencia de la dislexia en los programadores de computadores son muy escasos, se hace necesario abordar esta temática y proponer alternativas de soluciones, que ayuden a mitigar los efectos negativos de la dislexia en el desarrollo de software.

En la presente Tesis, a partir de un análisis de los lenguajes de programación visual (LPV) más comunes, se realizan estudios que establezcan las diferencias de desempeño, comportamentales y de satisfacción de programadores con y sin dislexia, para poder establecer alguna estrategia de integración que ayude a éstos en los procesos de desarrollo de software en equipo.

2 Estado de la cuestión

2.1 Dislexia

Según Pedro Felipe Monlau, la palabra dislexia está formada por los prefijos griegos $\delta\upsilon\varsigma$ (dis- = con dificultad), $\lambda\acute{\epsilon}\xi\iota\varsigma$ (lexis = palabra) y el sufijo -ia (-ia = cualidad) (Monlau, 1856). De acuerdo con lo expuesto por Rudolf Wagner, quien hizo un estudio acerca de los orígenes de la dislexia, el término fue usado por primera vez por el médico alemán Oswald Berkhan (1834-1917) quien lo clasificó como una condición llamada “*Wordblindheit*” (ceguera para las palabras). Posteriormente el oftalmólogo Rudolf Berlín (1833-1897) formalizó el concepto al referirse a una persona que manifestaba incapacidad para deletrear y describir palabras, pero cuyo resto de facultades mentales eran normales (Wagner, 1973).

La Organización Mundial de la Salud concibe la dislexia como:

“Un trastorno específico de la lectura donde la capacidad para la comprensión lectora, el reconocimiento de palabras escritas, la lectura oral y la realización de tareas escritas, están afectadas. Comúnmente, los trastornos del desarrollo específico de la lectura van acompañados de una historia previa de alteraciones en el desarrollo del habla y del lenguaje, siendo frecuentes la asociación de problemas emocionales y de la conducta durante todo el periodo escolar (Organización Mundial de la Salud, 2011).”

2.1.1 Prevalencia de la dislexia

A pesar del carácter universal de la dislexia, sus estimaciones de prevalencia son variables. Estas variaciones son atribuidas en su mayoría a la diversidad ortográfica de cada idioma.

Mientras que Katusic afirma que las tasas de prevalencia mundial varían desde el 5,3% hasta el 11,8% (Katusic, 2001), Lyon establece –específicamente para los Estados Unidos– que, entre el 15% y el 20% de la población tienen una dificultad de aprendizaje basada en el lenguaje, de las cuales, el 80% corresponde a dislexia (Lyon, 2003). Por otro lado, Everatt afirma que entre el 5% y el 10% de la población adulta mundial tiene dislexia (Everatt, 2004).

En España, son pocos los estudios que dan una estadística fiable al respecto, pero se dice que la incidencia de dislexia en la educación primaria y secundaria se sitúa entre el 5% y 10% (American Psychiatric Association, 2013). No obstante, específicamente en Murcia se reportó que un 11,8% de los estudiantes se encuentran en esta condición (Carrillo, 2011), mientras que en las Islas Canarias se presenta un 8,6% (Jiménez, 2009).

En Latinoamérica, específicamente en Colombia, no se tienen cifras exactas; sin embargo, en la ciudad de Barranquilla se reportó en la población infantil un porcentaje de 3,3% (De los Reyes, 2008). En Bogotá, en un estudio realizado en las escuelas de la localidad de Ciudad Bolívar, se obtuvo un porcentaje del 3,6% (Pardo, 2015).

Más recientemente, Rello y Baeza han recopilado datos estadísticos sobre la presencia de la dislexia en la población mundial, dejando claro que difiere en porcentaje dependiendo del idioma. En su estudio, establecen que en los hispanohablantes los datos de prevalencia oscilan entre el 7,5 % y 11,8% dependiendo del país. Sin

embargo, estas estimaciones se basan en pruebas de lectura y no de escritura (Rello, 2012), por lo que la prevalencia podría ser mayor.

2.1.2 Características de los adultos con dislexia

Durante años, la dislexia ha sido un trastorno del aprendizaje oculto, y la falta de un manejo adecuado o de una intervención oportuna, la ha convertido en una diversidad funcional supuestamente poco relevante para los centros de formación de adultos, pero con gran afectación en el desarrollo personal de los implicados.

Según Ingram, la dislexia afecta a cada persona de forma diferente, concluyendo que las necesidades y experiencias de aprendizaje son diferentes. Por lo tanto, afirma que es casi imposible hacer generalizaciones al establecer las características de los adultos con dislexia. Sin embargo, Ingram realizó un primer acercamiento a partir de los diferentes problemas que caracterizan a los niños con dislexia, agrupándolos en 6 áreas:

- Problemas de lectura
- Problemas de escritura
- Problemas en el habla
- Problemas de memoria
- Problemas organizativos
- Problemas con las matemáticas.

A partir de la anterior clasificación y teniendo en cuenta que los síntomas de cada problema se manifiestan de forma diferente, Ingram estableció 3 categorías que disponen los principales tipos de dificultades que pueden manifestar las personas con dislexia (Ingram, 1964):

- Dificultades visoespaciales
- Dificultades de sonido y del habla
- Dificultades de correlación.

En torno a esta relación, Haber ha realizado un estudio que determina ciertas características de las personas adultas con dislexia, que se resumen en diversas manifestaciones que se pueden clasificar por síntomas (Haber, 2012): emocionales, conductuales y físicos.

- **Síntomas Emocionales.** Están agrupados en esta categoría los siguientes síntomas: ansiedad, depresión, baja autoestima, angustia, tristeza, rabia, indefensión, irritabilidad, cambios frecuentes de humor, falta de concentración, dispersión, desconfianza.
- **Síntomas Conductuales.** Están agrupados en esta categoría los siguientes síntomas: conductas de evasión para acudir a la escuela, trabajo, etc., conductas para llamar la atención, amenaza de suicidio, hiperactividad causada por la ansiedad, hipo actividad (no tener ganas de nada), agresividad, retraimiento, autoaislamiento, compulsiones, tics y manías.
- **Síntomas Físicos.** Están agrupados en esta categoría los siguientes síntomas: debilitación del sistema inmunológico, problemas gastrointestinales, taquicardias, sudoración excesiva, tensión muscular, insomnio, dificultad para levantarse por las mañanas, continuos dolores de cabeza, migrañas, visión desordenada, fatiga, dificultad para tragar, problemas respiratorios (asma), alergias.

En la Figura 1 se puede observar la relación entre problemas y dificultades.

Figura 1. Relación Dificultades / Problemas de la Dislexia. Fuente: (Alsobhi, 2014)

2.1.3 Ayudas informáticas para las personas con dislexia

Algunos estudios demuestran que en el Reino Unido se ofrece el apoyo legislativo más importante para las personas con dislexia y esto se debe a que el porcentaje de estudiantes con diversidad funcional más elevado es el de los que padecen de este trastorno. Ahora bien, tanto en Inglaterra como en el resto del mundo, el sistema educativo no está preparado para atender las necesidades específicas de este grupo poblacional, razón por la cual se ha buscado apoyo en diversas herramientas informáticas que tratan de mejorar las condiciones y calidad de vida de los pacientes.

Son diversas las investigaciones que se han realizado alrededor del mundo con miras a proveer herramientas que ayuden a las personas con dislexia. No obstante, el problema de estas herramientas va más allá de la accesibilidad y del diseño Web. Se afirma que las herramientas informáticas pueden beneficiar a personas con otras diversidades, proporcionando la experiencia de diferentes estilos de aprendizaje, pero al mismo tiempo pueden ayudar a crear barreras severas en los estudiantes con dislexia. Lo anterior en gran parte se debe a que esta condición afecta a las personas de manera diferente. En raras ocasiones, dos personas con dislexia tienen los mismos síntomas. La mayoría de los estudiantes que la tienen, experimentan diversas formas de aprendizaje y necesidades, no obstante, la dislexia va asociada a un coeficiente intelectual normal o por encima de la media de referencia (Alsobhi, 2013).

Según Kalyvioti y Mikropoulos, las principales ayudas informáticas relacionadas con la dislexia, se enfocan en tres áreas que tienen como objetivo la evaluación, intervención y sensibilización. Sin embargo, la mayoría de éstas se enfocan en la intervención, específicamente en optimizar los procesos de escritura y lectura (Kalyvioti, 2014).

Diversas investigaciones indican que la forma de presentación del texto puede tener un impacto en el rendimiento de la lectura en las personas con dislexia. Entre los estudios anteriores se puede destacar el trabajo de (Gregor, 2000) y el de (Kurniawan, 2006). Ambos trabajos demostraron que la posibilidad de personalizar el diseño del texto mejoraba subjetivamente la legibilidad de las personas con dislexia.

Otros estudios relacionados con la forma de visualizar el texto se centraron en:

- *Tamaño de la letra.* O'Brien afirma que éste puede bajar el rendimiento a la hora de leer. Para los niños con dislexia, el tamaño crítico a partir del cual se reporta disminución en el rendimiento es más grande que para los no disléxicos (O'Brien, 2008).
- *Espaciado de la letra.* Zorzi afirma que un espaciado mayor en los textos conlleva a un menor número de errores y una mayor velocidad de lectura en niños con dislexia (Zorzi, 2012).
- *Longitud de línea* en dos dimensiones de pantalla. Los investigadores encontraron que el uso de un terminal pequeño mejoró la legibilidad. También afirman que en cuanto al espaciado extra mejora la comprensión en aquellos lectores con más dificultades (Schneps, 2013).

Dado lo anterior se han desarrollado diversas aplicaciones que proponen solucionar dichos aspectos, destacándose las siguientes:

- *Theme Font & Size changer* Extensión que permite a los usuarios con dislexia personalizar la fuente y el tamaño del texto de la interfaz de usuario de Firefox (Saunders, 2017).
- *Penfriend XL.* Predice la siguiente palabra que las personas con dislexia quieren escribir (Pendfriend Ltd, 2018).
- *ScreenRuler Suite.* Parte la pantalla en dos zonas para destacar o subrayar, cambiar color de fondo y contraste. Ayuda al proceso de lectura a través de una regla de superposición (ClaroSoftware, 2014).

Otras ayudas son las aplicaciones de lectura que utilizan las tecnologías de conversión de texto a voz, por ejemplo: *ReadingPenTS* (Wizcomtech, 2014) y *Di3* (RehaSoft, 2012).

2.1.4 Dytective: herramienta para detección de la dislexia

A la fecha, solo se conoce una herramienta informática que ayude a la detección temprana de la dislexia. Se trata de un sistema que realiza una prueba en forma de juego mediante la cual realiza un diagnóstico de la posibilidad de que una persona esté en riesgo de padecer dislexia.

El software "*Dytective*", desarrollado por Rello y su grupo de investigación de la *Carnegie Mellon University* es la primera herramienta que integra tecnología puntera de aprendizaje automático usando Redes Neuronales Recurrentes y juegos informáticos con una fiabilidad del 90%. La prueba, que dura 15 minutos, analiza un total de 197 variables relacionadas con el rendimiento del usuario ante una serie de ejercicios lingüísticos y de atención relacionados con manifestaciones de la dislexia.

Una vez realizada la prueba, se emite un informe donde se especifica si hay riesgo de tener dislexia con más de 90% de precisión (Change Dyslexia, 2016).

En la Figura 2 se observa el informe del resultado que se le entrega a la persona que se sometió a la prueba.

En la Figura 3, se puede observar un resumen gráfico del rendimiento que esta persona tuvo en la prueba. "*Dytective*", además, ofrece una amplia categoría de actividades y ejercicios que mejoran de forma científica la competencia lingüística de los estudiantes con dificultades de aprendizaje (Rello, 2016).

Fecha: 11/01/2017

INFORME DE RESULTADOS PARA: LUIS FERNANDO

SUPERVISOR: LUIS FERNANDO GONZALEZ ALVARAN

Riesgo de tener dificultades de lectura y escritura:

RIESGO ALTO

Nota: Esto **no** es un diagnóstico de dislexia, es sólo un test de riesgo de dificultades de lecto-escritura. El resultado de Dyetective¹ for Samsung puede ser uno de estos cuatro: **Sin Riesgo**, **Riesgo Bajo**, **Riesgo Medio** y **Riesgo Alto**. Una persona disléxica detectada a tiempo y que ha sido tratada, si realiza este test, puede dar como resultado Sin Riesgo.

En el gráfico, se presenta la media obtenida por el participante (en **rojo**) comparada con la correspondiente al desempeño de las personas de su edad (en **azul**), en función de las medidas de rendimiento que se han tomado a partir de las tareas evaluadas: comprensión lectora, conocimiento de la ortografía, velocidad de lectura, velocidad de escritura y capacidad de reconocimiento y de corrección de errores.

Figura 2. Informe Dyetective. Parte 1

Figura 3. Informe Dyetective. Parte 2

2.1.5 La dislexia en los desarrolladores de software

Son pocos los estudios que se conocen con relación a las incidencias de la dislexia en la capacidad de desarrollar software o aprender a programar.

Uno de los más relevantes, corresponde al realizado por Powell, en el cual el investigador relaciona las características de la dislexia con las habilidades que se requieren para ser un programador eficiente (Powell, 2004). En este estudio, Powell hizo uso del modelo de diseño de software planteado por Shigley y Mitchell, que establece los pasos para el diseño y resolución de un problema: reconocimiento de

necesidades, definición del problema, síntesis, análisis, implementación y evaluación (Shigley, 1993).

Los investigadores resaltaron las principales necesidades de los PCD relacionadas con ciertas etapas del modelo de desarrollo de software, y las resume en:

- Tareas relacionadas con la generación y corrección de código. Afirman que los PCD se ven más afectados por los rasgos negativos de su diversidad. Por ejemplo, presentan gran dificultad el manejo de nombres de variables, ya que el cumplimiento de normas de ortografía y diversas reglas de sintaxis, les imponen una carga considerable en la memoria a corto plazo.
- Fase de implementación. Los investigadores establecen que no se puede dejar de lado la correcta estructura del programa, lo cual apunta a requerir habilidades de organización que pueden poner a un PCD en desventaja con sus compañeros.
- Fase evaluación del producto. Se concluye que la principal necesidad está relacionada con las habilidades de organización y planificación aspectos que son deficientes en los PCD.

Powell concluye que, a pesar de que las manifestaciones de la dislexia en cada individuo son diferentes, se puede generalizar que los PCD requieren de una gran cantidad de habilidades lógicas y organizativas, combinadas con una expresión precisa de sintaxis y que éstos a menudo, presentan deficiencia en la destreza comunicativa (Powell, 2004).

Otra investigación realizada por Hammond y Hercules, hace referencia a las diversas habilidades requeridas para ser un buen programador, sobresaliendo la necesidad de dividir el problema en partes o componentes y el manejo de una secuencia lógica. Los investigadores concluyen que estas habilidades se encuentran difícilmente en las personas con dislexia (Hammond, 2013)

2.2 Programación visual

En este apartado se verá una introducción al concepto de programación visual, y una caracterización de los aspectos más relevantes que permiten clasificar los distintos lenguajes de programación visual.

1.1.1 El concepto de la programación visual

El primer acercamiento a la programación visual (PV) lo realizó Smith en 1975 cuando lanzó el proyecto *Pygmalion* (Figura 4), que era un sistema de programación visual bidimensional implementado en un ordenador interactivo con pantalla gráfica (Smith, 1975).

No obstante, el concepto de PV surge realmente en los años 80 partiendo de la necesidad de encontrar formas más sencillas de hacer las tareas de programación. La inclusión del uso de gráficos en los lenguajes de programación fue lo que en un principio se llamó programación gráfica.

Clarisse y Chang afirman que ésta utiliza la información en un formato que está más cerca de las representaciones mentales de los problemas del usuario, y permiten que los datos se procesen en un formato más cercano a la forma en que los objetos son manipulados en el mundo real (Clarisse, 1986). A partir de esta hipótesis surge la idea de una programación visual que sea más fácil de crear y entender, especialmente para los no programadores o para los desarrolladores de software novatos. La programación visual propone un nivel más alto de descripción de las acciones a

realizar, restando importancia a aspectos como la sintaxis y dando disponibilidad a un mayor nivel de abstracción, logrando que la tarea de programación sea mucho más fácil, incluso para programadores profesionales.

Figura 4. Captura de pantalla de Pygmalion. Fuente: (Smith, 1975)

El término programación visual se refiere a cualquier sistema que permite al usuario especificar un programa en dos (o más) dimensiones. La PV no incluye los sistemas que utilizan los lenguajes de programación convencional (lineal o textual) para definir imágenes (Myers, 1986).

La PV se fundamenta en el uso de los lenguajes de programación visual (LPV), que se definen como aquéllos que usan una representación visual (ya sea gráficos, dibujos, animaciones o iconos) para manipular la información, soportar la interacción, expresar relaciones o permitir crear las diferentes expresiones y estructuras de un programa de cómputo. En síntesis, un LPV lleva a cabo todas las tareas de programación de forma visual (Chang, 1986).

2.2.1 Características relevantes en los lenguajes visuales

Existen ciertas características de los LPV que son importantes conocer para el desarrollo de la presente Tesis, debido a que pueden convertirse en un factor diferenciador a la hora de la selección el LPV a incluir en los experimentos.

Estas características se dividen en dos grupos: características técnicas y características facilitadoras para programadores con dislexia. Este segundo grupo se

divide, a su vez, en relacionadas con trasposición de letras, relacionadas con estrés visual y relacionadas con accesibilidad.

2.2.1.1 Características Técnicas

Según Boshernitsan, las siguientes son las principales características que pueden ayudar a establecer la fortaleza de un LPV frente a otro (Boshernitsan, 2004):

- **Control de flujo.** Los lenguajes de programación visuales abarcan dos nociones de control de flujo: imperativo y declarativo. Con el enfoque imperativo, un programa visual constituye uno o más diagramas de control de flujo o de datos que indican cómo fluye el flujo de control a través del programa. Con el enfoque declarativo, sólo hay que preocuparse por los cálculos que se realizan y no por la forma en que se realizan las operaciones reales. La eficacia de un lenguaje de programación depende en gran medida de si el flujo de tiempo de ejecución puede modificarse mediante estructuras de control y bucles.
- **Abstracción procedimental.** Se refiere a la capacidad del lenguaje de descomponer el programa en módulos, funciones o subprogramas. Los LPV normalmente cubren todo el espectro de las facilidades de programación, que van desde características de bajo nivel, incluyendo condicionales, recursividad e iteración, a instalaciones de alto nivel que permiten combinar la lógica de bajo nivel en módulos abstractos (procedimientos, clases, bibliotecas, etc.).
- **Abstracción de datos.** Se refiere a si la técnica de programación puede utilizarse para crear o definir nuevos tipos de datos (establecidos por el usuario) adaptados a la aplicación que se desea implementar. La abstracción en la PV es muy similar a la noción de abstracción de datos en la programación convencional, con el único requisito de que los tipos de datos abstractos se definen visualmente, tienen una representación visual (icónica) y un comportamiento interactivo.

Otros autores establecen características secundarias que pueden ayudar a la selección del mejor LPV y éstas se enuncian a continuación:

- **Lenguaje de implementación.** Esta característica se refiere al lenguaje o lenguajes textuales en los que se desarrolló el LPV.
- **Sistema operativo.** Hace referencia a los sistemas operativos en los que se ejecuta el entorno del lenguaje.
- **Tipo de licencia.** Con la mira de futuras decisiones de desarrollo de proyectos, es importante determinar si el LPV tiene una licencia de código abierto o cerrado.
- **Método de construcción visual.** Se refiere a cómo se crea el programa.
 - Arrastrar y soltar simple. Cuando hay varios tipos de bloques sin muescas que el usuario arrastra y suelta para crear el código.
 - Arrastrar y soltar. Donde la mayoría de los bloques tienen muescas y éstas deben encajar perfectamente como las piezas de un rompecabezas.
 - Visual. Cuando hay un conjunto de iconos que gobiernan tareas específicas de construcción de programas.
- **Sistemas de tipos.** La tipificación de datos está estrechamente relacionada con el tiempo de ejecución del programa, dependiendo de si el sistema de tipos es estático o dinámico. Según Ousterhout, un sistema de tipos dinámico siempre va a representar mayor desempeño (Ousterhout, 1998).

- **Programación concurrente.** Ésta se ha convertido en una medida de eficiencia que debe ser considerada a la hora de determinar la potencia de cualquier desarrollo de software. Si se analizan con detalle los problemas del mundo real, unidos a la potencia, características y arquitecturas de los actuales procesadores, las nuevas herramientas de software deben asumir prácticas de programación basadas en arquitecturas paralelas, que impactarán positivamente en todos los desarrollos. La programación concurrente se refiere a la programación de notaciones y técnicas para expresar el paralelismo potencial y resolver problemas de manera más eficiente (Schneider, 2012).
- **Otro lenguaje.** Esta característica se refiere a si el LPV permite generar un código equivalente en un lenguaje de texto o metalenguaje, ya sea directamente o a través de procesos externos.

2.2.1.2 Características facilitadoras para programadores con dislexia

Acorde con la revisión de literatura, existen ciertas características que pueden ayudar a mejorar el rendimiento de los PCD. Éstas han sido agrupadas en 3 categorías que se describen a continuación.

2.2.1.2.1 Características relacionadas con la transposición de las letras

En los lenguajes de programación visual predomina la imagen, al contrario que en los lenguajes basados en texto (Canfield Smith, 1977). Sin embargo, el texto sigue siendo una característica de muchos LVP; hay 4 aspectos clave relacionados con su uso que merecen ser analizados (se mencionan las siglas que posteriormente se utilizaran en este trabajo para referirse a estos conceptos):

- **Nodos.** Es importante analizar si se requiere mover el ratón o hacer clic en los nodos o elementos de la pantalla para que el software proporcione información al respecto. Los posibles valores que se pueden registrar son: información detallada sobre todos los nodos y elementos (DT), información detallada sobre algunos nodos y elementos (DA), información básica sobre todos los nodos y elementos (BT), información básica sobre algunos nodos y elementos (BA), una combinación de información básica y detallada (CD) y sin información sobre nodos y elementos (ND).
- **Nombres de identificadores.** Esta característica se refiere a la forma en que el LPV gestiona los nombres de constantes, variables y subprogramas. Los posibles valores que se pueden registrar son: el usuario puede definir nombres de identificador (Y) y el usuario no puede definir nombres de identificador (N).
- **Expresiones.** Esta característica se refiere a aspectos relacionados con la definición y uso de fórmulas matemáticas. Los posibles valores que se pueden registrar son: hay iconos para crear expresiones (Y), no hay iconos para crear expresiones y deben ser introducidas como texto (N) y existe un sistema mixto para crear expresiones (iconos y texto) (M).
- **Idioma.** Las personas con dislexia tienen dificultades para interactuar con un idioma que no sea su lengua materna. Basado en lo anterior, éste se convierte en una característica clave para los PCD. Los posibles valores que se pueden registrar son: inglés (E), español (S), otro (A) y multilenguaje (cambio dinámico) (M).

2.2.1.2.2 Características relacionadas con el estrés visual

Uno de los problemas que más ataca a los PCD tiene que ver con el estrés visual que pueden experimentar al trabajar con herramientas de software. Este síntoma está

estrechamente ligado con la escalabilidad, convirtiéndose, en un factor para tener en cuenta a la hora de seleccionar un LPV, pues se pueden volver inmanejables si se trata de escribir programas de gran tamaño. Los problemas de escalabilidad son:

- **Complejidad del diagrama.** Esta característica se refiere a la complejidad de los diagramas generados por el lenguaje. Los posibles valores que se pueden registrar son: alta complejidad (H), complejidad mediana (M) y baja complejidad (L).
- **Capacidad de zoom.** Se refiere a la forma en que se hace la tarea de acercar o alejar la imagen. Los posibles valores que se pueden registrar son: cualquier parte del entorno puede ser ampliada (Y), el zoom no está activado (N) y sólo algunos elementos pueden ser ampliados (P).
- **Nivel de detalle (LOD).** Es una característica similar al zoom que en este caso se rige por el concepto de ojo de pez, es decir, la ampliación de un área específica del lenguaje definida por el usuario. Los posibles valores que se pueden registrar son: LOD está habilitado (Y) y LOD no está habilitado (N).
- **Viscosidad de representación.** Se refiere a la resistencia de una representación a los cambios locales. Por ejemplo, a medida que se añaden objetos a una representación, puede ser necesario reorganizar todo el diagrama para que sea legible. Los posibles valores que se pueden registrar son: no hay cambios locales habilitados (H), algunos cambios pueden estar sujetos a restricciones (M) y no hay límites en los cambios (L).
- **Densidad de representación.** Esta variable evalúa el número de píxeles por nodo y la información útil por centímetro cuadrado que se utilizan en todas las representaciones en pantalla. Los posibles valores que se pueden registrar son: hay un pequeño número de píxeles por nodo y una gran cantidad de información por centímetro cuadrado (H), hay un número moderado de píxeles por nodo y una cantidad moderada de información por centímetro cuadrado (M) y hay muchos píxeles por nodo y una cantidad muy pequeña de información por centímetro cuadrado o se muestra después de una acción adicional del ratón, resultando en gestión rudimentaria de la pantalla (L).
- **Diccionario de iconos.** El lenguaje proporciona una galería de iconos que se pueden utilizar como bloques de construcción de objetos. Los posibles valores que se pueden registrar son: hay un diccionario de iconos (Y), no hay diccionario de iconos (N) y hay iconos para algunas tareas (S).
- **Sintaxis.** El lenguaje tiene una técnica para reducir los errores de sintaxis. Los posibles valores que se pueden registrar son: se habilita la agrupación de bloques (Y) y no se proporciona ninguna técnica (N).

2.2.1.2.3 Características relacionadas con la accesibilidad

Los tamaños y colores de fuente y de fondo, el espaciado de caracteres y líneas, tienen el potencial de reducir los efectos de la diversidad funcional de las personas con dislexia (Rello, 2012). Estos valores se evalúan mediante la siguiente característica:

- **Configuración de la adaptabilidad.** Característica que evalúa si el LPV permite modificar los ajustes visuales en función de determinados parámetros de adaptabilidad. Los posibles valores que se pueden registrar son: todos los ajustes de los elementos anteriores se pueden cambiar (H), sólo se pueden cambiar los ajustes de fuente y fondo (M), sólo se pueden cambiar los ajustes de fondo (L) y no se puede cambiar ninguno de los elementos anteriores (N).

3 Planteamiento del problema

Gracias a las leyes de inclusión que se han establecido alrededor de todo el mundo, hay cada vez más personas con diversidad funcional integradas en distintos ámbitos, como por ejemplo en los estudios universitarios de Informática. En este sentido, es cada vez más probable que puedan existir personas con dislexia (diagnosticada o no) dentro de un grupo de desarrolladores de software.

Tras analizar el impacto de la dislexia en el estudio de la Informática, específicamente en las asignaturas de programación, Powell concluye que los estudiantes con dislexia que asumen dichos currículos se enfrentan a diversos problemas de aprendizaje. Entre otras, al trabajar con una interfaz de desarrollo tradicional, se les dificulta recordar detalles del código, tener un buen manejo en el ordenamiento de éste y presentan dificultades a la hora de codificar el programa o buscar errores y en la correcta definición de variables (específicamente al combinar letras en los nombres de éstas y seguir ciertos patrones de los estándares definidos) (Powell, 2004).

Ruth M. Dixon corrobora estudios realizados en otras universidades del mundo con relación a las diversas dificultades que tienen las personas con dislexia en el uso del ordenador. También afirma que los PCD que trabajan con herramientas de programación visuales o gráficas, son un 20% más exitosos en el desarrollo y seguimiento de programas de cómputo que los programadores sin dislexia (PSD a partir de ahora), mientras que los PSD se desenvuelven en conjunto un 37,5% mejor que los PCD en ambientes de programación basados en texto (Dixon, 2007).

3.1 Pregunta de investigación

Dada la posibilidad de encontrar en un grupo de programadores, que alguno de ellos presente síntomas de dislexia, y teniendo en cuenta la necesidad de favorecer la integración de éstos en equipos de desarrollo, y dado que la investigación sobre la relación entre dislexia y programación es escasa, en esta Tesis se plantea la necesidad de conocer cómo son los programadores con dislexia y cómo varía su desempeño usando lenguajes de programación visuales o textuales. Así, se plantea la siguiente pregunta de investigación:

¿Cuáles son las principales diferencias de desempeño, satisfacción, calidad y comportamentales de los programadores con y sin dislexia, cuando trabajan en un lenguaje de programación visual versus uno textual?

3.2 Hipótesis

Para este estudio se proponen dos hipótesis:

- H1. **Los programadores con dislexia experimentan dificultades al trabajar en un lenguaje de programación textual**, y éstas posiblemente se manifiestan en diferencias de satisfacción, desempeño y comportamentales, en comparación con aquéllos que no padecen los síntomas de este trastorno.
- H2. **Los programadores con dislexia tienen mejor desempeño cuando trabajan en un lenguaje de programación visual** y éste puede facilitar los procesos de desarrollo de software, y a futuro, puede ayudar al proceso de integración de programadores con y sin dislexia, cuando éstos trabajan en equipo.

3.3 Objetivos

A partir de la hipótesis, en esta Tesis se define un objetivo general y cuatro específicos.

3.3.1 Objetivo general

Para dar respuesta a la pregunta de investigación planteada anteriormente se ha definido el siguiente objetivo general de la investigación:

Establecer las diferencias entre programadores con y sin dislexia en el desarrollo de software bajo un lenguaje de programación textual y otro visual, con el fin de facilitar la integración de PCD en equipos de desarrollo.

3.3.2 Objetivos específicos

Los objetivos específicos permiten abordar desde una perspectiva teórica, analítica y experimental el problema expuesto, permitiendo el alcance del objetivo general con miras a validar o no las hipótesis planteadas:

- OE1. Determinar las principales características y dificultades de los programadores con dislexia, cuando desarrollan software, a partir del análisis de su comportamiento físico, emocional y conductual.
- OE2. Construir una taxonomía de la programación visual que permita la selección de un lenguaje apropiado para los programadores con dislexia.
- OE3. Determinar los niveles de desempeño, satisfacción y efectividad de programadores con y sin dislexia en el uso de un lenguaje visual y uno textual.
- OE4. Proponer una estrategia que facilite la integración de programadores con dislexia.

3.4 Metodología

El presente trabajo se realiza con un enfoque analítico experimental, mediante el cual, haciendo uso de cuestionarios, entrevistas y diversos análisis en su mayoría cuantitativos, se pretenden validar las hipótesis de estudio.

Para el logro del objetivo general, se tendrá una combinación de tres metodologías acordes con los diferentes objetivos que se desean alcanzar en la investigación:

- Con relación al OE1, se plantea un estudio de tipo analítico observacional, de corte transversal para el cual, tomando como entrada las características de las personas con dislexia, se diseñará una encuesta como instrumento de recolección de información, indagando acerca de la percepción de un grupo de programadores frente a diversas variables.
- Para el cumplimiento del OE2, también se plantea un estudio analítico observacional, pero de corte descriptivo, para el cual, tomando como punto de partida algunos de los resultados obtenidos en el estudio diseñado para el logro del objetivo 1, específicamente lo relacionado con algunas de las variables de programación, se construirá la caracterización de los lenguajes de programación visual más importantes.
- Para alcanzar el OE3 se realizará un estudio de tipo experimental basado en un análisis comparativo entre programadores con y sin dislexia, que permitirá –a partir de dos experimentos– establecer diferencias de rendimiento, satisfacción y desempeño de los programadores, calcular medidas de calidad en el software desarrollado y, por último, haciendo uso de una técnica de

medición pareada, se establecerán diferencias entre los resultados observados en los dos experimentos.

- Para alcanzar el OE4 se realizará un estudio analítico de los lenguajes seleccionados, permitiendo establecer diferentes relaciones entre ellos. A partir de éstas se formulará una propuesta de integración de programadores con y sin dislexia haciendo uso de ambos lenguajes.

En la Tabla 1 se pueden observar las diferentes actividades, con sus respectivos resultados, asociadas a cada uno de los objetivos específicos. En los siguientes capítulos de esta Tesis se describirá el trabajo realizado y los resultados obtenidos para cada uno de los cuatro objetivos específicos.

Tabla 1. Resumen de actividades y resultados

Objetivos específicos	Actividades	Resultados
OE1 Determinar las principales características y dificultades de los programadores con dislexia, cuando desarrollan software, a partir del análisis de su comportamiento físico, emocional y conductual.	A1.1 Análisis y documentación de problemas y capacidades de las personas con dislexia.	R1 Características de los programadores con dislexia.
	A1.2 Estudio del comportamiento de los programadores con dislexia.	
OE2 Construir una taxonomía de la programación visual que permita la selección de un lenguaje apropiado para los programadores con dislexia.	A2.1 Clasificación de los lenguajes de programación visual más usados en el entorno.	R2.1 Listado de lenguajes de programación visual relevantes y aplicables para el estudio.
	A2.2 Análisis de los lenguajes de programación seleccionados	R2.2 Lenguaje de programación visual seleccionado.
	A2.3 Caracterización de los principales lenguajes de programación visual.	
OE3 Determinar los niveles de desempeño, satisfacción y efectividad de programadores con y sin dislexia en el uso de un lenguaje visual y uno textual.	A3.1 Clasificación de los lenguajes de programación textual más usados.	R3.1 Lenguaje de programación textual seleccionado.
	A3.2 Experimento 1. Experiencias de desempeño, satisfacción y efectividad.	R3.2 Análisis estadístico de desempeño, experiencias y satisfacción en el uso de cada lenguaje.
	A3.3 Experimento 2. Experiencias de desempeño, satisfacción, efectividad y medidas de calidad.	R3.3 Análisis estadístico de medidas de calidad de los programas desarrollados.
	A3.4 Comparación del desempeño de los programadores después de un período de tiempo de uso en ambos lenguajes.	R3.3 Análisis estadístico comparativo.

Objetivos específicos	Actividades	Resultados
OE4 Proponer una estrategia que facilite la integración de programadores con dislexia.	A4.1 Análisis de particularidades y de relación entre los elementos de los lenguajes visual y textual seleccionados	R4.1 Diagramas de componentes y relaciones
	A4.3 Propuesta de integración.	R4.2 Modelo propuesto

4 Características de los programadores con dislexia

El primer objetivo planteado es OE1: “determinar las principales características y dificultades de los programadores con dislexia, cuando desarrollan software, a partir del análisis de su comportamiento físico, emocional y conductual”.

Para cubrirlo, se ha partido de los problemas y capacidades de los adultos con dislexia, descritos en el estado de la cuestión. Se ha realizado un estudio de tipo analítico observacional (Veiga de Cabo, 2008) a través de una técnica de corte transversal, que ayudó a determinar las características de los PCD. Ese estudio, que se describe en detalle a continuación, consistió en una encuesta electrónica aplicada a un grupo de programadores, donde se evaluaron las variables descritas en el siguiente apartado.

4.1 Variables Estudiadas

Se definieron cuatro grupos de variables, que se relacionan a continuación.

4.1.1 Variables específicas

Se incluyen variables que se relacionan con aspectos específicos de la dislexia y otras de información general. En la Tabla 2 se muestran las definidas para este grupo.

Tabla 2. Variables específicas

Nombre de Variable	Definición	Naturaleza	Valor o categoría
Dislexia	Determina el estado de una persona con relación a la presencia o no de dislexia	Cualitativa nominal dicotómica	Sí / No
Diagnóstico	Determina si una persona ha sido diagnosticada profesionalmente como disléxica	Cualitativa nominal dicotómica	Sí / No
Tratamiento	Determina si una persona diagnosticada profesionalmente como disléxica, ha recibido algún tratamiento o ayuda específica al respecto	Cualitativa nominal dicotómica	Sí / No
Desempeño	Determina si una persona diagnosticada, considera si la dislexia afecta negativamente su desempeño a la hora de programar	Cualitativa nominal dicotómica	Sí / No
Colaboración	Determina si una persona manifiesta querer continuar colaborando en futuros estudios de la investigación	Cualitativa nominal dicotómica	Sí / No
Correo	Correo electrónico de las personas que desean seguir colaborando con la investigación	Cualitativa nominal	correo electrónico

4.1.2 Variables sociodemográficas

Como variables demográficas que permiten describir la población de participantes en la encuesta, se definieron tres: edad, nacionalidad y género.

4.1.3 Variables de comportamiento

En este grupo se relacionan los síntomas definidos por Haber para los adultos con dislexia (Haber, 2012), que fueron enunciados y clasificados (en el estado de la cuestión) en tres grupos: Emocionales (Tabla 3), Conductuales (Tabla 4) y Físicos (Tabla 5).

Tabla 3. Variables emocionales

Nombre de Variable	Definición	Naturaleza	Valores
Ansiedad_P	Determina el grado de ansiedad de una persona cuando está programando	Cuantitativa discreta	Entre 1 y 5
Angustia	Determina el grado de angustia de una persona cuando está programando.	Cuantitativa discreta	Entre 1 y 5
Rabia	Determina el grado de rabia que una persona puede experimentar cuando está programando.	Cuantitativa discreta	Entre 1 y 5
Concentración	Determina si una persona experimenta niveles de baja concentración cuando está programando.	Cuantitativa discreta	Entre 1 y 5
Dispersión	Determina el grado de dispersión que una persona puede experimentar cuando está programando.	Cuantitativa discreta	Entre 1 y 5
C_Humor	Determina si una persona presenta con frecuencia cambios de humor cuando está programando.	Cuantitativa discreta	Entre 1 y 5
Depresión	Determina el grado de depresión que una persona puede experimentar cuando está programando.	Cuantitativa discreta	Entre 1 y 5
Tristeza	Determina el grado de tristeza que una persona puede experimentar cuando está programando.	Cuantitativa discreta	Entre 1 y 5
Irritabilidad	Determina el grado de tristeza que una persona puede experimentar cuando está programando.	Cuantitativa discreta	Entre 1 y 5
Indefenso	Determina el grado de sentimiento de indefenso que una persona puede experimentar ante ciertas circunstancias de la vida.	Cuantitativa discreta	Entre 1 y 5
B_Autoestima	Determina el grado de baja autoestima que una persona manifiesta poseer.	Cuantitativa discreta	Entre 1 y 5
Desconfianza	Determina el grado de desconfianza que una persona manifiesta tener.	Cuantitativa discreta	Entre 1 y 5

Tabla 4. Variables conductuales

Nombre de Variable	Definición	Naturaleza	Valores
Excusa_E	Determina la frecuencia de excusas para no asistir a clases en el colegio.	Cuantitativa discreta	Entre 1 y 5
Excusa_T	Determina la frecuencia de excusas para no asistir al trabajo o universidad.	Cuantitativa discreta	Entre 1 y 5
Atención	Determina la frecuencia con que una persona utiliza conductas para llamar la atención.	Cuantitativa discreta	Entre 1 y 5

Nombre de Variable	Definición	Naturaleza	Valores
Hipoactividad	Determina si una persona experimenta con frecuencia síntomas de hipoactividad o ganas de no hacer nada.	Cuantitativa discreta	Entre 1 y 5
Hipoactividad_P	Determina si la hipoactividad afecta a la persona a la hora de programar.	Cuantitativa discreta	Entre 1 y 5
Agresividad	Determina si una persona experimenta con frecuencia síntomas de agresividad.	Cuantitativa discreta	Entre 1 y 5
Agresividad_P	Determina si la agresividad afecta a la persona a la hora de programar.	Cuantitativa discreta	Entre 1 y 5
Retraído	Determina si una persona se considera retraído.	Cuantitativa discreta	Entre 1 y 5
Retraído_P	Determina si el ser agresivo afecta a la persona a la hora de programar.	Cuantitativa discreta	Entre 1 y 5
Compulsivo	Determina si una persona se considera compulsiva.	Cuantitativa discreta	Entre 1 y 5
Compulsivo_P	Determina si el ser compulsivo afecta a la persona a la hora de programar.	Cuantitativa discreta	Entre 1 y 5
Aislamiento	Determina si una persona le gusta estar aislado de los demás.	Cuantitativa discreta	Entre 1 y 5
Aislamiento_P	Determina si el gustarle estar aislado afecta a la persona a la hora de programar.	Cuantitativa discreta	Entre 1 y 5
Hiperactividad	Determina si una persona experimenta con frecuencia síntomas de hiperactividad.	Cuantitativa discreta	Entre 1 y 5
Hiperactividad_P	Determina si la hiperactividad afecta a la persona a la hora de programar.	Cuantitativa discreta	Entre 1 y 5
Ansiedad_G	Determina el grado general de ansiedad de una persona.	Cuantitativa discreta	Entre 1 y 5
Ansiedad_H	Determina si la ansiedad causa hiperactividad en la persona.	Cuantitativa discreta	Entre 1 y 5

Tabla 5. Variables físicas

Nombre de Variable	Definición	Naturaleza	Posibles valores
Insomnio	Determina si una persona con frecuencia padece de insomnio.	Cuantitativa discreta	Entre 1 y 5
Madrugar	Determina si una persona tiene dificultad para levantarse en las mañanas.	Cuantitativa discreta	Entre 1 y 5
Migrañas	Determina si una persona sufre de continuos dolores de cabeza o migrañas.	Cuantitativa discreta	Entre 1 y 5
Fatiga	Determina si una persona con frecuencia se siente fatigada.	Cuantitativa discreta	Entre 1 y 5
Visión_D	Determina si una persona padece de visión desordenada (identifica figuras y/o letras al revés.	Cuantitativa discreta	Entre 1 y 5

Para la captura de información relacionada con estas variables se usó una escala de Likert puntuada de 1 a 5. Cada variable está asociada a una frase con la que el participante en la encuesta muestra su grado de acuerdo: 1 significa totalmente en desacuerdo, 3 es indiferente y 5 totalmente de acuerdo.

4.1.4 Variables de programación

Se incluyeron diversas variables asociadas con las preferencias de los programadores respecto a diferentes lenguajes de programación, tanto visuales como textuales, al igual que otras relacionadas con comportamientos específicos a la hora de programar. En la Tabla 6 se exponen las definidas para este grupo.

Para la captura de información relacionada con estas variables se usó una escala semántica diferencial de 1 a 7, donde 1 es lenguaje visual y 7 es lenguaje textual.

Tabla 6. Variables de programación

Nombre de Variable	Definición	Naturaleza	Valor o categoría
Preferencia	Preferencia de una persona con relación al uso de un lenguaje visual o uno textual.	Cuantitativa Discreta	Entre 1 y 7
Gusto_P	Determina si una persona trabaja más a gusto en un lenguaje visual o uno textual.	Cuantitativa Discreta	Entre 1 y 7
Rapidez	Determina si una persona trabaja más rápido en un lenguaje visual o uno textual.	Cuantitativa Discreta	Entre 1 y 7
Errores	Determina si una persona comete más errores en un lenguaje visual o uno textual.	Cuantitativa Discreta	Entre 1 y 7

Nombre de Variable	Definición	Naturaleza	Valor o categoría
Seguridad	Determina si una persona se siente más segura al trabajar en un lenguaje visual o uno textual.	Cuantitativa Discreta	Entre 1 y 7
Programación_V	Determina si una persona ha trabajado alguna vez en un lenguaje visual	Cualitativa nominal dicotómica	Sí / No
Lenguaje_V_U	Determina que lenguajes de programación visual ha usado una persona	Cualitativa nominal	Respuesta abierta (nombre LPV)
Lenguaje_T_U	Determina que lenguajes de programación textual ha usado una persona	Cualitativa nominal	Respuesta abierta (nombre LPT)
Problemas_P	Determina los principales problemas experimentados por una persona al programar	Cualitativa nominal	Respuesta abierta

Es importante resaltar con relación a las variables Lenguaje_V_U y Lenguaje_T_U, que éstas no se consideran politómicas, porque en la encuesta se suministrará una lista de lenguajes a utilizar, además de dar la opción de que el participante escriba el nombre de otro lenguaje no especificado en la lista.

4.2 Resultado de la encuesta

A continuación, se exponen los diversos hallazgos del estudio.

4.2.1 Muestra

En un principio se realizó contacto con María Sanz, presidenta de la Fundación “Madrid con la Dislexia”, Yolanda Granados, vicepresidenta de la Fundación “Aprender” y Luz Rello, fundadora de “Change Dyslexia”, con el fin de contactar con personas con dislexia que fueran programadores de computadores. Estas fundaciones mostraron interés por la propuesta del estudio, pero la búsqueda resultó ser infructuosa, por las características específicas de la población buscada. Todas las fundaciones contactadas manifestaron que su población estaba enfocada a niños y reconocieron que la dislexia en adultos se encontraba en un segundo plano. Sin embargo, Luz Rello sugirió hacer uso del software “Dytective” y lo puso a disposición del proyecto, para intentar identificar pacientes con riesgo de padecer dislexia.

Dado lo anterior y ante la dificultad de encontrar una población de PCD, se optó por una solución alternativa, consistente en contactar con estudiantes universitarios de carreras relacionadas con la Informática. Por un lado, se invitó a participar en el estudio a 300 estudiantes matriculados en la asignatura obligatoria “Interacción Persona-Ordenador”, del título de Grado en Ingeniería Informática de la Universidad Politécnica de Madrid (UPM). Por otro lado, se realizó la misma invitación a 343 estudiantes de las asignaturas obligatorias “Taller de Programación”, “Lenguajes de Programación” y “Taller de Fundamentos de Programación” del programa Ingeniería Informática del Politécnico Colombiano Jaime Isaza Cadavid (PCJIC). Estos números de participantes invitados incluyen también a los profesores de las asignaturas mencionadas.

En la encuesta de UPM participaron 172 del total de 300 invitados (el 57%). En la encuesta del PCJIC participaron 143 del total de 343 invitados (el 42%). En resumen, participaron un total de 315 programadores (el 49% del total de invitados).

En conclusión, se realizó un doble muestreo a conveniencia y de bola de nieve discrecional, que surge a partir de la respuesta dada a la invitación a participar en las dos encuestas.

4.2.2 Análisis de variables sociodemográficas

Como se puede observar en la Figura 5, con relación al género se halló que la muestra contenía un 19% de mujeres y un 81% de varones. Esta distribución es razonable y habitual para el contexto del área informática, donde algunos estudios han establecido que la población femenina de ésta se encuentra alrededor 15% (Glass, 2012).

Figura 5. Frecuencia según género

Con relación a las variables sociodemográficas, se halló que, del total de participantes, el 49,2% eran de nacionalidad española ($n = 155$), el 45,4% colombiana ($n = 143$) y el resto pertenecían a otras nacionalidades ($n = 17$). Se observa una distribución más o menos similar entre España y Colombia y ésta es razonable, ya que el estudio se realizó con estudiantes y profesionales del área informática residentes en ambos países. La distribución detallada por cada país se refleja en la Tabla 7.

Tabla 7. Frecuencia nacionalidad por Género

País	Varones		Mujeres	
	Frecuencia	Porcentaje	Frecuencia	Porcentaje
Argentina	1	0,4%	0	0%
Bulgaria	0	0%	1	1,7
Colombia	116	45,5%	27	45%
Cuba	1	0,4%	0	0%
Ecuador	1	0,4%	3	5%
España	127	49,8%	28	46,7
México	1	0,4%	0	0%
Perú	3	1,2%	1	1,7%
Rumania	1	0,4%	0	0%
Ucrania	3	1,2%	0	0%
Venezuela	1	0,4%	0	0%
Total	255	100%	60	100%

Se obtuvo una edad media de 24,6 años con una mínima de 16 años y una máxima de 59 años (Tabla 8). Estos valores son razonables, ya que la mayoría de la población

participante corresponde a estudiantes jóvenes y esto se puede corroborar a través de la mediana de la edad que es de 22 años.

Tabla 8. Frecuencia de Edades

Edades	Varones		Mujeres	
	Frecuencia	Porcentaje	Frecuencia	Porcentaje
16	1	0,4%	0	0%
17	2	0,8%	0	0%
19	6	2,4%	0	0%
20	59	23,1%	18	30%
21	38	14,9%	7	11,7%
22	31	12,2%	8	13,3%
23	22	8,6%	6	10%
24	2	0,8%	6	10%
25	15	5,9%	0	0%
26	6	2,4%	3	5%
27	3	1,2%	1	1,7%
28	9	3,5%	3	5%
29	10	3,9%	1	1,7%
30	7	2,7%	2	3,3%
31	11	4,3%	1	1,7%
32	9	3,5%	1	1,7%
33	4	1,6%	1	1,7%
34	4	1,6%	0	0%
35	2	0,8%	0	0%
36	2	0,8%	0	0%
39	1	0,4%	0	0%
40	1	0,4%	0	0%
42	2	0,8%	0	0%
43	1	0,4%	0	0%
44	1	0,4%	0	0%
49	1	0,4%	0	0%
50	1	0,4%	0	0%
51	1	0,4%	0	0%
54	1	0,4%	0	0%
55	1	0,4%	0	0%
59	1	0,4%	0	0%
Total	255	100%	60	100%

4.2.3 Análisis de variables específicas

Tras el análisis de la variable Dislexia, se halló que, del total de encuestados, el 12,4% creen tener dislexia (n = 39). Este hallazgo se distribuye como se observa en la Tabla 9.

Tabla 9. Prevalencia de dislexia

España		Colombia		Otros países		Total	
(n=155)	%	(n=143)	%	(n=17)	%	(n=315)	%
10	6,5%	29	20,3%	0	0%	39	12,4%

Los porcentajes anteriores están por encima de las cifras de prevalencia mundiales, siendo coherente con lo planteado por la OMS en el sentido de que la dislexia cada día va en aumento, aunque vale la pena resaltar que estos hallazgos obedecen a mediciones realizadas en personas adultas; contrario a todos los estudios encontrados, los cuales han sido realizados en población infantil.

Lo anterior ratifica la idea de que la dislexia en lo adultos es una diversidad funcional olvidada y poco estudiada en los centros de educación superior.

De los 39 programadores que creen tener dislexia, sólo 3 han sido diagnosticados profesionalmente, es decir, el 7,7% y de éstos, 2 han recibido tratamiento al respecto.

El 0,95% de los adultos que manifestaron creer tener dislexia han sido diagnosticados por un profesional, lo cual permite inferir el desconocimiento sobre este trastorno del aprendizaje

De otro lado, de los 39 programadores que creen tener dislexia, el 46,2% (n = 18) consideran que la dislexia les afecta negativamente a la hora de programar.

La Tabla 10 muestra la frecuencia de presencia de dislexia, la Tabla 11 el diagnóstico profesional de dislexia, la Tabla 12 la frecuencia de tratamiento de dislexia y la Tabla 13 la frecuencia de afección de la dislexia en los programadores.

Tabla 10. Frecuencia presencia de dislexia

Dislexia	Varones		Mujeres	
	Frecuencia	Porcentaje	Frecuencia	Porcentaje
No	224	87,8%	52	86,7%
Sí	31	12,2%	8	13,3%
Total	255	100%	60	100%

Tabla 11. Diagnóstico de dislexia

Diagnóstico Dislexia	Varones		Mujeres	
	Frecuencia	Porcentaje	Frecuencia	Porcentaje
Sin Dislexia	224	87,8%	52	86,7%
No	29	11,4%	7	11,7%
Sí	2	0,8%	1	1,7%
Total	255	100%	60	100%

Tabla 12. Frecuencia Tratamiento de dislexia

Tratamiento Dislexia	Varones		Mujeres	
	Frecuencia	Porcentaje	Frecuencia	Porcentaje
Sin Dislexia	224	87,8%	52	86,7%
No	30	11,8%	7	11,7%
Sí	1	0,4%	1	1,7%
Total	255	100%	60	100%

Tabla 13. Frecuencia afección de la dislexia en el desempeño

Afección de la Dislexia	Varones		Mujeres	
	Frecuencia	Porcentaje	Frecuencia	Porcentaje
Sin Dislexia	224	87,8%	52	86,7%
No	15	5,9%	6	10,0%
Sí	16	6,3%	2	3,3%
Total	255	100%	60	100%

Dado que el análisis para la variable ‘nacionalidad’ arrojó una población estadísticamente significativa para España (n=155) y Colombia (n=143), y debido a que los 39 programadores con dislexia son de nacionalidad española y colombiana, el resto de los análisis se realizan centrados en estos 2 países (n=298).

Con relación a la muestra colombiana, se establece que el 20,3% de los programadores encuestados (n = 29) han sido diagnosticados como disléxicos por un

profesional o creen tener dislexia; en éstos se encontró una edad media de 25 años, mediana de 27, edad mínima de 20 y una máxima de 51 años.

De total de los programadores colombianos con dislexia, el 18,9% eran mujeres y el 81,1% varones, conservándose casi la misma proporción hallada para el total de la encuesta. Se halló que solo una persona (el 3,4% de los 29 PCD) ha sido diagnosticada profesionalmente como disléxico y que ninguno ha recibido algún tratamiento o ayuda específica al respecto, lo cual permite inferir el desconocimiento sobre esta alteración y la falta de un manejo adecuado.

El 48,3% de las personas que creen tener dislexia o que han sido diagnosticadas por un profesional (n = 14) consideran que la dislexia afecta negativamente su desempeño como programadores.

Para la muestra española se halló que el 6,5% de los encuestados (n = 10) han sido diagnosticados como disléxicos por un profesional o creen tener dislexia; la edad media es 22 años, la mediana 21, con una edad mínima de 20 y una máxima de 44 años.

Además, del total de los programadores españoles con dislexia, el 20% eran mujeres y el 80% varones, conservándose casi la misma proporción de la encuesta total. De otro lado, respecto al tratamiento recibido, se tiene que 2 personas (el 20% de los 10 PCD) ha sido diagnosticados profesionalmente como disléxicos, pero ninguno ha recibido algún tratamiento o ayuda específica al respecto. El 40% de las personas que creen tener dislexia o que ha sido diagnosticadas por un profesional (n = 4) consideran que la dislexia afecta negativamente su desempeño como programadores.

En la Figura 6 se puede observar la distribución de la muestra según el género y la frecuencia de programadores con dislexia clasificados por los dos países más relevantes.

Figura 6. Distribución por género PCD

4.2.4 Análisis de variables de comportamiento

Para las variables de comportamiento, se analizaron las 34 definidas en las tablas 2 a la 6. Con el fin de establecer el nivel de desacuerdo y de acuerdo de los programadores, se han agrupado los valores 1 2 (muy en desacuerdo o desacuerdo) y 4 5 (de acuerdo o muy de acuerdo) respectivamente, acorde con los datos obtenidos según la escala definida.

En el análisis, un dato se considera relevante cuando el porcentaje de uno de los valores agrupados está por encima del 50%. Las estadísticas relevantes relacionadas con el subgrupo de las variables emocionales se presentan en la Tabla 14.

Tabla 14. Variables emocionales: porcentajes más relevantes.

Variable	España				Colombia			
	PCD (n = 10)		PSD (n = 145)		PCD (n = 29)		PSD (n = 114)	
	1-2	4-5	1-2	4-5	1-2	4-5	1-2	4-5
Ansiedad_P	20,0%	70,0%	-	-	10,3%	55,2%	-	-
Angustia	-	-	-	-	-	-	52,6%	22,8%
Rabia	-	-	-	-	-	-	57,0%	25,4%
Concentración	80,0%	10,0%	68,3%	17,2%	-	-	63,2%	19,3%
Dispersión	40,0%	60,0%	29,7%	50,3%	17,2%	65,5%	-	-
C_Humor	50,0%	40,0%	61,3%	22,1%	55,2%	31,0%	58,8%	21,9%
Depresión	90,0%	0,0%	78,7%	7,6%	69,0%	10,3%	74,6%	10,5%
Tristeza	50,0%	10,0%	71,7%	12,4%	62,1%	13,8%	71,9%	10,5%
Irritabilidad	-	-	61,4%	19,3%	51,7%	20,7%	64,9%	14,0%
Indefenso	-	-	55,2%	21,4%	51,7%	17,2%	57,0%	17,5%
B_Autoestima	60,0%	20,0%	60,7%	17,3%	62,1%	13,8%	80,7%	12,3%

Los porcentajes más relevantes relacionados específicamente con el subgrupo de las variables conductuales se recogen en la Tabla 15.

Tabla 15. Variables conductuales: porcentajes más relevantes.

Variable	España				Colombia			
	PCD n = 10		PSD n = 145		PCD n = 29		PSD n = 114	
	1-2	4-5	1-2	4-5	1-2	4-5	1-2	4-5
Excusa_E	80,0%	20,0%	86,2%	5,5%	75,9%	13,8%	86,8%	6,1%
Excusa_T	80,0%	10,0%	83,5%	5,5%	86,2%	3,4%	93,0%	1,8%
Atención	70,0%	30,0%	85,5%	4,2%	89,7%	0,0%	86,8%	5,3%
Agresividad	80,0%	20,0%	83,5%	4,2%	89,7%	3,4%	77,2%	7,9%
Agresividad_P	-	-	-	-	100,0%	0,0%	77,8%	11,1%
Retraído	90,0%	10,0%	67,6%	17,2%	58,6%	24,1%	71,9%	14,0%
Retraído_P	-	-	-	-	-	-	62,5%	12,5%
Compulsivo	50,0%	40,0%	55,2%	19,3%	55,2%	10,3%	75,4%	11,4%
Cumpulsivo_P	-	-	-	-	66,7%	0,0%	-	-
Aislamiento	60,0%	10,0%	62,1%	20,7%	-	-	54,4%	25,4%
Aislamiento_P	-	-	-	-	61,5%	38,5%	65,5%	27,6%
Hiperactividad	40,0%	50,0%	64,8%	14,5%	58,6%	20,7%	64,0%	19,3%
Hiperactividad_P	-	-	-	-	66,7%	33,3%	50,0%	40,9%
Ansiedad_G	40,0%	60,0%	57,2%	23,4%	-	-	-	-

Los valores relacionados con el subgrupo de las variables físicas se presentan en la Tabla 16.

Tabla 16. Variables físicas: porcentajes más relevantes.

Variable	España				Colombia			
	PCD n = 10		PSD n = 145		PCD n = 29		PSD n = 114	
	1-2	4-5	1-2	4-5	1-2	4-5	1-2	4-5
Insomnio	80,0%	10,0%	68,3%	19,3%	-	-	58,8%	25,4%
Madrugar	40,0%	50,0%	29,7%	55,2%	41,4%	51,7%	-	-
Migrañas	70,0%	10,0%	78,7%	12,4%	69,0%	27,6%	70,2%	17,5%
Visión_D	70,0%	30,0%	95,9%	1,4%	51,7%	34,5%	87,7%	2,6%
Fatiga	-	-	66,2%	15,1%	-	-	50,9%	26,3%

Al conjunto de las 2 muestras (española y colombiana) se les aplicó una prueba de Chi cuadrado a la variable Dislexia (¿Cree tener dislexia? sí o no), asociado con todas las variables de comportamiento, encontrándose una asociación estadísticamente significativa en las variables que se observa en la Tabla 17.

Tabla 17. Asociación estadísticamente significativa

Variables	Chi Cuadrado
Ansiedad_P	(p = 0,028)
Depresión	(p = 0,023)
Excusas_E	(p = 0,027)
Hipoactividad_P	(p = 0,003)
Fatiga_	(p = 0,005)
Visión_Desordenada	(p = 0,000)

Estadísticamente, toda variable que registre una p menor o igual a 0,005, establece una relación positiva con la variable Dislexia, reflejando las variables que sí influyen en la presencia de dislexia, en la población de programadores estudiados.

Opuesto a lo planteado por Shaywitz, quien manifiesta que los adultos con dislexia se caracterizan por tener fuertes síntomas de baja autoestima, desconfianza e indefensión ante ciertas circunstancias de la vida (Shaywitz, 1996), el resultado de estas variables tanto en la muestra española como en la colombiana no refleja datos significativos para que se puedan establecer como una característica de los PCD.

En la Figura 7 se pueden observar las variables que sí son relevantes en la caracterización de los PCD españoles, en función del grado de aceptación ante la afirmación expuesta en la encuesta. En ésta se ve que los PCD consideran que son dispersos y experimentan ansiedad cuando están programando, además de considerarse ansiosos en términos generales; de hecho, para las dos variables de ansiedad lo afirman con más seguridad que los PSD.

Figura 7. Variables comportamentales relevantes en la caracterización de PCD. Grado desacuerdo

En la Figura 8 se pueden observar las variables que sí son relevantes en la caracterización de los PCD colombianos, en función del grado de aceptación de la afirmación expuesta en la encuesta. En ésta se ve que los PCD consideran que son dispersos y experimentan ansiedad cuando están programando; de hecho, para las dos variables lo afirman con más seguridad que los PSD.

Figura 8. Variables comportamentales relevantes en la caracterización de PCD. Grado de acuerdo

4.2.5 Análisis de variables de programación

Con relación a los aspectos de programación, se indagó acerca del uso y preferencia de lenguajes tanto visuales como textuales, al igual que ciertos comportamientos específicos que se pueden presentar a la hora de programar. Los principales hallazgos se describen a continuación.

Para la variable Programación_V se encontró que, del total de encuestados en España, el 20,7% de los programadores (n = 32) han programado alguna vez en un lenguaje

de programación visual. De estas 32 personas el 9,4% (n = 3) corresponden a PCD y el 90,6% (n = 29) corresponden a PSD.

De otro lado, para la muestra colombiana, se encontró que el 29,4% de los programadores encuestados (n = 42) han programado alguna vez en un lenguaje de programación visual y que de éstos, el 19,1% (n = 8) son programadores con dislexia y el 81,0% (n = 34) son programadores sin dislexia.

Con relación a la variable Lenguaje_V_U se halló que, de los 315 programadores encuestados, 74 han usado alguna vez un lenguaje de programación visual. La frecuencia de uso de los lenguajes que han sido usados más de una vez se observa en la Tabla 18.

Tabla 18. Frecuencia de uso de los LPV más relevantes

Lenguaje	Programadores	Porcentaje
Scratch	40	54,1%
Alice	20	27,0%
AppInventor	18	24,3%
Kodu	6	8,1%
Google Blockly	5	6,8%
Netlogo	5	6,8%
Vis Sim	5	6,8%
LabVIEW	2	2,7%

Para la variable Lenguaje_T_U se identificaron 51 lenguajes de programación textual como los ‘comúnmente más usados’ por los 315 programadores encuestados. En la Tabla 19 se observa la frecuencia de uso de los que han sido usados más de una vez.

Tabla 19. Frecuencia de uso de los LPT reportados

Lenguaje	Programadores	Porcentaje
Java	293	93,
C	229	72,7
Java Script	161	51,1
C++	149	47,3
Phyton	134	42,5
PHP	114	36,2
C#	102	32,4
Visual. Net	80	25,4
Matlab	16	5,1
Basch	10	3,2
SQL	7	2,2
RUBY	7	2,2
ENSAMBLADOR	7	2,2
VHDL	5	1,6
VISUAL BASIC	3	0,9
Pascal	3	0,9
HTML	3	0,9
HASKELL	3	0,9
XML	2	0,6
SCALA	2	0,6

Lenguaje	Programadores	Porcentaje
PLSQL	2	0,6
My SQL	2	0,6
MAPLE	2	0,6
HTML5	2	0,6
Fox Pro	2	0,6
CSS	2	0,6
Cobol	2	0,6
ABAP	2	0,6

En la Figura 9 se pueden observar los porcentajes de uso, de los 9 lenguajes más utilizados por los encuestados.

Figura 9. Frecuencia de uso de Lenguajes de Programación Textual

De otro lado, con las dos muestras de programadores que han trabajado en un lenguaje de programación visual (españoles = 32 y colombianos = 42) se evaluaron separadamente diversas variables de la programación visual en contraste con la textual.

Con el fin de establecer el nivel de inclinación por los LPV o los LPT, se han agrupado los valores 1 2 3 (mayor o menor preferencia por visual) y 5 6 7 (mayor o menor preferencia por textual), a partir de los datos obtenidos según la escala definida.

En el análisis, un dato se considera relevante cuando el porcentaje de uno de los valores agrupados está por encima del 50%.

Los porcentajes más relevantes se evidencian en la Tabla 20.

Tabla 20. Porcentajes más relevantes variables de programación

Variable	España		Colombia					
	PCD		PSD		PCD		PSD	
	n = 10		n = 145		n = 29		n = 114	
	Visual 1-2-3	Textual 5-6-7	Visual 1-2-3	Textual 5-6-7	Visual 1-2-3	Textual 5-6-7	Visual 1-2-3	Textual 5-6-7
Preferencia	66,7%	33,3%	24,1%	58,5%	12,5%	75,0%	38,2%	59,9%
Gusto_P	66,7%	33,3%	34,4%	51,7%	12,5%	75,0%	44,1%	44,1%
Rapidez	66,7%	33,3%	48,2%	37,9%	25,0%	50,0%	41,2%	47,1%
Errores	33,3%	66,7%	48,2%	37,9%	37,5%	37,5%	41,2%	14,7%
Seguridad	33,3%	33,3%	31,0%	44,8%	0,0%	62,5%	29,4%	41,2%

En los resultados obtenidos, con relación a las variables de programación, específicamente para la muestra colombiana, deja ver que la preferencia, rapidez en el uso y sentirse más gusto por los lenguajes de programación textual, contradicen lo manifestado por Dixon, quien demostró que los PCD que trabajan herramientas de programación con ayudas visuales o gráficas, son un 20% más hábiles en el desarrollo y seguimiento de programas de cómputo que los PSD (Dixon, 2007) sin embargo, se observa que los PCD consideran que cometen menos errores que los PSD al trabajar en un lenguaje de programación visual.

La Figura 10 ayuda a entender que, en términos generales, los LPV son de poco uso por los encuestados, pero de éstos, los que dicen tener dislexia, afirman preferir un lenguaje de programación visual frente a uno textual, al igual que manifiestan sentirse más a gusto, realizan más rápido las tareas de programación, cometen menos errores y se sienten más seguros cuando trabajan en un LPV.

Figura 10. Análisis de frecuencia, variables de programación

En contraste con lo anterior, los datos obtenidos en la muestra española con relación a las variables de: preferencia, sentirse más a gusto y rapidez al programar, permiten suponer que los PCD españoles son un 33,4% más hábiles en el desarrollo y seguimiento de programas de cómputo bajo un entorno visual, resultado que corrobora la hipótesis planteada por Dixon.

Con relación a la variable Problemas_P, la Tabla 21 da a conocer las principales dificultades que manifestaron los programadores a la hora de desarrollar software, su frecuencia y el porcentaje con relación al total de los PCD reportados en cada país.

Tabla 21. Principales dificultades de los PCD a la hora de programar

Dificultad	España PCD = 10		Colombia PCD = 29	
	Frecuencia	%	Frecuencia	%
Falta de dedicación	0	0%	1	3,4%
Ganas de hacer nada	0	0%	1	3,4%
Motivación	0	0%	1	3,4%
Pereza	0	0%	1	3,4%
Cansancio	0	0%	2	6,9%
Entender el código escrito por otros.	1	10,0%	0	0%
Seguir secuencia de un código "desordenado"	1	10,0%	0	0%
Transmitir ideas	1	10,0%	0	0%
Ubicación en una línea específica del texto	1	10,0%	0	0%
Ansiedad P	1	10,0%	1	3,4%
Frustración	1	10,0%	1	3,4%
Lógica	1	10,0%	1	3,4%
Manejo del Tiempo	1	10,0%	1	3,4%
Concentración	1	10,0%	2	6,9%
Diseño de solución	1	10,0%	2	6,9%
Manejo de sintaxis específicas	1	10,0%	2	6,9%
Confusión de comandos	1	10,0%	3	10,3%
Distracciones	1	10,0%	3	10,3%
Perdida de la línea de pensamiento	1	10,0%	3	10,3%
Recordar el código recién escrito	1	10,0%	3	10,3%
Identificar el problema	2	20,0%	1	3,4%
Transposición de letras, al leer o escribir	2	20,0%	3	10,3%

La Figura 11 resume las dificultades más relevantes de los PCD distribuidos en los dos países estudiados.

Figura 11. Distribución demográfica. Principales dificultades de los PCD

En dicha figura se puede observar que, en los programadores colombianos, la confusión de comandos, la transposición de letras, la pérdida de la línea del pensamiento y el recuerdo del código recién escrito, son las dificultades que se presentan con mayor frecuencia ($f = 3$). Con relación a los programadores españoles, las dificultades con mayor frecuencia son la identificación del problema y la transposición de letras al leer o escribir, corroborando las hipótesis de Dixon (2007) y Powell (2004) con relación a las particularidades de los PCD.

4.3 Resultado: caracterización de PCD

De acuerdo con la metodología planteada y en relación con el resultado R1 “Características de los programadores con dislexia”, tras realizar un análisis de las variables que resultaron ser relevantes en los PCD y de las dificultades que éstos manifiestan tener a la hora de programar, se pueden establecer las siguientes:

- **Variables emocionales.** Las principales características identificadas como relevantes son:
 - Son ansiosos al programar.
 - Se dispersan con facilidad al programar.
 - En términos generales poseen buena concentración.
 - No sufren de episodios frecuentes de depresión.
- **Variables conductuales.** Las principales características identificadas como relevantes son:
 - En términos generales se consideran personas ansiosas.
 - Se distraen con facilidad.
 - No les gusta llamar la atención.
 - No son retraídos.
 - Tienen buen grado de responsabilidad.
 - Con personas relativamente calmadas, no son agresivos.
- **Variables físicas.** Las principales características identificadas como relevantes son:
 - Presentan problemas de transposición de letras al leer o escribir.
 - Experimentan deficiencia en la memoria a corto plazo.
 - Tienden a no sufrir de insomnio.
- **Variables de programación.** Las principales características identificadas son:
 - Presentan dificultad para la identificación del problema.
 - Confusión de comandos y manejo de sintaxis específicas.

En resumen, se establece que no todas las características emocionales, conductuales y físicas propuestas por (Shaywitz, 1996), son relevantes en la definición del perfil del PCD y se hace énfasis en que las siguientes son contrarias a lo planteado:

- Baja concentración
- Presencia de episodios de depresión
- Llamar la atención
- Ser retraídos
- Baja responsabilidad
- Agresividad
- Insomnio

En cambio, las siguientes características sí son acordes con las de los adultos con dislexia, identificadas en la revisión de literatura:

- Ansiosos al programar

- Dispersión con facilidad al programar
- Ansiedad general
- Distracción general
- Transposición de letras
- Deficiencia en la memoria a corto plazo

Las características identificadas con relación a las variables de programación son acordes con la literatura encontrada y permiten corroborar que los problemas de sintaxis se convierten en la principal barrera de los PCD a la hora de desarrollar software.

5 Lenguaje visual para programadores con dislexia

Una vez analizado el perfil de los programadores con dislexia, el siguiente objetivo planteado es OE2: “construir una taxonomía de la programación visual que permita la selección de un lenguaje apropiado para los programadores con dislexia”.

De acuerdo con la metodología de trabajo, se proponen realizar tres actividades que fueron identificadas como: “A2.1 Clasificación de los lenguajes de programación visual más usados en el entorno”, “A2.2 Análisis de los lenguajes de programación seleccionados” y “A2.3 Caracterización de los principales lenguajes de programación visual”, las cuales se desarrollan a continuación.

5.1 Clasificación de lenguajes de programación visual

Con el fin de determinar los lenguajes que se incluirán en un análisis posterior, se realizó una primera clasificación de 31 LPV o herramientas visuales, seleccionadas de los lenguajes reportados por los encuestados en el primer estudio y otros identificados como relevantes en la revisión de literatura. Para esta clasificación se han tenido en cuenta las siguientes características:

- **Propósito.** Esta variable determina si tiene un propósito específico o si es de uso general.
- **Área.** Cuando el lenguaje es de propósito específico, determina el campo de conocimiento para el que ha sido creado el lenguaje de programación o herramienta.
- **Relevancia.** Para la primera clasificación se seleccionó esta variable, ya que puede haber algunos lenguajes de propósito específicos que, por su relevancia justifiquen su inclusión en el estudio final.
- **Aplicabilidad.** Determina si el lenguaje es actualmente aplicable para ser incluido en los análisis posteriores.

A continuación, se hace una breve descripción de cada uno de los lenguajes, ordenados alfabéticamente.

5.1.1 Alice

Alice es un lenguaje de programación orientado a objetos. Se basa en el paradigma de “arrastrar y soltar” para crear mundos virtuales mediante el uso de modelos en 3D (Carnegie Mellon University, 2017).

Es considerado un lenguaje de propósito específico ya que sus funciones esenciales están enfocadas a la programación 3D, la cual permite crear animaciones para contar historias a partir de una serie de objetos tridimensionales (personas, muebles, vehículos) que pueblan un mundo virtual, creando con ellos un programa para animarlos. Alice está clasificado en el grupo de los lenguajes desarrollados para enseñar a programar y en la actualidad se considera relevante y aplicable e incluso es usado en varios centros de formación como herramienta para iniciar los niños al mundo de la programación.

En resumen, Alice tiene las siguientes características:

- Propósito: específico
- Área: programación 3D
- Relevancia: sí

- Aplicabilidad: sí.

5.1.2 AppInventor

AppInventor es un lenguaje de propósito general diseñado para la creación de aplicaciones de cualquier tipo para el sistema operativo Android. Permite fácilmente que toda persona, incluyendo las no familiarizadas con la programación, puedan crear aplicaciones de software fácilmente. Utiliza una interfaz gráfica, muy similar a la de Scratch que permite a los usuarios arrastrar y soltar objetos visuales (MIT Media Lab, 2015).

AppInventor ha sido clasificado en el grupo de los lenguajes desarrollados para programar y en la actualidad se considera relevante y aplicable e incluso a la fecha se han registrado más de diez millones de aplicaciones creadas con éste (MIT Media Lab, 2018).

En resumen, AppInventor tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: sí
- Aplicabilidad: sí.

5.1.3 Ark

Ark es un lenguaje de programación de propósito específico ya que sus funciones esenciales están enfocadas a la simulación. Proporciona a los usuarios un entorno animado 2D para crear simulaciones interactivas (Boshernitsan, 2004).

No se ha considerado como relevante por la especificidad de su aplicación y no es aplicable ya que en la actualidad no es de mucho uso y su furor fue en la década del 2000.

En resumen, Ark tiene las siguientes características:

- Propósito: específico
- Área: simulaciones interactivas
- Relevancia: no
- Aplicabilidad: no

5.1.4 Blockly

Blockly es una biblioteca de JavaScript de propósito específico para construir editores de programación visual. Está diseñado para ser flexible y admite un amplio conjunto de características para diferentes aplicaciones, pero Blockly no es en sí mismo un lenguaje (Google Inc, 2016).

Blockly facilita agregar programación visual basada en bloques a una aplicación. Se considera relevante por su gran uso en el entorno y no se ha elegido como aplicable para los próximos análisis, por no considerarse un lenguaje de programación como tal, además de que se ejecuta como una página web y no en un IDE de escritorio.

En resumen, Blockly tiene las siguientes características:

- Propósito: específico
- Área: editores de programación
- Relevancia: sí
- Aplicabilidad: no.

5.1.5 Code

Code es una herramienta de propósito específico cuyo objetivo es enseñar Informática a todos los niños. No es un lenguaje de programación (MathWorks, 2017).

Por sus características es considerado relevante en su área gracias a la gran acogida que ha tenido en los centros de enseñanza primaria. No se ha elegido como aplicable para los próximos análisis, por no considerarse un lenguaje de programación como tal.

En resumen, Code tiene las siguientes características:

- Propósito: específico
- Área: enseñanza de programación
- Relevancia: sí
- Aplicabilidad: no.

5.1.6 Cube

Cube fue el primer LPV tridimensional que por sus características de diseño 3D es considerado un lenguaje de propósito específico para realizar control automatizado. Sus programas se traducen en representaciones internas más simples, y se acerca mucho a ser un lenguaje completamente visual (Boshernitsan, 2004).

No es relevante, ni aplicable ya que su auge fue en la década del 2000.

En resumen, Cube tiene las siguientes características:

- Propósito: específico
- Área: control automatizado 3d
- Relevancia: no
- Aplicabilidad: no.

5.1.7 Drakon

Drakon es un lenguaje de programación algorítmico visual de propósito específico, ideado para el proyecto espacial soviético del transbordador Buran. Su objetivo era reemplazar los lenguajes especializados, como PROL2, DIPOL y LAKS, por un único sistema general (Mitkin, 2016).

A pesar de que es un proyecto reciente, no se considera relevante para el estudio y no fue clasificado como aplicable, porque está relacionado con un proyecto ruso, y toda la documentación encontrada está en este idioma.

En resumen, Drakon tiene las siguientes características:

- Propósito: específico
- Área: programación algorítmica
- Relevancia: no
- Aplicabilidad: no.

5.1.8 Fabrik

Fabrik es un lenguaje de programación de alto nivel de propósito específico para la creación de sistemas distribuidos abiertos con gran nivel de seguridad (Myers, 2016).

Se considera relevante en su área por ser un lenguaje de programación que controla la ubicación de la computación y los datos a través de anotaciones que establecen políticas para la seguridad de la información. Por sus características específicas de funcionalidad, no ha sido elegido como aplicable para los próximos análisis.

En resumen, Fabrik tiene las siguientes características:

- Propósito: específico
- Área: creación de sistemas distribuidos abiertos seguros
- Relevancia: sí
- Aplicabilidad: no.

5.1.9 Forms/3

Forms/3 es un lenguaje de programación visual de propósito general orientado a objetos, que se caracteriza por enfatizar la abstracción de datos. Forms/3 toma prestada la metáfora de la hoja de cálculo “de celdas y fórmulas” para representar datos y cálculos. Está basado en formularios que se les puede dar una representación pictórica (un icono) y puede ser instanciado en un objeto (Boshernitsan, 2004).

No se ha considerado como relevante ni aplicable ya que en la actualidad no es de mucho uso y su furor fue en la década de los 90.

En resumen, Forms/3 tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: no
- Aplicabilidad: no.

5.1.10 G

G es conocido también como G-code y es un lenguaje de propósito específico, mediante el cual las personas pueden decir a máquinas o herramienta controladas por ordenador qué hacer y cómo hacerlo (Wilensky, 1999).

A pesar de que su origen data de los años 80, en el año 2014 fue catalogado como el lenguaje más utilizado en la fabricación por ordenador. Se considera relevante, pero por su especificidad no ha sido elegido aplicable para los próximos análisis.

En resumen, G tiene las siguientes características:

- Propósito: específico
- Área: automatización y control
- Relevancia: sí
- Aplicabilidad: no.

5.1.11 GameSalad

GameSalad es un lenguaje de propósito específico para el desarrollo de juegos y que permite a cualquiera, crear juegos con una interfaz de programación visual. También es considerado como una herramienta para enseñar conceptos de programación y creación de medios digitales (GameSalad Inc, 2014).

Por sus características para el desarrollo de juegos en 2D para iPad e iPhone, ha sido utilizado para realizar más de 75 juegos reconocidos entre los mejores de la App Store, convirtiéndolo en una herramienta relevante en el entorno pero que no ha sido seleccionada como aplicable en los próximos análisis, gracias a su particularidad de programación para videojuegos.

En resumen, GameSalad tiene las siguientes características:

- Propósito: específico
- Área: Juegos 2D

- Relevancia: sí
- Aplicabilidad: no.

5.1.12 Max

Max es un entorno de desarrollo gráfico de propósito específico para música y multimedia. La API (Interfaz de Programación de Aplicaciones) permite el desarrollo de nuevas rutinas (llamadas «objetos externos») que se representan fácilmente de forma gráfica (Cycling, 2017).

Su última versión estable fue lanzada en el año 2015 pero, a pesar de esto, su uso es relevante únicamente en el entorno musical. Por sus características específicas no ha sido seleccionado como aplicable para ser incluido en los próximos análisis.

En resumen, Max tiene las siguientes características:

- Propósito: específico
- Área: música y multimedia
- Relevancia: sí
- Aplicabilidad: no.

5.1.13 Netlogo

Netlogo es un entorno de programación específico para la simulación de fenómenos naturales y sociales (Wilensky, 1999).

Su característica particular para modelar sistemas complejos que evolucionan en el tiempo lo convierten en un lenguaje relevante en su área, pero por su especificidad no ha sido considerado pertinente para ser incluido en los análisis posteriores.

En resumen, Netlogo tiene las siguientes características:

- Propósito: específico
- Área: simulación de fenómenos naturales y sociales
- Relevancia: sí
- Aplicabilidad: no.

5.1.14 Nxt-g

Nxt-g es lenguajes de cuarta generación específico para la creación de música y videos, con un entorno totalmente gráfico, el cual presenta herramientas en forma de iconos y que simplemente se “arrastran” a un área de trabajo para e ir interconectándolos y crear un programa (Lego Mindstorms México, 2016).

Actualmente no es relevante y prácticamente su uso se ha extinguido, y por su especificidad no ha sido considerado aplicable para ser incluido en los análisis posteriores.

En resumen, Nxt-g tiene las siguientes características:

- Propósito: específico
- Área: creación de música y video
- Relevancia: no
- Aplicabilidad: no.

5.1.15 Kodu

Kodu es un lenguaje orientado a eventos específico para la creación de videojuegos, que permite diseñar mundos en tres dimensiones a partir de una serie de elementos configurados previamente por el programa. Kodu no es únicamente un lenguaje de

programación visual, también es un entorno para diseñar, construir y jugar sus propias creaciones. Los elementos básicos del lenguaje son puramente iconos (Microsoft Student Partner, 2015).

Actualmente es considerado uno de los lenguajes de programación más relevantes y dada su popularidad ha sido seleccionado como aplicable para ser incluido en los análisis posteriores.

En resumen, Kodu tiene las siguientes características:

- Propósito: específico
- Área: creación de video juegos
- Relevancia: sí
- Aplicabilidad: sí.

5.1.16 Lava

Lava es un lenguaje de programación orientado a objetos, basado en un intérprete visual, con un entorno de programación asociado que usa a redactores de la estructura en vez de editores de textos (Helpes, 2016).

Fue considerado un lenguaje de uso general, pero sin relevancia y poco uso. Dado que no es considerado propiamente un LPV no es aplicable para ser incluido en análisis posteriores.

En resumen, Lava tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: no
- Aplicabilidad: no.

5.1.17 Limnor

Limnor es una plataforma abierta de programación visual y de propósito general, diseñada para que el programador agregue nuevas características fácilmente, instalando archivos DLL sin necesidad de cambiar o recompilar el sistema principal (Limnor Studio, 2017).

Fue diseñado para crear proyectos complicados y de desarrollo de software profesional, pero sin relevancia y poco uso. Por lo anterior no es aplicable para ser incluido en lo análisis posteriores.

En resumen, Limnor tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: no
- Aplicabilidad: no.

5.1.18 Prograph

Prograph es un lenguaje visual orientado a objetos que combina las nociones familiares de clases y objetos con un potente mecanismo visual de especificación del flujo de datos. Fue considerado el lenguaje visual de propósito general más exitoso (comercialmente) en la década de los 90 (Boshernitsan, 2004).

Actualmente no se considera relevante y por tal motivo no es aplicable para ser incluido en lo análisis posteriores.

En resumen, Prograph tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: no
- Aplicabilidad: no.

5.1.19 Pure data

Pure data es un lenguaje de programación visual de código abierto y propósito específico para desarrollo de multimedia (PD Community, 2015).

Dada su caracteriza de uso en ordenadores personales, dispositivos integrados (Raspberry Pi) y teléfonos inteligentes es considerado un entorno relevante, pero por su especificidad, no ha sido definido como un lenguaje aplicable para ser incluido en los análisis posteriores.

En resumen, Pure data tiene las siguientes características:

- Propósito: específico
- Área: multimedia
- Relevancia: sí
- Aplicabilidad: no.

5.1.20 Quartz Composer

Quartz Composer es una pequeña aplicación de uso específico para la programación de gráficos y que se instala con las Herramientas de desarrollo de Apple (Developer Tools). Quartz Composer no es un LPV, es una extraña herramienta, a medio camino entre el diseño multimedia y la pura diversión (Apple Inc, 2010).

Por sus potentes características, pero a la vez muy básicas, permite una explosión visual que requiere unos pocos conocimientos gráficos, algo de matemática y algo de práctica con el 3D, por lo que se convierte en una herramienta relevante en el diseño de apps para iOS. Por su especificidad y no ser considerado un LPV no ha sido definido como aplicable para ser incluido en los análisis posteriores.

En resumen, Quartz Composer tiene las siguientes características:

- Propósito: específico
- Área: programación de gráficos para las plataformas de Apple
- Relevancia: sí
- Aplicabilidad: no.

5.1.21 Robolab

Robolab (actualmente RobUAlab) es un sistema de uso específico que permite a estudiantes de asignaturas de Robótica, practicar comandos de posicionamiento en un robot industrial simulado, aprendiendo aspectos básicos de robótica, cinemática y diseño de trayectorias (LEGO Engineering, 2017).

En el área de la robótica es considerado una herramienta relevante, pero por sus características específicas, no ha sido definido como aplicable para ser incluido en los análisis posteriores.

En resumen, Robolab tiene las siguientes características:

- Propósito: específico
- Área: robótica

- Relevancia: sí
- Aplicabilidad: no.

5.1.22 Scratch

Scratch es un lenguaje de programación visual de propósito específico ya que sus funciones esenciales están enfocadas a introducir a los alumnos a otros lenguajes de programación más tradicionales. Obedece a una excelente herramienta la cual permite que, al desarrollar proyectos con ella, se aprendan conceptos matemáticos e informáticos fundamentales, como por ejemplo los procesos interactivos, las coordenadas en un plano o las variables (MIT Scratch Team, 2017).

En la actualidad se considera relevante y aplicable e incluso es usado en varios centros de formación como herramienta para iniciar los niños al mundo de la programación.

En resumen, Scratch tiene las siguientes características:

- Propósito: específico
- Área: enseñanza de la programación
- Relevancia: sí
- Aplicabilidad: sí.

5.1.23 Simulink

Simulink es un entorno de programación visual de uso específico usado para la simulación de modelos o sistemas, y que funciona sobre el entorno de programación Matlab (MathWorks, 2017).

Por sus potentes características, es considerado un entorno de programación de más alto nivel de abstracción que el lenguaje interpretado Matlab, lo que lo convierte en un entorno relevante, pero, por su especificidad, no ha sido definido como aplicable para ser incluido en los análisis posteriores.

En resumen, Simulink tiene las siguientes características:

- Propósito: específico
- Área: simulación
- Relevancia: sí
- Aplicabilidad: no.

5.1.24 Snap!

Snap! es una reimplementación ampliada de Scratch, y su nombre original era “Build Your Own Blocks” (BYOB). Este lenguaje de propósito general enfocado a la programación a partir del concepto de arrastrar y soltar permite construir sus propios bloques de código. Con una interfaz visual clara que separa las diversas funciones en diferentes categorías y colores, permite fácilmente aprender a programar *scripts* simples desde el mismo navegador, sin necesidad de instalar nada. Cada función (mover, rotar, incluir movimientos y sonidos), está diseñada en forma de pieza de puzle, para que con un simple ajuste con las demás piezas se pueda hacer un programa en secuencia (Brian H, 2017).

En la actualidad se considera relevante y específicamente por sus características visuales ha sido seleccionado como aplicable para ser incluido en los análisis posteriores.

En resumen, Snap! tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: sí
- Aplicabilidad: sí.

5.1.25 Stencyl

Stencyl es un entorno de desarrollo de propósito específico, que, partiendo de la filosofía “desarrolla una vez, júégalo en cualquier parte”, propone todas las herramientas necesarias para crear minijuegos en Flash para Web e iOS de una forma fácil, rápida y cómoda. No es necesario tener conocimientos de programación elevados (aunque cuando se llegue a cierto nivel sí resultan convenientes) (Stencyl, 2017).

En la actualidad se considera relevante; incluso su equipo de desarrollo se encuentra inmerso en el lanzamiento de la versión 3.0, la cual dará la posibilidad de exportación a HTML5 o Android entre otros. Por sus características específicas no ha sido seleccionado como aplicable para ser incluido en los análisis posteriores.

En resumen, Stencyl tiene las siguientes características:

- Propósito: específico
- Área: creación de video juegos
- Relevancia: sí
- Aplicabilidad: no.

5.1.26 Subtext

Subtext es un lenguaje de programación visual de propósito general para la escritura de software de aplicación. Es un intento de investigación experimental para desarrollar un nuevo modelo de programación, llamado “Ejemplo Centric programación” que funciona mediante el tratamiento de bloques copiados como prototipos de primera clase (Jonathan, 2017).

Es un proyecto en construcción y crecimiento que se considera relevante, pero por sus limitaciones en desarrollo no se ha considerado aplicable para ser incluido en los análisis posteriores.

En resumen, Subtext tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: sí
- Aplicabilidad: no.

5.1.27 ToonTalk

ToonTalk es un entorno de programación de propósito específico creado para enseñar a programar a los niños a partir de sencillos videojuegos. Los niños crean programas realizando acciones en un mundo animado (Ken, 2009).

Es un proyecto que no ha tenido la suficiente acogida como entorno de desarrollo, por lo que no se considera relevante ni aplicable para ser incluido en los análisis posteriores.

En resumen, Toontalk tiene las siguientes características:

- Propósito: específico
- Área: enseñanza de la programación

- Relevancia: no
- Aplicabilidad: no.

5.1.28 Agilent Vee Pro

Agilent Vee Pro es un entorno de programación gráfico de propósito específico, diseñado para aplicaciones de pruebas y medición, que reduce el tiempo de transmisión y conexión con instrumentos que permiten realizar mediciones, desplegar y reportar datos automatizado (Agilent Technologies, 2017).

Es una herramienta de gran acogida por lo que se considera relevante en su área, pero por sus características específicas, no ha sido seleccionada como aplicable para ser incluido en análisis posteriores.

En resumen, Agilent Vee Pro tiene las siguientes características:

- Propósito: específico
- Área: aplicaciones de prueba y medición
- Relevancia: sí
- Aplicabilidad: no.

5.1.29 VIPR

VIPR (“Visual Imperative Programming”) es un lenguaje de programación visual que representa un enfoque único para la programación de propósito general. Su ideología de uso de otras representaciones gráficas (como anillos) en lugar de iconos, llamó la atención en la década de los 90, pero rápidamente dejó de ser relevante (Boshernitsan, 2004).

No ha sido seleccionado como aplicable para ser incluido en los análisis posteriores.

En resumen, VIPR tiene las siguientes características:

- Propósito: general
- Área: programación
- Relevancia: no
- Aplicabilidad: no.

5.1.30 VISSIM

VISSIM es una herramienta de propósito específico para el modelo y simulación microscópica basado en tiempo y comportamiento. Fue desarrollado para modelar tráfico urbano, operaciones del transporte público y flujos de persona (SolidThinking, 2017).

En su área es relevante, pero por sus características específicas, no ha sido seleccionado como aplicable para ser incluido en los análisis posteriores.

En resumen, VISSIM tiene las siguientes características:

- Propósito: específico
- Área: modelo y simulación
- Relevancia: sí
- Aplicabilidad: no.

5.1.31 VVVV

VVVV es un entorno de programación gráfico de propósito específico para la creación de multimedia. Se caracteriza por el desarrollo y prototipado rápido, es decir, se

programa a través de nodos y estructuras que se conectan para realizar una acción específica (Joreg, 2014).

En su área es relevante, pero por sus características específicas, no ha sido seleccionado como aplicable para ser incluido en los análisis posteriores.

En resumen, VVVV tiene las siguientes características:

- Propósito: específico
- Área: creación de multimedia
- Relevancia: sí
- Aplicabilidad: no.

5.1.32 Resultado de la clasificación de los 31 lenguajes

En la Tabla 22 se resumen la clasificación de los 31 lenguajes de programación según la información anterior:

Tabla 22. Clasificación de los lenguajes de programación visual

Lenguaje	Variables analizadas			
	Propósito	Área	Relevancia	Aplicabilidad
Alice	Específico	Programación 3d	Sí	Sí
Appinventor	General	Programación	Sí	Sí
Ark	Específico	Simulaciones interactivas	No	No
Blockly	Específico	Editores de programación	Sí	No
Code	Específico	Enseñanza de programación	Sí	No
Cube	Específico	Control automatizado bajo 3D	No	No
Drakon	Específico	Programación algorítmica	No	No
FabriK	Específico	Creación de sistemas distribuidos abiertos seguros	Sí	No
Forms/3	General	Programación	No	No
G	Específico	Automatización y control	Sí	No
GameSalad	Específico	Juegos 2D para iPhone e iPad	Sí	No
Max	Específico	Música y multimedia	Sí	No
Netlogo	Específico	Simulación de fenómenos naturales y sociales	Sí	No
Nxt-g	Específico	Creación de música y video	No	No
Kodu	Específico	Creación de video juegos	Sí	Sí
Lava	General	Programación	No	No
Limnor	General	Programación	No	No
Prograph	General	Programación	No	No
Pure data	Específico	Multimedia	Sí	No
Quartz composer	Específico	Programación de gráficos para las plataformas Apple	Sí	No
Robolab	Específico	Robótica	Sí	No
Scratch	Específico	Enseñanza de programación	Sí	Sí
Simulink	Específico	Simulación	Sí	No
Snap!	General	Programación	Sí	Sí
Stencyl	Específico	Creación de video juegos	Sí	No
Subtext	General	Programación	Sí	No
Toontalk	Específico	Enseñanza de programación	No	No
Vee Pro	Específico	Aplicaciones de pruebas y Medición	Sí	No
Vipr	General	Programación	No	No
Vissim	Específico	Modelado y simulación	Sí	No
V V V V	Específico	Creación de multimedia	Sí	No

Aunque Blockly, Stencyl, Code y Subtext son herramientas relevantes, no se seleccionaron porque no se consideran lenguajes de programación como tal y éstas se ejecutan como una página web.

Para el estudio posterior, inicialmente se pensaba considerar solo los lenguajes de propósito general, pues el objetivo era preseleccionar un lenguaje capaz de realizar tareas de programación complejas. Sin embargo, dado que algunos lenguajes de programación con fines específicos se consideraron relevantes, aplicables y con características importantes, fueron seleccionados para los análisis posteriores. La clasificación fue resumida en la Tabla 23.

Tabla 23. Resumen de clasificación de los LPV

Clasificación	Cantidad
LPV no aplicables de fines específicos	5
LPV de propósito específico irrelevantes	16
LPV no aplicables de uso general	3
LPV de propósito general irrelevantes	2
LPV aplicables y relevantes para fines específicos	3
LPV aplicables y relevantes de uso general	2

Acorde con la metodología planteada y con relación al resultado “R2.1 Listado de lenguajes de programación visual relevantes y aplicables”, se seleccionaron 5 LPV para realizar los análisis posteriores: 3 lenguajes de uso específico, aplicables y relevantes (Alice, Kodu y Scratch) y 2 lenguajes de uso general, aplicables y relevantes (AppInventor y Snap!)

5.2 Caracterización de los principales lenguajes de programación visual

En los cinco lenguajes seleccionados, se analizaron las características descritas en el estado de la cuestión. A continuación, se relacionan los posibles valores que puede presentar cada característica:

- Lenguaje de implementación. Valores posibles:
 - Java (J).
 - Microsoft XNA (X).
 - Kawa (K).
 - Squek (S).
- Sistema operativo. Valores posibles:
 - Windows (W).
 - MacOS (M).
 - Linux (L).
 - Android (A).
 - Xbox 360 (X).
- Tipo de licencia. Valores posibles:
 - Licencia de código abierto (O).
 - Licencia de código cerrado (C).
- Control de flujo. Posibles valores:
 - Cerrar pantalla (C_S).
 - Cerrar pantalla con resultado (C_SR).
 - Contar (C).
 - Hacer (DO).
 - Hacer en orden (D_O).
 - Hacer juntos (D_T).
 - Cada uno en conjunto (E_IT).
 - Para (F).
 - Para cada uno (F_E).

- Para cada uno en (F_EI).
- Para siempre (F_E).
- Para el rango (F_R).
- Obtenga el texto de inicio (G_ST).
- Si (I).
- Si no (I_TE).
- Repetir (R).
- Repetir hasta (R_U).
- Cuando (W_H), mientras (W).
- Método de construcción visual. Posibles valores:
 - Arrastrar y soltar simple (DD) cuando hay varios tipos de bloques sin muescas que el usuario arrastra y suelta para crear el código.
 - Arrastrar y soltar (PDD) donde la mayoría de los bloques tienen muescas y estas deben encajar perfectamente como las piezas de un rompecabezas.
 - Visual (V) cuando hay un conjunto de iconos que gobiernan tareas específicas de construcción de programas.
- Sistemas de tipos. Posibles valores:
 - Estático (S).
 - Dinámico (D).
 - Estático-dinámico (SD).
- Programación concurrente. Posibles valores:
 - La programación concurrente está habilitada (Sí).
 - La programación concurrente no está habilitada (No).
- Abstracción procedimental. Posibles valores:
 - La abstracción procedimental está activada (Sí).
 - La abstracción procedimental no está activada (No).
- Abstracción de datos. Posibles valores:
 - La abstracción de datos está activada (Sí).
 - La abstracción de datos no está activada (No).
- Otro lenguaje. Posibles valores:
 - Java directamente (J_D).
 - XML directamente (XML_D)
 - XML por conversión externa (XML_E)
 - No permite la conversión (NP).

Los resultados obtenidos se resumen a continuación.

5.2.1 Descripción de los lenguajes seleccionados

A continuación, se hace una breve descripción de estos lenguajes.

5.2.1.1 Scratch

Scratch es un proyecto desarrollado por *Lifelong Kindergarten Group* en el *Media Lab* del Instituto Tecnológico de Massachusetts (MIT Scratch Team, 2017).

En la Figura 12 se puede observar el aspecto de la interfaz Scratch en el desarrollo de un programa, después de insertar algunas sentencias de repetición.

Figura 12. Interfaz de Scratch

Algunas de las principales características de Scratch son:

- Está construido bajo el lenguaje de programación Squeak² y se encuentra disponible para los entornos operativos Windows, Mac y Linux, bajo una licencia GPLv2, Scratch Source Code License.
- Presenta multiprocesamiento integrado en paralelo.
- Permite el multi-idioma (cambio de idioma de forma dinámica).
- Maneja sistema de tipos dinámico.
- Permite programación concurrente.
- No permite abstracción procedimental.
- No permite la abstracción de datos.

5.2.1.2 Kodu

Kodu fue desarrollado por *Microsoft Research* (Fuse Labs, 2017).

Algunas de las principales características de este lenguaje se relacionan a continuación:

- Está construido bajo Microsoft XNA³.
- Se encuentra disponible para el entorno operativo Windows Xbox360.
- Presenta Multiprocesamiento en paralelo.
- No maneja multi-idioma, solo está disponible en inglés.
- El sistema de tipos lo maneja mediante una combinación de estático-dinámico.
- Permite programación concurrente, donde un objeto puede tener hasta 12 páginas de código o 12 estados diferentes.
- No permite la abstracción procedimental
- No permite la abstracción de datos

En la Figura 13 se puede observar el aspecto de la interfaz principal de Kodu.

² Es un lenguaje reflexivo de programación, orientado a objetos y con tipado dinámico, portable y de código abierto, cuya máquina virtual está escrita enteramente como un entorno de objetos. Es una tecnología de más de 30 años y que hoy se mantiene vigente, considerada como lo mejor que existe para modelar la realidad de cualquier dominio que se quiera resolver mediante software.

³ Conjunto de herramientas, con un entorno de ejecución, administrado por Microsoft, que facilita el desarrollo y gestión de juegos de ordenador (Microsoft Research, 2004).

Figura 13. Interfaz principal de Kodu

5.2.1.3 Snap!

Snap! (antes BYOB) es un lenguaje de programación visual creado por Mönig y Harvey en la Universidad de Berkeley (California) (Brian, 2017).

Algunas de las principales características de este lenguaje se relacionan a continuación:

- Está construido bajo el lenguaje de programación Squeak.
- Se encuentra disponible para ser accedido desde la Web de forma gratuita.
- Se distribuye de acuerdo con la licencia Affero de GNU.
- Aún no dispone de una versión descargable.
- Sus autores afirman que el lenguaje combina la gramática de Scratch con las estructuras dinámicas de Scheme⁴ lo que permite crear estructuras de control propias
- No maneja multi-idioma, solo está disponible en inglés.
- Maneja un sistema de tipos estático.
- No permite programación concurrente.
- Permite la abstracción de datos.

En la Figura 14 se puede observar el aspecto de un programa desarrollado en Snap!.

⁴ Lenguaje funcional, dialecto de Lisp, desarrollado por Steele y Sussman.

Figura 14. Interfaz de Snap!

5.2.1.4 AppInventor

AppInventor fue creado originalmente por Google Labs, pero en la actualidad es un proyecto mantenido por el Instituto de Tecnología de Massachusetts (MIT Media Lab, 2015).

Algunas de las principales características de este lenguaje se relacionan a continuación:

- Está construido bajo el lenguaje de programación Kawa⁵.
- Se ejecuta en la Web, pero se debe instalar una pequeña aplicación para mostrar el editor de Bloques (Block Editor).
- Se distribuye para los sistemas operativos Windows, Mac OS y Linux bajo una licencia Freeware.
- Está disponible en inglés y español.
- Maneja sistema de tipos dinámicos durante la ejecución, en vez de ser en la compilación.
- Permite programación concurrente.
- Permite la abstracción de datos

En la Figura 15 se puede observar el aspecto de un programa desarrollado en AppInventor.

⁵ Lenguaje de programación que obedece a una implementación de Scheme, que se caracteriza por una buena integración con Java.

Figura 15. Interfaz de AppInventor

5.2.1.5 Alice

Alice es un lenguaje de programación orientado a objetos. Es el resultado de un proyecto de la Carnegie Mellon University que arrancó en 1999. Sus funciones esenciales están enfocadas a la programación 3D que permite crear animaciones para contar historias: un juego interactivo o un vídeo, una serie de objetos tridimensionales (personas, muebles, vehículos) que pueblan un mundo virtual, creando con ellos un programa para animarlos (Alice Team Carnegie Mellon University, 2017).

Algunas de las principales características de este lenguaje se relacionan a continuación:

- Está construido bajo Java
- Se encuentra disponible para los sistemas operativos Windows, Mac y Linux, bajo Licencia BSD (Berkeley Software Distribution).
- Maneja sistema de tipos estático.
- Permite programación concurrente.
- Permite abstracción de datos

En la Figura 16 se puede observar el aspecto de un programa desarrollado en Alice.

Figura 16. Interfaz Alice

5.2.2 Características Técnicas

Para los 5 lenguajes seleccionados anteriormente, se analizaron algunas de las características técnicas y de facilitación para programadores con dislexia.

La Tabla 24 muestra cada uno de los valores analizados.

Tabla 24. Resumen. Características técnicas de los LPV seleccionados

Características	Lenguajes				
	Alice	App Inventor	Kodu	Scratch	Snap!
Lenguaje de Implementación	J	K, J	X	S	S
Sistema operativo	W, L, M	W, L, M, A	W, X	W, M, L	W
Tipo de licencia	O	O	O	O	O
Método de construcción visual	DD	DDP	V	DDP	DD
Sistema de Tipos	S	D	SD	S	S
Programación concurrente	Yes	Yes	Yes	Yes	No
Abstracción procedimental	Yes	Yes	No	No	Yes
Abstracción de datos	Yes	Yes	No	No	Yes
Control de flujo	D_IO C I_TE W D_T E_IT	I, FE F_RW G_ST C_S CSR	DO W_H	I_TE R R_U F_E	I R F
Otro Lenguaje	J_D	J_E	N_P	XML_E	XML_D

5.2.3 Características facilitadoras del lenguaje de programación visual

La Tabla 25 expone las características que se analizan en este punto, las cuales están específicamente relacionadas con los problemas clave que los PCD enfrentan con el uso del ordenador. Dependiendo de los valores notificados, éstas pueden disminuir o aumentar los problemas identificados. En este contexto, dichas variables desempeñan un papel importante para el análisis de los lenguajes seleccionados.

Tabla 25. Resume Características facilitadoras para PCD en los LPV

Características	Lenguajes				
	Alice	App Inventor	Kodu	Scratch	Snap!
Nodos	ND	ND	DT	ND	ND
Nombres de identificadores	Y	N	N	Y	Y
Expresiones	M	N	N	Y	Y
Idioma	E, S	M	M	M	M
Complejidad de diagramas	H	M	M	M	M
Capacidad de zoom	N	N	Y	N	P
Nivel de detalle	N	N	N	N	N
Viscosidad de representación	M	H	L	M	M
Densidad de representación	H	L	L	H	H
Diccionario de íconos	N	N	S	N	N
Sintaxis	Y	Y	N	Y	Y

5.3 Selección de lenguaje visual

Después de clasificar los cinco LPV, y determinar sus principales características, se analizó brevemente cada uno para determinar el mejor lenguaje que debe ser incluido en los experimentos futuros.

Se encontró que Kodu es el LPV menos adecuado para la selección porque, en primer lugar, sus características técnicas sugieren que no es un lenguaje fuerte para procesos de programación complejos. Por otro lado, no es muy fuerte en algunas de las características facilitadoras para los programadores con dislexia, tales como el manejo de la sintaxis, la baja viscosidad de representación, y los nodos poseen demasiada documentación.

El segundo lenguaje menos adecuado, según el análisis, es Snap!, ya que, al igual que Kodu, le faltan características clave para procesos de programación complejos, como programación concurrente; sin embargo, con respecto a las características facilitadoras para los programadores con dislexia, se encontró que este lenguaje tiene algunos puntos positivos que el grupo de población podría encontrar útiles.

Aunque App Inventor tiene características técnicas sólidas para llevar a cabo procesos de programación complejos, fue rechazado debido a su alta viscosidad de representación. Ésta es una variable muy importante para manejar el estrés visual al que están expuestos los programadores con dislexia.

Los lenguajes Alice y Scratch son igualmente poderosos. Esto los determina como posibles lenguajes para su inclusión en el desarrollo de los estudios planteados, pero dado que Scratch no maneja la abstracción procedimental ni de datos, lo pone en desventaja como un buen lenguaje para desarrollar tareas de programación complejas; además, el hecho de que no esté asociado directamente con un lenguaje de programación textual y no permita la visualización del código generado, son razones suficientes para no seleccionarlo como candidato a incluir en los estudios futuros de éste trabajo.

Lo anterior, en respuesta al R2.2 “Lenguaje de programación visual seleccionado” planteado en la metodología, se ha seleccionado a Alice como el lenguaje que será utilizado en los experimentos de la presente Tesis.

6 Desempeño, satisfacción y efectividad de los PCD y PSD

Una vez seleccionado el lenguaje de programación visual a incluir en los experimentos, el siguiente objetivo planteado es OE3 “Determinar los niveles de desempeño, satisfacción y efectividad de programadores con y sin dislexia en el uso de un lenguaje visual y uno textual”.

El presente capítulo se desarrolla de acuerdo con la metodología de trabajo, que propone realizar cuatro actividades identificadas como: A3.1 “Clasificación de los lenguajes de programación textual más usados”, A3.2 “Experimento 1. Experiencias de desempeño, satisfacción y efectividad”, A3.3 “Experimento 2. Experiencias de desempeño, satisfacción, efectividad y medidas de calidad” y A3.4 “Comparación del desempeño de los programadores después de un periodo de tiempo de uso en ambos lenguajes”.

6.1 Clasificación de los lenguajes de programación textual más usados

Partiendo de los 51 LPT identificados como los más usados por los 315 programadores del primer estudio y basados en el Índice TIOBE que establece los lenguajes más utilizados a nivel mundial, se realizó una clasificación por frecuencia de uso, que ayudó a determinar el lenguaje de texto a incluir en los estudios posteriores (TIOBE Software BV, 2019).

La frecuencia. Determina el número de personas que utilizaron este lenguaje según el primer estudio.

La Tabla 26 clarifica la frecuencia y el porcentaje de uso de los 9 lenguajes de programación textual más utilizados por los 315 programadores.

Tabla 26. Frecuencia de uso de los lenguajes de programación textual

Lenguaje de Programación Textual	Frecuencia	Porcentaje
Java	293	83,5%
C	229	65,2%
JavaScript	161	45,9%
C++	149	42,5%
Python	134	38,2%
PHP	114	32,5%
C#	102	29,1%
Visual .Net	80	22,8%
M (MATLAB)	16	4,6%

Acorde con la metodología, la cual establece como R3.1 “Lenguaje de programación textual seleccionado” y después de clasificar y analizar los 9 lenguajes de programación textual identificados como los más utilizados por los 315 programadores, se determinó que Java sería el lenguaje ideal para ser incluido en los experimentos.

6.2 Experimento 1. Experiencias de desempeño, satisfacción y efectividad

La primera prueba buscaba comprobar las experiencias de los programadores con y sin dislexia en el uso de los lenguajes seleccionados, buscando validar la hipótesis y

proponer la PV, como una herramienta que, a futuro, pueda mejorar la integración entre programadores con y sin dislexia cuando trabajan en equipo.

6.2.1 Diseño del experimento

En el diseño del experimento se tuvieron en cuenta los aspectos que se tratan en los siguientes apartados.

6.2.1.1 Muestra

En este experimento se seleccionó una muestra a conveniencia, no probabilística, de 104 programadores colombianos, que resultaron entre los participantes del primer estudio, quienes manifestaron su interés en colaborar en la investigación. Tras aplicar los criterios de inclusión y exclusión, la muestra se redujo a 39 programadores y con éstos se desarrolló el experimento.

6.2.1.2 Población

La población está compuesta por varones y mujeres mayores de 16 años con y sin dislexia, estudiantes o profesionales de los programas del área informática del Politécnico Colombiano Jaime Isaza Cadavid de la ciudad de Medellín (Colombia).

6.2.1.3 Criterios de inclusión

Los siguientes criterios fueron definidos como requisitos para ser parte del experimento:

- Consentimiento de participación. Podían participar en el experimento las personas que manifestaran su intención de participación en trabajos futuros, después de asistir al primer estudio.
- Conocimiento del lenguaje de programación textual. Los interesados debían demostrar conocimientos medios en el manejo del lenguaje de programación textual seleccionado para el experimento.

6.2.1.4 Criterios de exclusión

Se define como único criterio de exclusión, el no aprobar un curso básico en el lenguaje de programación visual que resulte seleccionado para el experimento.

6.2.1.5 Variables de estudio

En la Tabla 27,

Tabla 28 y Tabla 29 se pueden observar las variables tenidas en cuenta para el experimento, separadas en 3 grupos: generales, de eficiencia y de satisfacción:

Tabla 27. Variables generales

Variable	Definición	Naturaleza		Valor o Categoría
Nro_id	Característica de identificación que permite relacionar al sujeto con los estudios anteriores	Cualitativa	Nominal	Número de identificación
Nivel_J	Determina el nivel de conocimiento en el uso del lenguaje de programación Java.	Cualitativa	Ordinal	Bajo Medio Alto
Nivel_A	Determina el nivel de conocimiento en el uso del lenguaje de programación Alice.	Cualitativa	Ordinal	Bajo Medio Alto

Variable	Definición	Naturaleza		Valor o Categoría
Dislexia	Determina si el resultado de una prueba de dislexia es positivo.	Cualitativa	Nominal	Sí No
Nivel_Educación	Determina si un sujeto es profesional o no.	Cualitativa	Nominal	Estudiante Profesional
Ansiedad_J	Característica de componente comportamental que determina el nivel de ansiedad experimentado por el programador al trabajar en Java	Cualitativa	Ordinal	Entre 1 y 5
Ansiedad_A	Característica de componente comportamental que determina el nivel de ansiedad experimentado por el programador al trabajar en Alice	Cualitativa	Ordinal	Entre 1 y 5
Concentracion_J	Característica de componente comportamental que determina el nivel de concentración experimentado por el programador al trabajar en Java	Cualitativa	Ordinal	Entre 1 y 5
Concentracion_A	Característica de componente comportamental que determina el nivel de concentración experimentado por el programador al trabajar en Alice	Cualitativa	Ordinal	Entre 1 y 5
Sintaxis_J	Característica de componente comportamental que determina la experiencia con relación a la facilidad de cometer errores de sintaxis cuando programa en Java	Cualitativa	Ordinal	Entre 1 y 5
Sintaxis_A	Característica de componente comportamental que determina la experiencia con relación a la facilidad de cometer errores de sintaxis cuando programa en Alice	Cualitativa	Ordinal	Entre 1 y 5

Tabla 28. Variables de eficiencia

Variable	Definición	Naturaleza		Categorías o valores
T_Desarrollo_Java_1 T_Desarrollo_Java_2	Característica de eficiencia que determina el tiempo tardado por un sujeto para desarrollar los ejercicios en Java	Cuantitativa	Discreto	Tiempo en minutos
T_Desarrollo_Alice_1 T_Desarrollo_Alice_2	Característica de eficiencia que determina el tiempo tardado por un sujeto para el desarrollo de los ejercicios en Alice	Cuantitativa	Discreto	Tiempo en minutos
Completado_Java_1 Completado_Java_2	Característica de eficiencia del programador que determina la finalización o no de los programas en Java	Cualitativa	Nominal	Sí No
Completado_Alice_1 Completado_Alice_1	Característica de eficiencia del programador que determina la	Cualitativa	Nominal	Sí No

Variable	Definición	Naturaleza		Categorías o valores
	finalización o no de los programas en Alice			
Precisión_Java_1 Precisión_Java_2	Característica de eficiencia que determina el nivel de exactitud de los resultados obtenidos en los programas desarrollados bajo Java, según casos de prueba	Cualitativa	Ordinal	Ninguno Bajo Medio Medio-Alto Alto
Precisión_Alice_1 Precisión_Alice_2	Característica de eficiencia que determina el nivel de exactitud de los resultados obtenidos en los programas desarrollados bajo Alice, acorde con los requerimientos iniciales y según casos de prueba	Cualitativa	Ordinal	Ninguno Bajo Medio Medio-Alto Alto
Nro_Errores_Java_1 Nro_Errores_Java_2	Característica de eficiencia del programador, que determina la cantidad de errores de compilación detectados en los programas de Java	Cuantitativa	Discreta	Cantidad de errores compilación
Nro_Errores_Alice_1 Nro_Errores_Alice_2	Característica de eficiencia del programador, que determina la cantidad de errores de compilación detectados en los programas de Alice	Cuantitativa	Discreta	Cantidad de errores de compilación

Tabla 29. Variables de satisfacción

Nombre de Variable	Definición	Naturaleza		Posibles valores
Satisfacción_A	Determina el nivel de satisfacción del usuario utilizando un entorno de programación visual.	Cuantitativo	Continuo	entre 1 y 100
Satisfacción_J	Determina el nivel de satisfacción del usuario utilizando un entorno de programación textual.	Cuantitativo	Continuo	entre 1 y 100

6.2.2 Desarrollo del experimento

Para el desarrollo del experimento se definieron cuatro fases secuenciales que se describen a continuación:

a) Fase 1. Prueba de riesgo de dislexia.

Con miras a validar la información recolectada en el primer estudio, del cual surgió la presente muestra, donde 15 de los 39 participantes (38,5%) afirmaron tener síntomas de dislexia, se utilizó “DyTECTive” (Change Dyslexia, 2016). Se identificó que 10 de los 39 (25,6%) estaban en riesgo de tener dislexia. De estos 10 participantes, 9 pertenecían al grupo que manifestaron tener dislexia en el primer estudio.

b) Fase 2. Desarrollo de programas.

Se definieron dos ejercicios para ser resueltos en cuatro pruebas, desarrollándose la misma actividad (primero en un lenguaje y luego en el otro). Al final de cada uno, se registraron los datos correspondientes a las variables generales y de eficiencia.

- Primera prueba. Los participantes desarrollaron un programa informático en Alice y luego en Java. Tiempo máximo: 60 minutos por programa.
- Segunda prueba. Los participantes desarrollaron un programa informático en Java y luego en Alice. Tiempo máximo: 120 minutos por programa. El ejercicio se describe a continuación:

Fundamentación. Los romanos utilizaban el sistema de numeración decimal, el mismo que utilizamos nosotros, lo particular de ellos era la forma de escribir esos números.

Solamente manejaban números naturales y no consideraban que el valor "nada" fuese un número, por eso el cero no se puede escribir en números romanos.

Hoy en día, este sistema es utilizado para ciertas referencias, no siendo utilizado de manera cotidiana, como en tiempos de antaño.

Para construir un número romano se tienen en cuenta varias reglas.

Como regla general los símbolos romanos se escriben y leen de izquierda a derecha, ubicados de mayor a menor valor.

La numeración romana utiliza siete letras mayúsculas a las que corresponden los siguientes valores

Letras	I	V	X	L	C	D	M
Valores	1	5	10	50	100	500	1.000

Si a la derecha de una cifra romana se escribe otra igual o menor, el valor de ésta se suma a la anterior.

Ejemplos: VI = 6.
 XXI = 21.
 LXVII = 67

La cifra "I" colocada a la izquierda de la "V" o la "X", les resta una unidad; la "X", precediendo a la "L" o a la "C", les resta diez unidades y la "C", delante de la "D" o la "M", les resta cien unidades.

Ejemplos: IV = 4.
 IX = 9.
 XL = 40.
 XC = 90.
 CD = 400.
 CM = 900.

En ningún número se puede poner una misma letra más de tres veces seguidas

Ejemplos: XIII = 13 XIV = 14
 XXXIII = 33 XXXIV = 34

La "V", la "L" y la "D" no pueden duplicarse porque otras letras ("X", "C", "M") representan su valor duplicado.

Ejemplos: X = 10 C = 100 M = 1.000.

Si entre dos cifras cualquiera existe otra menor, ésta restará su valor a la siguiente.

Ejemplos: XIX = 19 LIV = 54 CXXIX = 129

El valor de los números romanos queda multiplicado por mil tantas veces como rayas horizontales se coloquen encima de los mismos.

Requerimiento. Realizar un programa que lea n números enteros y que por cada uno se invoque una función que convierte dicho número a su equivalente en número romano, para posteriormente devolverlo al programa principal e irlo mostrando junto con su equivalente número leído.

c) Fase 3. Recopilación de información.

Los resultados obtenidos en la fase anterior se registraron mediante una encuesta electrónica (González, 2019).

6.2.3 Análisis estadístico

Acorde con la metodología planteada y los resultados establecidos para cada actividad se tiene para R3.2 “Análisis estadístico de desempeño, experiencias y satisfacción en el uso de cada lenguaje” lo siguiente:

Los datos capturados fueron analizados a través del software SPSS 23. Las tablas resúmenes de información arrojada por SPSS se pueden ver en (González, 2019). Los principales hallazgos se describen a continuación.

6.2.3.1 Pruebas de normalidad

Para una muestra de 39 individuos, y con el fin de determinar la normalidad de los datos, se aplicó una prueba Shapiro-Wilk cuyo objetivo era analizar los datos cuantitativos continuos (Shapiro, 1965). Se halló que los datos, reportados para estas variables provienen de una distribución normal, excepto para la variable ‘Edad’.

6.2.3.2 Análisis descriptivo

Para las variables cualitativas se realizó el cálculo de frecuencias absolutas y relativas. Para las cuantitativas se utilizaron medidas de tendencia central, dispersión y posición, acorde con la distribución de las variables.

Luego de realizar un análisis descriptivo de los datos -específicamente de las variables sociodemográficas- los principales comportamientos de la muestra se describen a continuación:

- Con relación a la edad de los participantes se encontró una media de 23 con un rango entre 17 y 53 años.
Las medidas de tendencia central relacionadas con la edad de los programadores son acordes con la muestra, dado que la mayoría de los participantes son estudiantes universitarios.
- Al observar la distribución por género se halló que 7 programadores, es decir, el 17,9% del total de los participantes, son mujeres y 32, es decir el 82,1%, son varones.
- La distribución de la muestra con relación al nivel educativo refleja que el 77,0% de los participantes son estudiantes y el 23,0% son profesionales.

Para las variables generales y de programación se obtienen los siguientes hallazgos:

- Con relación al conocimiento de los programadores acerca de los lenguajes se tiene que, para Java, la muestra se distribuyó así: bajo = 5, medio = 31 y alta = 3, mientras que, para Alice, se obtuvo: bajo = 8, medio = 28 y alta = 3.

Con relación al tiempo que tardaron los programadores en resolver los ejercicios, en la Tabla 30 se pueden observar los resultados al respecto. En la tabla se usaron las siguientes abreviaciones:

- Desviación estándar DE
- Intervalo de confianza IC

Tabla 30. Medidas de tendencia central MTC. Tiempo de desarrollo

Ejercicio	PCD		PSD			
	Mediana	IC	Media	DE	Media	DE
Primer ejercicio Alice	36	32,5-41,5	-	-	37,0	6,3
Primer ejercicio Java	-	-	43,3	7,2	45,3	7,2
Segundo ejercicio Java	-	-	71,6	20,4	81,1	25,1
Segundo ejercicio Alice	-	-	69,3	20,1	85,2	24,5

La Tabla 31 muestra un resumen del número de programas terminados y no terminados, clasificados por programadores con y sin dislexia.

Tabla 31. Resumen de programas finalizados y no finalizados

Ejercicio	PCD		PSD	
	Finalizado	No finalizado	Finalizado	No finalizado
Primer ejercicio Alice	10	0	24	5
Primer ejercicio Java	9	1	24	5
Segundo ejercicio Java	7	3	20	9
Segundo ejercicio Alice	9	1	19	10

Las medidas de tendencia central con relación a los errores presentados en el desarrollo de los programas incompletos al final del tiempo asignado se pueden observar en la Tabla 32.

Tabla 32. Medidas de tendencia central. Número de Errores

VARIABLES	PCD		PSD			
	N	Errores	N	Errores		
Numero_Errores_1 primer ejercicio Alice	-	-	-	5	Media = 4	DE = 2,4
Numero_Errores_2 primer ejercicio Java	1	6	-	10	Media = 6	DE = 3,5
Numero_Errores_1 segundo ejercicio Java	3	Media = 14	DE = 3,5	9	Mediana = 4	IC = 6-24
Numero_Errores_2 Segundo ejercicio Alice	1	5	-	10	Mediana = 3	IC = 2-7

Para la evaluación de la precisión, se realizaron 4 casos de prueba por cada ejercicio terminado; para su valoración se usó una escala puntuada de 1 a 5, donde 1 equivale

a 0% de precisión, 2 - 25% (1 resultado correcto), 3 - 50% (2 resultados correctos), 4 - 75% (3 resultados correctos) y 5 - 100% (4 resultados correctos). En la Tabla 33 se pueden observar los porcentajes de frecuencia por cada valor de la escala, separados por programadores con y sin dislexia.

Tabla 33. Porcentaje de precisión

Ejercicio	Programadores con dislexia			Programadores sin dislexia		
	Valor puntuación			Valor puntuación		
	5	4	3	5	4	3
1st Ejercicio (Alice)	100,0%	-	-	61,9%	33,3%	4,8%
1st Ejercicio (Java)	77,8%	22,2%	-	62,5%	29,2%	8,3%
2nd Ejercicio (Java)	57,1%	28,6%	14,3%	70,0%	25,0%	5,0%
2nd Ejercicio (Alice)	77,8%	22,2%	-	68,4%	10,5%	21,1%

Dada la importancia de las características ansiedad, concentración y nivel de error de sintaxis, identificadas en el perfil de los programadores con dislexia, se determinó medir el nivel de percepción de los programadores con relación a éstas cuando están desarrollando software: Para cuantificarlo, se usó una escala de Likert puntuada de 1 a 5 (donde 1 se entiende totalmente en desacuerdo y 5 totalmente de acuerdo). Para establecer el nivel de desacuerdo y de acuerdo, se agruparon los valores 1 y 2 y los valores 4 y 5. Los principales hallazgos antes las siguientes afirmaciones son:

- Cuando programo en Java a menudo me siento ansioso, el 80% de los programadores con dislexia estuvieron de acuerdo y el 6,9% de los programadores sin dislexia estuvieron de acuerdo.
- Cuando programo en Java sufro de baja concentración, el 90% de los programadores con dislexia estuvieron de acuerdo y el 10,3% de los programadores sin dislexia estuvieron de acuerdo.
- Cuando programo en Java cometo muchos errores de sintaxis, el 90% de los programadores con dislexia están de acuerdo y el 51,7% de los programadores sin dislexia están de acuerdo.
- Cuando programo en Alice a menudo me siento ansioso, el 80% de los programadores con dislexia no estaban de acuerdo, y el 69% de los programadores sin dislexia no estaban de acuerdo.
- Cuando programo en Alice, sufro de baja concentración, el 100% de los programadores con dislexia no estaban de acuerdo, y el 75,9% de los programadores sin dislexia no estaban de acuerdo.
- Cuando programo en Alice cometo muchos errores de sintaxis, el 100% de los programadores con y sin dislexia no están de acuerdo.

Por otro lado, para establecer el nivel de satisfacción de los programadores con y sin dislexia en el uso de los 2 lenguajes de programación propuestos, se adaptó un cuestionario de 10 preguntas (5 positivas y 5 negativas) basado en "*The System Usability Scale (SUS)*" (Brooke, 1996). Las respuestas fueron capturadas a través de una escala de Likert puntuada de 1 a 5 (donde 1 se entiende como totalmente en desacuerdo y 5 como totalmente de acuerdo). Posteriormente, cada respuesta se calificó de 0 a 5 siguiendo las mismas medidas establecidas por el método SUS, con el fin de establecer el nivel de satisfacción en una escala de 0 a 100. Los resultados fueron los siguientes:

a) Satisfacción con Alice. Con relación a la satisfacción de los programadores en el uso del lenguaje Alice, se halló:

- Media general de satisfacción de 68,7 (DE 6,7),
- Media de satisfacción de 71,3 (DE 4,5) para los programadores con dislexia

- Media de satisfacción de 67,9 (DE 7,1) para los programadores sin dislexia.

Al analizar por medio de una prueba “U de Mann-Whitney” los niveles de satisfacción (Figura 17), se encontró que las medianas de satisfacción en el uso del lenguaje Alice entre programadores con y sin dislexia son diferentes ($P = 0.013$).

Figura 17. Nivel de satisfacción lenguaje Alice. Experimento 1.

b) Satisfacción con Java. Con relación a la satisfacción de los programadores en el uso del lenguaje Java, se obtuvo:

- Media general de satisfacción de 69,7 (DE 8,4).
- Media de satisfacción de 64,3 (DE 3,9) para los programadores con dislexia
- Media de satisfacción de 71,6 (DE 8,7) para los programadores sin dislexia.

Al analizar, por medio de una prueba “U de Mann-Whitney”, los niveles de satisfacción de Java, se encontró que las medias de satisfacción entre programadores con y sin dislexia son diferentes ($P = 0.036$), como se puede observar en la Figura 18.

Figura 18. Nivel de satisfacción lenguaje Java. Experimento 1

6.3 Experimento 2. Experiencias de desempeño, satisfacción, efectividad y medidas de calidad

Se realizó un segundo experimento cuyo objetivo era establecer los niveles de desempeño y calidad, además de validar los niveles de satisfacción y de comportamiento de los programadores con y sin dislexia, luego de trabajar en ambos entornos de programación durante un periodo determinado.

6.3.1 Diseño experimental

Con relación al diseño del experimento a continuación se pueden observar los factores más relevantes:

6.3.1.1 Muestra.

Se realizó un muestreo a conveniencia no probabilístico, a partir de los programadores que participaron en el primer experimento, que cumplieran los criterios de inclusión y exclusión.

6.3.1.2 Población

La población estaba compuesta por varones y mujeres mayores de 16 años con y sin dislexia, estudiantes o profesionales de los programas del área informática del Politécnico Colombiano Jaime Isaza Cadavid de la ciudad de Medellín (Colombia).

6.3.1.3 Criterios de inclusión.

Se definieron como requisitos para ser parte del experimento, los siguientes:

- Consentimiento de participación.
- Haber participado en el primer experimento.

6.3.1.4 Criterios de exclusión

Como factores de exclusión del experimento, se tuvo:

- No programación continua en Alice. Se consideró como criterio de exclusión el no haber programado en el lenguaje visual seleccionado, durante el periodo comprendido en los 2 experimentos.
- No programación en Java. Se consideró como criterio de exclusión el no haber trabajado en el lenguaje de programación textual seleccionado durante el periodo comprendido en los 2 experimentos.

6.3.1.5 Variables de estudio

En la Tabla 34 se exponen las variables de calidad del software que fueron agregadas para esta prueba, ya que las otras, fueron definidas en el experimento 1 (ver Tabla 27,

Tabla 28 y Tabla 29).

Tabla 34. Variables de Calidad

Variable	Definición	Naturaleza		Categorías o valores
Lines_of_code	Características de calidad del software, que determinan la cantidad de líneas de código generadas en el programa	Cuantitativa	Discreta	Cantidad de líneas de código reportadas por SonarQube

Variable	Definición	Naturaleza		Categorías o valores
Functions	Características de calidad del software, que determinan la cantidad de funciones definidas por el usuario en un programa	Cuantitativa	Discreta	Cantidad de funciones reportadas por SonarQube
Rating	Características de calidad del software que determina el nivel de confiabilidad en una escala alfabética	Cualitativa	Ordinal	Calificaciones reportadas por SonarQube en la escala de: A, B, C, D
Code_Smell	Características de calidad del software que determina la cantidad de errores “code smell”	Cuantitativa	Discreta	Cantidad de errores de mantenibilidad reportados por SonarQube
Debt_Ratio	Características de calidad del software que determina el porcentaje de esfuerzo requerido para la corrección de errores de mantenibilidad en función del total de tiempo estimado para el desarrollo de un programa	Cuantitativa	Continuo	Porcentaje de esfuerzo
C_Cyclomatic	Características de calidad del software que determina el nivel de complejidad ciclomática del código, en el programa desarrollado	Cuantitativa	Discreto	Valor de complejidad ciclomática arrojado por SonarQube
C_Cognitive	Características de calidad del software que determina el nivel de complejidad cognitiva del código, en el programa desarrollado	Cuantitativa	Discreto	Valor de complejidad cognitiva arrojado por SonarQube
Issues	Determina la cantidad de errores de tipo “Bug”, “Vulnerability” y “code smell”	Cuantitativa	Discreto	Valor de Sugerencias arrojado por SonarQube
Lines_of_Comments	Característica de calidad que determina la cantidad de líneas de comentarios realizadas en un programa	Cuantitativa	Discreta	Cantidad de líneas de comentarios
D_Comments	Característica de calidad que determina la densidad de las líneas comentadas con relación al total	Cuantitativa	Continuo	Porcentaje de densidad de comentarios
Duplicate_Lines	Característica de calidad que determina la cantidad de líneas de código duplicadas	Cuantitativa	Discreta	Cantidad de líneas de código duplicadas
Density	Característica de calidad que determina la densidad de líneas duplicadas con relación al total	Cuantitativa	Continuo	Porcentaje de duplicidad

6.3.1.6 Metodología del experimento

Para la dinámica de la prueba se crearon 2 grupos, conformados por programadores con y sin dislexia.

A pesar de la conformación de los grupos, el desarrollo del experimento fue individual; se dividió en 3 fases:

- a) **Fase uno.** Capacitación de auxiliares. Para la evaluación de los programas desarrollados y la aplicación de pruebas de calidad y rendimiento, se capacitaron 4 personas en el uso de los diferentes programas utilizados. Esta labor se realizó previa al día central del experimento.
- b) **Fase dos.** Desarrollo de programas. Se definieron dos ejercicios para ser ejecutados en 2 etapas, como se observa en la Tabla 35.

Tabla 35. Etapas para el desarrollo de la fase dos

Etapas	Programa	Tiempo en Minutos	Equipos	Lenguaje
1	Ejercicio 1	90	Grupo A_PCD. Grupo A_PSD. Grupo B_PCD. Grupo B_PSD.	Visual Visual Textual Textual
2	Ejercicio 2	120	Grupo B_PCD. Grupo B_PSD. Grupo A_PCD. Grupo A_PSD.	Textual Textual Visual Visual

- c) **Fase 3.** Recopilación de información. Se realizaron diversas actividades tendientes a establecer medidas de calidad, satisfacción y comportamiento. En la Tabla 36 se clarifican las diferentes actividades y especificaciones:

Tabla 36. Actividades para el desarrollo de la fase 3

ETAPA	ACTIVIDAD	HERRAMIENTA	RESPONSABLE
1	Transformación	Plugin Java (instalado con anticipación)	Cada integrante
2	Recolección de información de eficiencia, comportamental, y satisfacción	Encuesta impresa	Auxiliares
3	Evaluaciones medidas de rendimiento, calidad	Software SonarQube	Investigador
4	Análisis estadístico de los datos	Software SPSS 23	Investigador

6.3.2 Desarrollo del experimento

De 39 programadores que participaron en el experimento anterior, 35 cumplieron los criterios de inclusión y exclusión definidos.

- Con relación a los criterios de inclusión, consentimiento de participación, de los 39 programadores candidatos para el experimento, 3 decidieron no continuar.
- Respecto al criterio de exclusión: no programación en Alice, una persona fue rechazada por este concepto, para una muestra final de 35 programadores

Para el desarrollo de la fase 2 del experimento se procedió como se relata a continuación:

Se crearon los grupos, conformados como se relaciona a continuación:

- Grupo A_PCD. Programadores con síntomas de Dislexia (n = 4)
- Grupo A_PSD. Programadores sin síntomas de Dislexia (n = 13)
- Grupo B_PCD. Programadores con síntomas de Dislexia (n = 5)
- Grupo B_PSD. Programadores sin síntomas de Dislexia (n = 13)

Posteriormente, se configuró el entorno de trabajo inicial para Alice, creando el método vacío para tener el punto particular de arranque que permitiera generar el resto del código requerido para la solución; se establecieron los puntos particulares de lectura para la entrada de datos y de escritura para la salida de datos.

A continuación, se hizo la importación desde NetBeans del código Alice generado hasta el momento. De esta forma se garantizó un punto de partida de código equivalente tanto para los programadores de Java como Alice.

Para las pruebas se definieron 2 ejercicios que se desarrollaron primero en un lenguaje y luego en el otro. Los ejercicios se describen a continuación:

a) Ejercicio 1. La conjetura de los números primos de Waring plantea que cada número entero impar, es primo o equivale a la suma de 3 números primos. Por otro lado, la conjetura de Goldbach, plantea que cada número entero par, es la suma de 2 números primos.

En este problema se tiene una tarea un poco menos exigente: Hallar una forma de expresar una serie de números enteros dados, como la suma exacta de 4 números primos.

Datos de entrada

- Cantidad de números
- Números enteros mayores que 3 y menores que 10000 (según la cantidad)

Datos de salida

Para cada uno de los números entrados, imprimir una línea de salida -estándar- con 4 números primos mayores o iguales a 3, cuya suma es igual al número ingresado; por ejemplo, para el caso del 24, se tendrían las siguientes posibles soluciones:

```
3 3 5 13
3 3 7 11
3 5 5 11
3 7 7 7
5 5 7 7
```

Si el número no puede ser expresado como una suma de 4 números primos, imprimir el mensaje "Imposible" en una sola línea.

Dato de Prueba

Ejemplo de entrada.	Ejemplo de salida.
4	
24	3 3 5 13
36	3 3 7 23
46	3 3 3 37
11	Imposible
Ejemplo de entrada.	Ejemplo de salida.
10	

24	3 3 5 13
36	3 3 7 23
46	3 3 3 37
11	Imposible
1072	3 3 3 1063
15	Imposible
28	3 3 3 19
5771	Imposible
10000	3 3 5 3
12	9 9 4 1

b) Ejercicio 2. La historia relata que durante una guerra contra los romanos, Joseph y 40 soldados judíos fueron rodeados por el ejército enemigo. El grupo decidió morir antes que rendirse ante el adversario, aunque Joseph y su colega no estaban de acuerdo con esta absurda idea. El método que escogieron para suicidarse fue el de formar en un círculo e irse matando cada tercera persona. Joseph como era un buen matemático, seleccionó para él y su amigo las posiciones con las que serían los últimos. Cuando solo quedaron ellos dos, salieron a rendirse ante los romanos.

Realizar un programa que dado un número entre 10 y 50, que corresponde a la cantidad de soldados de una compañía, devuelva las posiciones de los 2 salvados.

Datos de entrada:

Número de soldados de una compañía (mayor igual a 10 y menor igual a 50). Si ingresa un número no válido debe mostrar el mensaje: "Número no válido!".

Datos de salida:

Mostrar por la salida estándar las posiciones en que deben ubicarse los 2 que se salvarán en la compañía.

Datos de prueba

Datos de Entrada	Datos de Salida
15	5 14
25	4 14
11	2 7
7	¡Número no válido!
10	4 10
45	2 8
35	4 3

Para finalizar la fase 2, se hizo entrega del primer ejercicio a los programadores de los grupos A_PCD y A_PSD para que lo desarrollasen en Alice y a los grupos B_PCD y B-PSD en Java. Pasados 90 minutos se dio por finalizado la primera etapa de la fase 2. Cada participante entregó el código en Java (NetBeans) del programa desarrollado, (esto quiere decir que quienes trabajaron en Alice debieron realizar la transformación a Java). Si algún participante finalizaba su ejercicio antes del plazo máximo debía informar para que se registrara el tiempo tardado en la solución, y recibiera el código transformado a Java si era necesario.

Posteriormente, se hizo entrega del segundo ejercicio a los programadores de los grupos A_PCD y A-PSD para que lo desarrollaran en Java y a los grupos B_PCD y B-PSD para que lo desarrollen en Alice. Pasados 90 minutos se dio por finalizada la primera etapa de la fase 2, entregando el código en Java (NetBeans) del programa desarrollado. Igual que en la actividad anterior, en caso de que acabaran antes, debían informar para registrar el tiempo.

Para el desarrollo de la fase 3, los participantes respondieron una encuesta impresa de evaluación del software (ver anexo A).

Lo anterior, permitió comparar el trabajo de los programadores y el código resultante, en 2 entornos de desarrollo diferentes con el mismo ejercicio, que al ejecutarlo debía de tener una apariencia exactamente igual.

Después de finalizado el experimento, los programas desarrollados en Alice fueron sometidos a un proceso de conversión a Java y posteriormente se aplicaron las diferentes pruebas de calidad. Cabe destacar que cuando un programador trabaja en este software visual, tiene una opción que permite ver en formato Java el código que se está generando, pero que no puede ser editado por el usuario.

Para llevar a cabo el proceso de transformación de Alice a Java, es necesario instalar en “NetBeans” un *plugin* suministrado por el equipo de Alice; quien ha creado una librería NetBeans que permite desarrollar o ampliar mundos Alice directamente en el entorno de NetBeans. Esto le permite cargar los mundos guardados desarrollados en la plataforma Alice 3 y continuar modificándolos directamente en el código Java (Alice Team Carnegie Mellon University, 2017).

Todos los programas terminados fueron sometidos a pruebas de calidad utilizando el software SonarQube, que corresponde a “una plataforma de código abierto para realizar revisiones automáticas con análisis estático del código para detectar errores y vulnerabilidades de seguridad en más de 25 lenguajes de programación...” (SonarQube.org, 2017). Allí se analizaron las siguientes medidas:

a) Complejidad. Se analizaron los siguientes factores:

- Complejidad cognitiva: una nueva medida diseñada en 2016 para medir la comprensibilidad y hacer un breve estudio de los problemas de Complejidad Cognitiva en un subconjunto de proyectos de código abierto (Ann Campbell, 2018).
- Complejidad ciclomática: fue introducida en 1976 para identificar módulos de software que eran difíciles de probar o mantener (McCabe, 1976).

b) Duplicaciones. En esta medida se refiere a detalles específicos de un programa; para el experimento se analizó la densidad de líneas duplicadas.

c) Mantenibilidad. Se analizaron los siguientes factores:

- Calificación de Mantenibilidad: se relaciona con el valor del esfuerzo para corregir todo el “smell code”.
- Porcentaje de la relación de deuda: se refiere al costo de desarrollar el software y el costo de fijarlo en los programas desarrollados.

d) Tamaño. Esta medida se refiere a detalles específicos de un programa; se seleccionaron las siguientes:

- Líneas de código, que es el número de líneas físicas que contienen al menos un carácter que no es un espacio en blanco ni una tabulación ni parte de un comentario.
- Funciones: número de funciones implementadas en un programa.
- Líneas de comentario, que es el número de líneas que contienen comentarios.
- Porcentaje de comentarios: es equivalente a la densidad de las líneas de comentario en relación con las líneas de código.

e) Problemas. En esta medida se analizó el conteo total de incidencias en todos los estados, incluyendo incidencias de “código Smell”, de vulnerabilidad y de errores.

6.3.3 Análisis estadístico

Acorde con la metodología planteada y los resultados establecidos para cada actividad, se tiene para “R3.2 Análisis estadístico de medidas de calidad de los programas desarrollados”, lo siguiente:

Los datos capturados fueron analizados a través del software SPSS 23 (IBM Corp, 2015). Las tablas resúmenes de información arrojada por SPSS se pueden ver en (González, 2019). Los principales hallazgos se describen a continuación.

6.3.3.1 Pruebas de normalidad

Para una muestra de 35 individuos, con el fin de determinar la normalidad de los datos cuantitativos continuos, se aplicó una prueba Shapiro-Wilk (Shapiro, 1965). Los datos reportados para estas variables provienen de una distribución normal, excepto para la variable ‘edad’ y ‘satisfacción’, filtrada por dislexia = 0 (No).

6.3.3.2 Análisis descriptivo

Para las variables cualitativas, se realizaron cálculos de frecuencias absolutas y relativas.

Para las variables cuantitativas se utilizaron medidas de tendencia central, dispersión y posición, acorde con la distribución de las variables.

En cuanto al número de ejercicios terminados en el tiempo definido para el experimento, se puede establecer que los programadores con dislexia (PCD) muestran un mayor nivel de éxito que los programadores sin dislexia (PSD). Además, el tiempo que tardan en resolver los ejercicios, según las medidas de la tendencia central, es menor en los PCD, independiente del lenguaje que utilicen.

Al analizar el comportamiento de los programadores que no completaron los ejercicios en el tiempo asignado, se observa que la mayor frecuencia de fracasos se presentó en los PSD, independiente del lenguaje de programación analizado.

Para la medición de las variables generales, específicamente las relacionadas a las experiencias comportamentales experimentadas durante el desarrollo de los programas, se usó una escala de Likert puntuada de 1 a 5 (donde 1 se entiende totalmente en desacuerdo y 5 totalmente de acuerdo). Para establecer el nivel de desacuerdo y de acuerdo, se agruparon los valores 1 y 2 y los valores 4 y 5 respectivamente. Los principales hallazgos para cada una de las variables definidas, ante las siguientes afirmaciones son:

- **Ansiedad_J.** Cuando programo en Java a menudo me siento ansioso, el 88,9% de los programadores con dislexia estuvieron de acuerdo y el 0% de los programadores sin dislexia estuvieron de acuerdo.
- **Concentración_J.** Cuando programo en Java sufro de baja concentración, el 88,8% de los programadores con dislexia estuvieron de acuerdo y el 0% de los programadores sin dislexia estuvieron de acuerdo.
- **Sintaxis_J.** Cuando programo en Java cometo muchos errores de sintaxis, el 77,8% de los programadores con dislexia están de acuerdo y el 23% de los programadores sin dislexia están de acuerdo.
- **Ansiedad_A.** Cuando programo en Alice a menudo me siento ansioso, el 100% de los programadores con dislexia no estaban de acuerdo, y el 73,1 de los programadores sin dislexia no estaban de acuerdo.

- **Concentración_A.** Cuando programo en Alice, sufro de baja concentración, el 100% de los programadores con dislexia no estaban de acuerdo, y el 65,4 de los programadores sin dislexia no estaban de acuerdo.
- **Sintaxis_A.** Cuando programo en Alice cometo muchos errores de sintaxis, el 100% de los programadores con dislexia no están de acuerdo y el 96,1% de los programadores sin dislexia no estaban de acuerdo.

Los anteriores resultados permiten concluir que:

- Los PCD son más ansiosos que los PSD cuando trabajan en un lenguaje textual como Java, en cambio, su ansiedad baja cuando están programando en Alice.
- En Java los PCD tienen bajos niveles de concentración, mientras que los PSD tienen niveles medios de concentración; en Alice, los PCD tienen un alto grado de concentración, mientras que los PSD tienen un nivel de concentración medio-alto.
- Respecto a los errores de sintaxis, se tiene que en Java los PCD cometieron más errores que los PSD, mientras que en Alice no reportan errores de sintaxis frecuentes.

Todos estos hallazgos afianzan las afirmaciones realizadas por otros investigadores mencionados en capítulos anteriores con relación a las particularidades de los programadores con dislexia, cuando trabajan con herramientas de programación visual; esto ratifica los resultados del capítulo 3 que establece las características de los programadores de computadores con dislexia.

Ahora bien, para establecer el nivel de satisfacción de los programadores con y sin dislexia en el uso de los dos lenguajes de programación propuestos, se adaptó un cuestionario de 10 preguntas (5 positivas y 5 negativas) basado en "*The System Usability Scale (SUS)*" (Brooke, 1996). Las respuestas fueron capturadas a través de una escala de Likert puntuada de 1 a 5 (donde 1 se entiende como totalmente en desacuerdo y 5 como totalmente de acuerdo). Posteriormente, cada respuesta se calificó de 0 a 5 siguiendo las mismas medidas establecidas por el método SUS, con el fin de establecer el nivel de satisfacción en una escala de 0 a 100.

- **Satisfacción_J.** Con relación a la satisfacción de los programadores en el uso del lenguaje Java, se halló:
 - Media general de satisfacción de 58,9 (DE 18,8).
 - Media de satisfacción de 31,9 (DE 7,9) para los programadores con dislexia.
 - Media de satisfacción de 68,3 (DE 10,4) para los programadores sin dislexia.

Al analizar los niveles de satisfacción por medio de una prueba "U de Mann-Whitney", se encontró que las medianas de satisfacción en el uso del lenguaje Java entre programadores con y sin dislexia son diferentes ($P=0,000$), como se puede observar en la Figura 19.

Figura 19. Nivel de satisfacción lenguaje Java. Experimento 2

- **Satisfacción A.** Con relación a la satisfacción de los programadores en el uso del lenguaje Alice, se obtuvo:
 - Media general de satisfacción de 58,3 (DE 14,2).
 - Media de satisfacción de 76.4 (DE 6,7) para los programadores con dislexia.
 - Media de satisfacción de 52.0 (DE 10,1) para los programadores sin dislexia.

Estudiando los niveles de satisfacción por medio de una prueba “U de Mann-Whitney”, se evidenció que las medianas de satisfacción en el uso del lenguaje Alice entre programadores con y sin dislexia son diferentes ($P=0,000$) según la Figura 20.

Figura 20. Nivel de satisfacción lenguaje Alice. Experimento 2

Las diferencias halladas entre los niveles de satisfacción de los PCD cuando trabajan en Java y Alice dan a entender que éstos no se sienten satisfechos cuando trabajan en un lenguaje de programación textual como Java, estando incluso por debajo del valor mínimo de aceptación definido para la escala -el cual es de 65- mientras que los PSD presentan mayores niveles de satisfacción en el uso de este.

Las medidas de tendencia central, en relación con las medidas de calidad, reportadas por el software SPSS son:

a) Pruebas de complejidad

- Complejidad ciclomática:
 Alice: **PCD** 54,6 (DE 31,2) **PSD** 67,2 (SD 28,2)
 Java: **PCD** 28,1 (DE 26,2) **PSD** 16,9 (SD 23,0)
- Complejidad cognitiva
 Alice: **PCD** 146,7 (DE 125,4) **PSD** 165,7 (SD 126,2)
 Java: **PCD** 14,9 (DE 7,3) **PSD** 9,2 (SD 5,3)

b) Pruebas de duplicidad

- Densidad de líneas duplicadas
 Alice: **PCD** 3,4% (DE 8,3) **PSD** 2,6% (DE 2,7)
 Java: **PCD** 1,7 (DE 4,4) **PSD** 1,5 (DE 3,3)

c) Pruebas de mantenibilidad

- Porcentaje de la proporción de deuda
 Alice: **PCD** 0,6% (DE 0,5) **PSD** 0,8% (DE 0,7)
 Java: **PCD** 0,4% (DE 0,2) **PSD** 0,3% (DE 0,1)

d) Pruebas de tamaño

- Líneas de código
 Alice: **PCD** 1.411 (DE 162) **PSD** 1.403 (DE 225)
 Java: **PCD** 1.432 (DE 90) **PSD** 1.367 (DE 138)
- Funciones
 Alice: **PCD** 19 (DE 13) **PSD** 18 (DE 10)
 Java: **PCD** 8 (DE 5) **PSD** 2 (DE 2,3)
- Líneas de comentarios
 Alice: **PCD** 176 (DE 43) **PSD** 145 (DE 35)
 Java: **PCD** 148 (DE 7) **PSD** 107 (DE 54,3)
- Densidad de comentarios
 Alice: **PCD** 11,1% (DE 2,0) **PSD** 9,4% (DE 2)
 Java: **PCD** 9,3% (DE 0,3) **PSD** 7,1% (DE 3,5)

e) Pruebas de Problemas

- Total de problemas
 Alice: **PCD** 39 (DE 39) **PSD** 20 (DE 26,7)
 Java: **PCD** 15 (DE 7) **PSD** 9,2 (DE 53)

6.4 Comparación de desempeño de los programadores

Acorde con la metodología planteada y los resultados establecidos para cada actividad se tiene para R3.3 “Análisis estadístico comparativo”. A continuación, se exponen los resultados hallados.

Con el fin de comparar los resultados de ambos experimentos, y poder realizar pruebas pareadas, se filtró la base de datos de la primera prueba para dejar únicamente los participantes que tomaron parte de la segunda. En la Tabla 37 se pueden observar los resultados de satisfacción de ambos experimentos; se usan las siguientes abreviaciones:

Primer experimento de Java Java_I
 Segundo experimento de Java Java_F
 Primer experimento de Alice Alice_I
 Segundo experimento de Alice Alice_F
 Rango mínimo Min
 Rango máximo Max

Tabla 37. Comparativo de satisfacción. Experimento 1 y 2

Ejercicio	PCD n = 9				PSD n = 26				Totales n = 35			
	Min	Max	Media	DE	Min	Max	Media	DE	Min	Max	Media	DE
Java_I	60,0	72,5	64,7	3,8	60,0	90,0	71,6	9,2	60,0	90,0	69,9	8,7
Java_F	22,5	47,5	31,9	7,9	45,0	82,5	68,3	10,4	22,5	82,5	58,9	18,8
Alice_I	67,5	80,0	71,9	4,1	60,0	82,5	67,3	6,4	60,0	82,5	68,5	6,2
Alice_F	67,5	87,5	76,4	6,7	35,0	70,0	52,0	10,1	35,0	87,5	58,3	14,2

Dado que al realizar las pruebas de normalidad se halló que los datos de los PCD (n = 9) se distribuyen normalmente, se realizó una prueba de “T pareada” que permitió establecer si existen diferencias estadísticamente significativas entre los dos experimentos, después de un periodo de 10 meses que transcurrió entre éstos. La Tabla 38 muestra los resultados.

Tabla 38. Prueba de T pareada. Satisfacción Java y Alice PCD

Nivel de Satisfacción	Media	Diferencia	Valor p*
Java_I	64,722	-32,8	0,000
Java_F	31,944		
Alice_I	71,944	4,4	0,031
Alice_F	76,389		

*Valores de p calculados con un Alpha de 0,05

Las diferencias halladas en ambos experimentos entre los niveles de satisfacción de los PCD cuando trabajan en Java y Alice dan a entender que no se sienten satisfechos cuando trabajan en un lenguaje de programación textual como Java, estando incluso por debajo del valor mínimo de aceptación definido para la escala -el cual es de 65- mientras que los PSD presentan mayores niveles de satisfacción en el uso de éste.

De otro lado, las diferencias para los PCD dan a entender que se sienten más satisfechos cuando trabajan en un lenguaje de programación visual como Alice, mientras que los PSD presentan niveles contrarios, estando, incluso, en el límite del valor mínimo de aceptación definido para la escala.

Con relación a los resultados de la prueba “T pareada” se puede afirmar que la satisfacción de los PCD en el uso del lenguaje Java, disminuye considerablemente en el tiempo, mientras que para el lenguaje Alice, aumenta, incluso ubicándose 10 puntos por encima del valor mínimo sugerido.

Al realizar las pruebas de normalidad, los datos de los PSD (n = 26) no se distribuyen normalmente, por lo que se realizó una prueba no paramétrica de “Wilcoxon” que permitió establecer si existen diferencias estadísticamente significativas entre los 2 experimentos, después del periodo de 10 meses que transcurrió entre éstos. La Tabla 39 muestra los resultados.

Tabla 39. Prueba de Wilcoxon Satisfacción Java Alice. PSD

Nivel de Satisfacción	Valor p*
Java_I	0,365
Java_F	
Alice_I	0
Alice_F	

*Valores de p calculados con un Alpha de 0,05

La prueba no paramétrica de “Wilcoxon”, realizada a las medianas de satisfacción de los PSD en el uso del lenguaje Java, establecen que no hay diferencias estadísticamente significativas a lo largo del tiempo, aspecto que se esperaba que sucediera, ya que no existen factores que generaran diferencias; con relación a Alice sí existen diferencias estadísticamente significativas.

7 Estrategia para la integración de programadores con y sin dislexia

Tras identificar las bondades que el lenguaje de programación visual Alice puede brindar a los programadores de computadores con dislexia, y teniendo en cuenta que el lenguaje de programación Java es de gran aceptación en el mercado, además de ser identificado por el índice TIOBE como el más utilizado actualmente a nivel mundial (TIOBE Software BV, 2019), se han analizado las particularidades de cada uno de estos lenguaje, con el fin de establecer las diferentes relaciones entre los elementos de éstos y las relaciones entre ambos lenguajes.

7.1 Análisis de particularidades y relaciones de Alice y Java

Con relación a la actividad “A4.1 Análisis de particularidades y de relación entre los elementos de los lenguajes visual y textual seleccionados” definida en la metodología, a continuación, se describe con más detalle cada uno de estos lenguajes con el fin de establecer las diferentes relaciones entre sus elementos y así, poder definir relaciones entre ambos lenguajes. Los principales resultados relacionados con lo anterior se describen a continuación.

7.1.1 Alice

Según (Eaton, 2018), las siguientes son los diferentes componentes de Alice son los indicados a continuación (**¡Error! No se encuentra el origen de la referencia.**).

Figura 21. Diagrama de componentes de Alice.

Entities: Se muestran las dependencias de los principales componentes,

En la figura anterior se observa una interfaz en la cual los operadores se pueden conectar, debido a que un método, una función o un evento pueden enviar como parámetro uno de estos tipos.

Las Entities se conectan a una interfaz de *DataTypes*, debido a que una *class* tiene propiedades y estas tienen un tipo de dato.

De igual manera las *operations* se conectan a la interfaz de *Data types* porque las funciones y métodos pueden retornar un tipo de dato, ya sea en sus parámetros o en la funcionalidad implementada.

Los *contracts* también se conectan a un *Data Types* debido a que estos realizan validaciones.

7.1.1.1 Entidades

Las principales entidades de Alice interactúan con las demás estructuras del lenguaje, debido a que tienen diversos comportamientos.

- **Scene.** Es una animación en 3D que interactúa con varios objetos de códigos.
- **Class.** Plantilla a partir de la cual se puede crear una instancia a un objeto, puede tener propiedades y métodos.
- **Object.** Es un actor o cosa que se coloca en la escena de Alice. Interactúa con los objetos usando métodos.
- **Properties.** Son datos que representan la descripción de un objeto en Alice.

7.1.1.2 Operaciones

Las operaciones en Alice están encapsuladas en las entidades y éstas son las que definen el comportamiento del objeto.

- **Method.** Un método es una acción o instrucción dada a un objeto.
- **Function.** Una función le permite al programador hacer preguntas sobre las diversas interacciones y relaciones de los objetos en la escena.
- **Events.** Un evento es una acción que ocurre a la que un programador puede responder.
- **Parameters.** Un parámetro es un marcador de posición de un tipo de datos específico.

7.1.1.3 Tipos de datos

Los diferentes tipos de datos que se pueden definir en Alice son:

- **Boolean.** Un tipo de datos lógico representa un valor que es verdadero o falso.
- **Object.** Un tipo de datos de objeto representa una referencia a un objeto.
- **String.** Un tipo de datos cadena representa datos textuales.
- **Number.** Este tipo de dato representa un valor numérico.

En Alice, el número puede ser entero (un número entero como 1, 8 o 357) o un número de coma flotante (un número con decimales como 3.14), a diferencia de otros lenguajes en los que estos tipos de datos siempre están separados.

7.1.1.4 Constructs

- **Conditionals.** Condicionales probará si una expresión es verdadera o falsa.
- **Loops.** Los bucles son una estructura de repetición. Van a recorrer un bloque de código según sea necesario.

- **WhileLoops.** Son estructuras que repiten el código según una condición.

7.1.1.5 Relaciones de Alice

En el proceso de análisis de las estructuras y componentes de Alice, se identifican las relaciones y dependencias de este lenguaje, que se pueden observar en la Figura 22.

Figura 22. Diagrama de relaciones de Alice.

En la figura se puede observar que todos los tipos de datos en Alice heredan de una clase y ésta a la vez puede tener propiedades.

Un objeto usa la clase encapsulando todas las funcionalidades que tiene y éste puede tener uno o varios eventos.

Un objeto puede tener uno o muchos métodos; un método y una función pueden tener parámetros.

7.1.2 Java

Según (Oracle, 2017c) las siguientes son los principales componentes de Java.

7.1.2.1 Operadores

Los principales operadores en Java y su respectiva descripción se pueden observar en la Tabla 40.

Tabla 40. Operadores Java

Operadores	Precedencia	Descripción
Unario	expr++ expr--	Ejecuta una expresión e incrementa o decrementa en una unidad
Unario	++expr --expr +expr -expr ~ !	Decremento o aumenta en una unidad y ejecuta una expresión. Más y menos unarios. Negación de bits y negación lógica
Multiplicativo	* / %	Operaciones aritméticas básicas
Aditivo	+ -	Operaciones aritméticas básicas
Cambio	<< >> >>>	Desplazamientos

Operadores	Precedencia	Descripción
Relacional	< > <= >= instanceof	Comparaciones
Igualdad	== !=	Comparaciones
Y de bits	&	Operación AND de bits
O exclusivo de bits	^	Operación XOR de bits
O de bits		Operación OR de bits
Y lógico	&&	Para unir varias condiciones y se deben cumplir todas para que sea verdadero
O lógico		Para unir varias condiciones y con una que se cumpla es verdadero
Condicional	? :	Expresión condicional
Asignación	= += -= *= /= %= &= ^= = <<= >>= >>>=	Asignaciones

7.1.2.2 Sentencias de flujo de control

- **Sentencias de decisión.** Permiten tomar una decisión para poder ejecutar un bloque de instrucciones u otro.
 - if-then
 - if-then-else
 - switch (permite evaluar múltiples decisiones)
- **Sentencias de bucle.** Permiten ejecutar un bloque de sentencias tantas veces como se quiera, o como se cumpla una condición.
 - while (bucle que se ejecuta mientras se cumple la condición, pero puede que no se llegue a ejecutar nunca, si no se cumple la condición la primera vez.
 - do-while (Permite que el bloque de sentencias se ejecute, al menos, una vez.
 - for (permite escribir toda la estructura del bucle de una forma más concreta. Si bien, su cometido es el mismo.
- Sentencias de ramificación. permiten romper con la ejecución lineal de un programa.
 - break
 - continue
 - return
 -

7.1.2.3 Relaciones

Basados en la revisión de literatura se construyó un diagrama de relaciones que se puede observar en la Figura 23. Éste se aprecia más granular, debido a que los datos se segmentan en numéricos y objetos. Los flujos de control se segmentan en condicionales, bucles y derivación:

7.1.3 Equivalencias entre Alice y Java

Para realizar el establecimiento de la equivalencia entre ambos lenguajes, se empezó con una perspectiva a alto nivel, inicialmente con diagramas de relaciones de cada lenguaje. Luego se pasó por el diagrama de componentes, el cual permite saber cómo se comunican entre sí y cuáles son sus relaciones. A continuación, se analiza a un nivel más bajo para determinar qué estructuras en Java corresponden a Alice y viceversa.

En la Tabla 41 se observa la relación de estructuras de asignación y ramificación.

Tabla 41. Correspondencia estructuras de asignación y ramificación

Alice	Java
Assign	=
No aplica	The break
No aplica	The continue

En la Tabla 42 se observa la relación entre los tipos de datos.

Tabla 42. Correspondencia Tipos de datos

Alice	Java
Boolean	Boolean
TextString	String
WholeNumber	Long
	Int
	Short
	Byte
DecimalNumber	Float
	Double
Array	Array
No aplica	Character
No aplica	Number

En la Tabla 43 se observa la relación de estructuras.

Tabla 43. Correspondencia de estructuras

Alice	Java
Class	Class
GalleryClass	No aplica
Objects	Objects
Scene	No aplica
Events	Events
Method	Method
Function	Function
Parameters	Parameters
Assign	=
No aplica	Enums
No aplica	Interfaces

En la Tabla 44 se observa la relación de flujos de control.

Tabla 44. Correspondencia de control de flujo

Alice	Java
If	If-then
Else	If-then-else
No aplica	Switch
No aplica	Do-While
While	While
ForEach in	For
Do In Order	No aplica
Count	For
Do Together	doTogether
Each in Together	eachInTogether

7.2 Propuesta de integración

Con relación a la actividad “A4.3 Propuesta de integración”, descrita en la metodología y teniendo en cuenta que las relaciones halladas entre los dos lenguajes (Alice y Java) viabilizan el diseño de una herramienta que interactúe entre ambos, se propone el desarrollo de una interfaz que facilite la interacción de programadores al desarrollar software.

La interfaz propuesta permitirá que cada integrante de un equipo de trabajo, donde uno de los programadores presente síntomas de dislexia, pueda seleccionar el entorno de su preferencia (sea textual o visual).

El objetivo de la interfaz es permitir la transformación bidireccional de códigos de lenguajes entre Alice y Java, de tal forma que cada programador tenga acceso al código realizado por su compañero de trabajo, independiente del entorno de desarrollo (visual o textual) que ambos hayan seleccionado.

La propuesta de un modelo de transformación a través de una interfaz de desarrollo que convierta código basado en estándares visuales (específicamente bajo el lenguaje Alice) a código de texto (lenguaje Java) y viceversa, puede llegar a ser un gran aporte al desarrollo de software, ayudando a los programadores con síntomas de dislexia, pues éstos podrán seleccionar en una sola interfaz el entorno de programación preferido, mientras su compañero observa en otra ventana dentro de la misma interfaz lo realizado por su par, pero en el entorno seleccionado por él. Esto se daría en tiempo real y con actualización permanente en cada uno de los 2 lenguajes de programación.

Aunque esta propuesta no llegará a solucionar todos los problemas que genera la dislexia en los programadores, sí puede ayudar en el proceso de interacción de éstos con el ordenador y su compañero de equipo de desarrollo, ayudando a mejorar la integración entre programadores con y sin dislexia y convirtiéndose a la vez en una propuesta que aumentaría los niveles de calidad del software desarrollado por equipos de programadores con estas características.

La idea de poder brindar una herramienta de fácil uso para los PCD puede ayudar como factor motivacional. Según investigaciones recientes, las personas con síntomas de dislexia que abordan carreras del área de informática usualmente presentan altos niveles de desmotivación y baja autoestima, por todos los efectos que genera la dislexia en el uso del ordenador.

Con miras a realizar un acercamiento a lo que puede ser el desarrollo del modelo de transformación bidireccional de códigos, se establecen las relaciones entre programadores con y sin dislexia, representando el proceso que van a realizar éstos; para posteriormente, poder especificar la funcionalidad y comportamiento del modelo. En la Figura 25 se observa el diagrama de casos de uso, que hace un acercamiento a lo que puede ser el proceso de transformación de códigos.

Figura 25. Diagrama de casos de uso

En el desarrollo de la propuesta de integración, se realizaron un conjunto de actividades basadas en prácticas definidas por la ingeniería del software, que llevaron a la implementación de un prototipo, que demuestra la posibilidad de transformación bidireccional de código entre Alice y java. Este prototipo solo realiza la transformación de una estructura específica.

Al crear un proyecto en Alice, este genera una estructura base definida, debido a que implanta una *scena* y una estructura por defecto, como se ve en la parte izquierda de la Figura 26 e igualmente de código Java, como se observa en la parte derecha y crea un método vacío en el cual se enfocara el proceso de transformación.

Figura 26: Estructura inicial de un programa en Alice

Para el proceso de transformación de código, en primera instancia se debía conocer el tipo de información generada en un archivo .3ap de Alice y se realizaron las siguientes áreas

- Inicialmente se intentó decodificar este archivo y se realizaron pruebas de revisión en GitHub (herramienta de control de versiones).
- Usar métodos utilizados para decodificar imágenes a base 64 y analizar resultados.

Dado que las dos actividades anteriores no arrojaron ninguna información útil para el proyecto se decidió solicitar información a los creadores de Alice. Se obtuvo respuesta por parte del equipo del proyecto Alice en donde se menciona como está conformado el archivo, el cual contiene 3 archivos, que son los siguientes:

- ProgramType.XML. Este archivo contiene un XML en donde está segmentado el código generado en Alice.
- Thumbnail.png: contiene la imagen de la pizarra de Alice.
- Version.txt: contiene la versión de Alice utilizada.

Para ver todos estos archivos se debe realizar un proceso que consiste en cambiar el nombre de la extensión del archivo de “.3ap” por “.zip”, descomprimirlo y finalmente se podrá ver la información en su interior.

Análisis del archivo XML

El archivo XML generado por Alice contiene muchos datos, ver Figura 27

Figura 27. Apariencia del XML

Partiendo de lo anterior se presenta la inquietud de qué información está en el XML y si este tiene también una estructura base, para lo cual se plantearon los siguientes interrogantes:

- **Pregunta 1.** ¿Debido a que este XML es generado dinámicamente, puede tener un XSD que facilite la comprensión de su estructura?
- **Pregunta 2.** ¿Todo este XML es necesario para crear un proyecto en Alice?
- **Pregunta 3.** ¿Qué pasa si borro una parte de este XML?
- **Pregunta 4.** ¿Este XML tiene un patrón?

- **Pregunta 5.** ¿Para acceder a la información necesariamente se debe hacer con un XML o se puede convertir a otra estructura?
- **Pregunta 6.** ¿Dónde se puede ver de una manera más organizada el XML?

Con estas preguntas iniciales se comenzó a consultar y a realizar laboratorios, para poder solucionar estos interrogantes y tomar decisiones de cómo implementar los algoritmos y se obtuvieron las siguientes respuestas a cada uno de los interrogantes planteados:

- **Respuesta pregunta 1.** El XML no cuenta con un XSD, por lo tanto, no se puede validar su estructura.
- **Respuesta pregunta 2.** Sí, es necesario debido a que es la base inicial de todo proyecto en Alice.
- **Respuesta pregunta 3.** Si se borra cualquier segmento del XML pasan dos situaciones: Al borrar una apertura de la etiqueta hasta el cierre, si dicha etiqueta no se usa después, solo se pierde la referencia de la etiqueta borrada. Sin embargo, si luego se usa genera error. Si se borra una parte de la etiqueta un atributo genera error.
- **Respuesta pregunta 4.** Aparentemente el XML tiene una estructura, se analiza y se crea un modelo que se puede observar en la Figura 28.
- **Respuesta pregunta 5.** Como el XML no tiene un XSD que proporcione más información al respecto, podría convertirse en un JSON, debido a que se tiene más experiencia con esta estructura y a que es más sencillo de manipular.
- **Respuesta pregunta 6.** Existen páginas que permiten ver de manera estructurada el XML y editores de texto como Notepad ++ y Visual Studio Code.

Basado en las respuestas, se decidió convertir en JSON el XML para para realizar el proceso de análisis de éste. A continuación, se observa las estructuras del XML y el JSON.

Figura 28. Modelo del XML

Figura 29. Modelo del JSON

Partiendo del archivo Json se analiza que valores pueden corresponder a las estructuras de Alice y posteriormente se creó un repositorio en el que se iban adicionando estructuras y haciendo uso de GitHub se determinaban las diferencias del código generado como se puede observar en Figura 28 y Figura 29.

```

1341 </property>
1342 <property name="expression">
1343 +
1344 <node type="org.igna.project.ast.InstanceCreation" uuid="8a758632-e682-45e1-ba1d-40554501c7ab">
1345 <property name="constructor">
1346 <node key="40" type="org.igna.project.ast.JavaConstructor" uuid="31d8cde9-9522-47fa-a7a6-4ce3fc6c2439">
1347 <constructor isVarArgs="false">
1348 @@ -1355,45 +1355,45 @@ DO NOT EDIT</value>
1349 </property>
1350 <property name="requiredArguments">
1351 <collection type="java.util.ArrayList">
1352 -
1353 <node type="org.igna.project.ast.SimpleArgument" uuid="125dd15c-487e-4573-abac-341d739a8a31">
1354 <property name="parameter">
1355 <node key="41" type="org.igna.project.ast.JavaConstructorParameter" uuid="540cb2ff-b2b8-4a32-ad57-451fd32809c8">
1356 <node key="40"/>
1357 </index></index>
1358 </node>
1359 </property>
1360 <property name="expression">
1361 -
1362 +
1363 </property>
1364 </property>
1365 </property>
1366 +

```

Figura 30. Primer análisis con GitHub

```

1600 </property>
1601 </node>

```

```

1602 +
1603 <node type="org.ignia.project.ast.ConditionalStatement" uuid="0fc19c0b-d321-4e4b-a164-c98559decff">
1604 +
1605 <property name="booleanExpressionBodyPairs">
1606 <collection type="java.util.ArrayList">
1607 +
1608 <node type="org.ignia.project.ast.BooleanExpressionBodyPair" uuid="2a543e7d-24ed-44cf-b6e4-9e3de1f4a68">
1609 +
1610 <property name="expression">
1611 +
1612 <node type="org.ignia.project.ast.MethodInvocation" uuid="fe574e7a-3c9f-425a-9698-31e24a12e755">
1613 +
1614 <property name="expression">
1615 +
1616 <node type="org.ignia.project.ast.StringLiteral" uuid="d7655d72-6a65-4125-b92e-e5f4322ff0c">
1617 +
1618 <property name="value">
1619 +
1620 <value type="java.lang.String">meencontro</value>
1621 +
1622 </property>
1623 +
1624 </node>
1625 +
1626 </property>
1627 +
1628 </collection>
1629 +
1630 </property>
1631 +
1632 </node>
1633 +
1634 </property>
1635 +
1636 </property name="method">
1637 +

```

Figura 31. Segundo análisis con GitHub

Lo anterior permite concluir que Alice genera unos identificadores únicos y que cada vez que se adiciona una estructura nueva, este identificador se vuelve a generar. Además, cada estructura queda agrupada, es decir, si se adiciona un if todo lo que corresponde a este queda dentro de una etiqueta padre llamada “node” y desde ahí se desprenden las diferentes estructuras que componen el *if* ingresado.

7.2.1 Algoritmo para la conversión de Alice a Java

Partiendo del análisis anterior se comenzó a implementar los algoritmos y se decide que el algoritmo inicial sería de Alice a Java, debido a que se tiene un referente que es el código en Java que muestra Alice, además de que se tiene como insumo el XML para iniciar. Se define un flujo básico para crear algoritmo de subida como se muestra en la Figura 32.

Figura 32. Diagrama de para la conversión

En este proceso se identificó que se debe crear una función para recorrer el JSON, puesto que es necesario ir identificando las estructuras creadas para la transformación.

7.2.1.1 Características importantes

Se observa que el proceso es repetitivo, sin embargo van cambiando las estructuras, las cuales finalmente convergen en un solo punto, dado que siempre será un JSON.

Por lo tanto, se concluye que la función debe ser recursiva para reutilizar y optimizar código.

Para la solución de esta función se implementa el diagrama de flujo, que se puede observar en la Figura 33

Figura 33. Diagrama de flujo conversión

7.2.2 Desarrollo del prototipo

Para la creación del prototipo se usaron diferentes tecnologías de desarrollo

3.1 Mockups

Estos tienen el objetivo de "mostrar el contenido de las páginas" (Rosenfeld, 2002), es un prototipo, en donde los usuarios pueden ver cómo va a quedar la página y así llegar a un acuerdo para iniciar el desarrollo.

Para empezar el diseño de la plataforma y tener un referente, tanto para el desarrollador como para el *productowner*, se realizan unos *mockups* que validen la aceptación.

La figura 34 representa la pantalla inicial y la separación por zonas, para adicionar el contenido estático de la plataforma, tal como información de interés para los usuarios.

Figura 34. Interfaz inicial del prototipo

La Figura 35 muestra una página que permita subir un archivo.

Figura 35. Interfaz subir archivo

8 Conclusiones y trabajos futuros

En este trabajo de Tesis se abordaron diferentes aspectos relacionados con la presencia de la dislexia en personas que programan ordenadores, habiéndose establecido a través de varios estudios los siguientes resultados:

- Principales características de comportamiento y programación de este grupo poblacional.
- Un posible lenguaje de programación visual que ayude a mitigar los efectos de la dislexia.
- Niveles de desempeño y satisfacción en el uso de los lenguajes Alice y Java.
- Propuesta de modelo para la integración de programadores con y sin dislexia

Las principales conclusiones relacionadas con cada resultado se abordan en el apartado 8.1.

En la búsqueda de soluciones a la pregunta de investigación planteada, han surgido nuevas ideas que, bien sea porque modificaban la planificación del trabajo a realizar o porque iban más allá de las pretensiones de esta Tesis, no han sido exploradas, pero constituyen el origen de nuevas líneas de investigación. Estas ideas se presentan al final del capítulo en el apartado 8.2.

8.1 Conclusiones

Los estudios realizados en los diferentes capítulos han permitido obtener conclusiones particulares para cada objetivo planteado, las cuales se describen a continuación.

8.1.1 OE1 “Determinar las principales características y dificultades de los programadores con dislexia, cuando desarrollan software, a partir del análisis de su comportamiento físico, emocional y conductual” .

En el capítulo 4, se han definido las principales características de los programadores de computadores con dislexia, el cual se considera nuevo conocimiento, pues a la fecha no existía una caracterización de este grupo poblacional. Es importante resaltar que la mayoría de las características emocionales propuestas por (Shaywitz, 1996). para definir el perfil de los adultos con dislexia, no fueron relevantes en los resultados obtenidos, acorde con las respuestas de los encuestados, lo que puede ayudar a inferir que el grupo de PCD estudiado, tiene particularidades específicas de comportamiento que puede suponer diferencias sustanciales con los adultos con dislexia.

Vale la pena resaltar que la diferencia de porcentajes de prevalencia de la dislexia en Colombia -entre los datos hallados en el presente estudio y los reportados en el capítulo 2- se atribuye a que estos últimos corresponden a una población infantil, lo que afianza la suposición de que la dislexia en los adultos es una diversidad funcional olvidada.

Se ha logrado ratificar que los programadores con dislexia tienen preferencia por la programación visual, además de cometer menos errores cuando utilizan un lenguaje de programación textual.

Los programadores de computadores con síntomas de dislexia que participaron en el estudio, al desarrollar software en un lenguaje de programación textual a menudo

pueden verse afectados por aspectos de su personalidad como la ansiedad y la baja concentración.

8.1.2 OE2 “Construir una taxonomía de la programación visual que permita la selección de un lenguaje apropiado para los programadores con dislexia”

Con relación a este objetivo, se puede establecer que un lenguaje que tenga las características de Alice, puede ser la base para la construcción de un entorno de programación ideal y apropiado para los programadores con dislexia. Alice parece ser a priori, un lenguaje con el que los programadores con dislexia se pueden sentir cómodos al trabajar, en lugar de un lenguaje de programación basado en texto; sin embargo, aunque sus diagramas son bastante complejos, tiene características visuales que pueden ser de ayuda para los programadores con dislexia, permitiendo a éstos la opción de minimizar los errores de sintaxis, uno de los mayores problemas que afrontan los programadores con dislexia a la hora de desarrollar software.

8.1.3 OE3 “Determinar los niveles de desempeño, satisfacción y efectividad de programadores con y sin dislexia en el uso de un lenguaje visual y uno textual”

En el capítulo 6 se han realizado dos experimentos y diversos análisis estadísticos y se resaltan los siguientes aspectos:

Al comparar la Figura 19 y Figura 20, se puede observar claramente que hay unos niveles de satisfacción inversamente proporcionales para programadores con y sin riesgo de dislexia entre el uso de ambos lenguajes, lo que afianza la hipótesis de que los PCD prefieren la programación visual versus la textual.

Después de analizar los resultados de los experimentos anteriores y poder demostrar que los PCD, cuando trabajan en un LPT, registran menores índices de desempeño y las medidas de calidad del software desarrollado son inferiores a las registradas en los LPV, se valida la hipótesis “H1: **Los programadores con dislexia experimentan dificultades al trabajar en un lenguaje de programación textual**”. Se puede afirmar que la programación visual es quizás mejor para los programadores con síntomas de dislexia, lo que, a futuro, puede ayudar al proceso de integración de programadores con y sin dislexia, cuando trabajan en equipo.

Aunque no existen estudios que demuestren que las personas con dislexia tienden a tener un coeficiente mental más alto, durante el desarrollo de los diferentes estudios se ha observado que los PCD muestran mejores habilidades a la hora de resolver determinados problemas, presentando mayores niveles de precisión en los resultados requeridos.

Los resultados de los experimentos refuerzan las teorías propuestas en la caracterización de los programadores con dislexia. Por otro lado, los altos porcentajes de satisfacción hallados en el uso del lenguaje Alice, validan la hipótesis: “H2. **Los programadores con dislexia tienen mejor desempeño cuando trabajan en un lenguaje de programación visual**”.

8.1.4 OE4 “Proponer una estrategia que facilite la integración de programadores con dislexia”

Las relaciones halladas entre los lenguajes de programación Alice y Java, viabilizan la propuesta de diseño de una interfaz de transformación bidireccional de códigos,

con miras a brindar una herramienta que ayude a integrar los programadores con y sin dislexia cuando desarrollan software en equipo.

8.2 Trabajo futuro

Un trabajo de investigación como el desarrollado en la presente Tesis podría no tener fin, ya que los estudios realizados son susceptibles a ampliaciones, dada la diversidad de la población estudiada y de las distintas manifestaciones que tiene la dislexia en cada persona.

Los estudios realizados en esta investigación se han concentrado en una población específica -de una muestra que puede ser demasiado grande- al igual que se limitó a una propuesta de solución que abre infinitas posibilidades de proyectos de investigación, relacionados con la dislexia en los programadores de ordenadores. Por lo anterior, se proponen tres líneas específicas de trabajo futuro, que pueden llegar a tener gran relevancia e impacto, en esta área que se encuentra un poco olvidada.

8.2.1 Caracterización de PCD angloparlantes

En esta Tesis, el estudio se centró en dos universidades de habla hispana, pero las diversas manifestaciones de la dislexia dependiendo de la lengua materna de cada persona, constituyen otra línea futura de investigación.

El desarrollo de un estudio similar en población angloparlante se justifica en la afirmación de Rello: “la dislexia es una diversidad funcional que tiene diferentes manifestaciones, y sus síntomas se pueden acentuar dependiendo del idioma nativo de las personas que sufren de este trastorno” (Rello, 2012).

Estudios realizados por Lallier y otros investigadores establecen ciertos niveles de dificultades -incluso diferentes a los hispanohablantes- que las personas disléxicas angloparlantes manifiestan a la hora de leer y escribir. (Lallier, 2013).

Al revisar la literatura, no existen estudios registrados que relacionen estas dificultades con los programadores de computadores.

De lo expuesto anteriormente se plantea la siguiente pregunta:

¿Cuáles son las principales características de los PCD angloparlante y sus influencias en el desarrollo de software?

8.2.2 Estudio de desempeño, satisfacción y calidad

Dados los hallazgos significativos con relación al uso de Alice, que ratifican las hipótesis planteadas en el presente estudio, se justifica la idea de desarrollar un estudio similar con otro lenguaje de programación visual.

El presente estudio arrojó que el lenguaje de programación Scratch, es una herramienta de uso frecuente y tiene fortalezas de programación que también pueden ayudar a los PCD.

Por lo expuesto se requiere conocer ¿cómo son los niveles de desempeño y satisfacción de los PCD y las medidas de calidad del software desarrollado por éstos, cuando trabajan en un LPV como Scratch frente al uso de Java?

8.2.3 Lenguaje de programación Visual para PCD

Por otro lado, se plantea el desarrollo de un lenguaje de programación basado en Alice pero que no esté enfocado a la animación, sino más bien al desarrollo de software comercial, que aplique en su diseño las directrices de accesibilidad para

personas con dislexia, de tal forma que los usuarios puedan personalizar el entorno de trabajo acorde con las necesidades especiales de su diversidad. Este entorno de programación puede ir asociado a un módulo que facilite la transformación bidireccional de códigos visuales y textuales, poniendo en práctica la estrategia de integración de programadores con y sin dislexia propuesta en el capítulo 6 de esta Tesis.

Dado lo anterior se requiere conocer ¿cómo sería un lenguaje de programación visual de uso general para PCD y que facilite la interacción de programadores con y sin síntomas de dislexia, cuando desarrollan software en equipo?

9 Referencias bibliográficas

- Agilent Technologies (2017). Agilent VEE pro. Disponible en:
<https://www.keysight.com/en/pd-1476554-pn-W4000C/vee-pro-932?nid=-34095.806312&cc=VE&lc=spa>
- Alice Team; Carnegie Mellon University (2017). Alice 3 with NetBeans. Disponible en: <http://www.alice.org/get-alice/alice-3-with-netbeans/>
- Alsobhi, A. & Abeysinghe, G. (2013a). An evaluation of accessibility of e-learning for dyslexic students. 2013 International Conference on Current Trends in Information Technology (CTIT), 1; 1-4; 4. doi:10.1109/CTIT.2013.6749468
- Alsobhi, A. Khan, N., & Rahanu, H. (2014). Toward linking dyslexia types and symptoms to the available assistive technologies. Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference On, 597-598. doi:10.1109/ICALT.2014.174
- American Psychiatric Association. (2013). Diagnostic and statistical manual of mental disorders (5th ed.). Washington DC: APA.
- Apple Inc. (2010). Introduction to quartz composer custom patch programming guide. Disponible en:
https://developer.apple.com/library/content/documentation/GraphicsImaging/Conceptual/QuartzComposer_Patch_Plugin_ProgGuide/Introduction/Introduction.html
- Boshernitsan, M & Downes, M. (2004). Visual programming languages. Visual programming languages: A survey (pp. 6-23). Berkeley, California: Computer Science Division (EECS).
- Brian, H., & Möning, J. (2017). Snap! Build your own blocks. Disponible en:
<http://snap.berkeley.edu/>
- Brooke, J. (1996). SUS-A quick and dirty usability scale. Usability Evaluation in Industry. Taylor & Francis publishers. Pag 190-213.
- Campbell, A. (2018). Cognitive complexity. an overview and evaluation. [Paper] TechDebt '18: TechDebt '18: International Conference on Technical Debt, Gothenburg, Sweden. 57-58. doi:<https://doi.org/10.1145/3194164.3194186>
- Canfield, D. (1977 and 1975). Pygmalion, a creative programming environment. Birkhauser-Verlag, Basel, and also Stanford University AI Memo 260, (CS Dept. Stan-CS-75-499,)
- Carnegie Mellon University. (2017). ALICE. An educational software that teaches students computer programming in a 3D environment. Disponible en:
<http://www.alice.org/index.php>
- Carrillo, M.; Alegría, J.; Miranda, P. & Sánchez, N. (2011). Evaluación de la dislexia en la escuela primaria: Prevalencia en español. Escritos De Psicología, 4(2), 3544.
- Chang, S. (1986). Visual languages. New York: Plenum Publishing.
- Change Dyslexia. (2016). Dytective test [computer software]. Disponible en:
<https://www.changedyslexia.org/>
- Clarisse, O. & Chang, S. (1986). VICON: A visual icon manager, Visual languages (pp. 151-39). New York: Plenum Press.

- ClaroSoftware. (2014). ScreenRuler Windows features. Disponible en:
<https://www.clarosoftware.com/portfolio/screenruler-for-pc/>
- Cycling '74. (2017). Max connects objects with virtual patch cords to create interactive sounds, graphics, and custom effects.
- De los Reyes, A.; Lewis, H.; Mendoza, R.; Neira, M.; León, J.; & Peña, O. (2008). The prevalence of reading difficulties in 7-year old children in the city of Barranquilla (Colombia), *Psicología Desde El Caribe*, 22
- Dixon, M. (2007). Comparative study of disabled vs. non-disabled evaluators in user-testing: Dyslexia and first year students learning computer programming. In C. Stephanidis (Ed.), (pp. 647-656) Springer Berlin Heidelberg.
doi:10.1007/978-3-540-73279-2_72 Disponible en:
http://dx.doi.org/10.1007/978-3-540-73279-2_72
- Everatt, J; Smythe, I; Ocampo, D & Gyarmathy, E. (2004). Issues in the assessment of literacy-related difficulties across language backgrounds: A cross-linguistic comparison. *Journal of Research in Reading*, 27, 141-151.
- Fuertes, J; González, L; & Martínez L. (2016). Characterization of programmers with dyslexia. In Springer (Ed.), *Computers helping people with special needs* (Klaus Miesenberger, Universidad de Linz ed.). Linz, Austria: Editorial Springer.
doi:10.1007/978-3-319-41267-2
- Fuse Labs (2017). Kodu game lab. Disponible en:
https://kodugamelab.fandom.com/wiki/FUSE_Labs
- Game Salad Inc. (2014). Game Salad teaches computer science by enabling students to easily and rapidly build great games. Disponible en:
<https://edu.gamesalad.com/>
- González, L. (2019a). Compendio de resultados de análisis en SPSS. Disponible en:
<http://oa.upm.es/55820/>
- González, L. (2019b). Encuesta, experimento uso de lenguajes de programación visual y textual. Disponible en: <http://oa.upm.es/55819/>
- González, L. (2019c). Estrategias para la integración de programadores con y sin dislexia. Disponible en: <http://oa.upm.es/55818/>
- Google Inc. (2016). Google for education: A library for building visual programming editors. Disponible en: <https://developers.google.com/blockly>
- Grafton, R. & Ichikawa, T. (1985). Guest editor's Introduction to the special issue on visual programming. *IEEE Comput.* 18, 8, 6-3.
- GSA Government-wide Section 508 Accessibility Program. (1998). Section.508.gov. Disponible en: <https://www.section508.gov/section-508-of-the-rehabilitation-act>
- Haber, N. (2012). La dislexia en la etapa adulta. *CINC - Centre D'Intervencio Conductal*, Col. N°. B, 421-424.
- Hammond, J. & Hercules F. (2013). In SHEFC National Coordinator (Ed.), *Understanding dyslexia an introduction for dyslexic students in higher education* Scottish Higher Education Funding Council.
- Helpes. (2016). Java [Lenguaje de programación]. Disponible en:
<https://www.java.com/es/download/help/sysreq.xml>

- IBM Corp. Released. (2015). IBM SPSS statistics for windows, version 23.0 [computer software]. Armonk, NY: IBM Corp.
- IMS Global (2016). IMS app note: Enhancing accessibility through IMS global learning standards. Disponible en: <https://www.imsglobal.org/ims-app-note-enhancing-accessibility-through-ims-global-learning-standards>
- International Dyslexia Association. (2002). Definition of dyslexia. Disponible en: <https://dyslexiaida.org/definition-of-dyslexia/>
- Jiménez, J; Guzmán, R; Rodríguez, C; & Artiles, C. (2009). Prevalencia de las dificultades específicas de aprendizaje: La dislexia en español". *Anales De Psicología*, 25(1), 78-85.
- Jonathan, E. (2017). Subtext: Uncovering the simplicity of programming. Disponible en: www.subtext-lang.org/
- Kalyvoti, K; & Mikropoulos, T. (2014). Virtual environments and dyslexia: A literature review. 5th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion, Dsai 2013, 27, 138-147. doi:10.1016/j.procs.2014.02.017
- Katusic, S; Colligan, R; Barbaresi, W; Schaid, D; & Jacobsen, S. (2001). Incidence of reading disability in a population-based birth cohort. Rochester, MN, (Mayo Clinic Proceedings), 1081.
- Ken, K. (2009). ToonTalk - making programming child's play. Disponible en: www.toontalk.com/English/toontalk.htm
- LEGO Engineering. (2017). Robolab. Disponible en: <https://robolab.unex.es/>
- Lego Mindstorms México. (2016). El lenguaje de programación NXT G 2. Disponible en: <https://nxtmexico.wordpress.com/2010/11/29/el-lenguaje-nxt-g-2/>
- Limnor Studio. (2017). Limnor studio for visual programming. Disponible en: <http://www.limnor.com/LimnorWhatIs.html>
- Lyon, G; Shaywitz, S; & Shaywitz, B. (2003). Defining dyslexia, comorbidity, teachers' knowledge of language and reading. *Annals of Dyslexia*, 53, 1-14.
- MathWorks. (2017). Simulink. simulación y diseño basado en modelos. Disponible en: <https://es.mathworks.com/products/simulink.html>
- McCabe, T. J. (1976). A complexity measures. *IEEE Trans. Softw*, 2(4), 308-320. doi:<https://doi.org/10.1109/TSE.1976.233837>
- Microsoft Research. (2004). Microsoft: Next generation of games starts with XNA. Disponible en: <https://web.archive.org/web/20080206234850/http://www.microsoft.com/presspass/press/2004/mar04/03-24XNALaunchPR.msp>
- Microsoft Student Partner. (2015). Kodu game lab. Disponible en: <https://www.kodugamelab.com/>
- MIT Media Lab. (2015). MIT app inventor. Disponible en: <http://appinventor.mit.edu/explore/>
- MIT Scratch Team. (2017). About scratch. Disponible en: <https://scratch.mit.edu/about>
- Mitkin, S; Ilyin, A; Zeeman, M; Dyadov, V. & Bachiashvili, V. (2016). Drakon. Disponible en: <http://drakon-editor.sourceforge.net/>

- Monlau, P. (1856). Diccionario etimológico de la lengua castellana (8th ed.). Madrid: Imprenta y Estereotipia de M. Rivadenetra.
- Myers, A; Arden, O; Liu, J. & Magrino, T. (2016). Fabric: Strong security, strong consistency, open and decentralized. Disponible en: www.cs.cornell.edu/projects/fabric/
- Myers, B. (1986). Visual programming, programming by example, and program visualization: A taxonomy. Proceedings SIGCHI '86: Human factors in computing systems ACM (pp. 59-7). Boston, MA.: Press.
- OMS, Organización Mundial de la Salud. (2011). Informe mundial sobre la discapacidad. Disponible en: http://www.who.int/disabilities/world_report/2011/es/
- Ousterhout, J. (1998). Scripting: Higher level programming for the 21st century. Computer, 31(3), 23-30. doi:10.1109/2.660187
- O'Brien, Justin; & Spencer, Janine. (2008). Comparison of Form and Motion Coherence Processing in Autistic Spectrum Disorders and Dyslexia. Journal of Autism and Developmental Disorders. 38. 1201 -1210.
- Pardo, N. (2015). Prevalencia del trastorno específico de la lectura en una muestra de instituciones educativas de la localidad 19 de Bogotá. Biblioteca Digital Universidad Nacional de Colombia. (05599185.2015). Disponible en: <http://www.bdigital.unal.edu.co/47004/1/05599185.2015.pdf>
- PD Community. (2015). Pure data. Disponible en <https://puredata.info/>
- Pendfriend Ltd. (2018). Pendfriend XP. Disponible en <http://www.penfriend.biz/>
- Powell, N; Moore, D; Gray, J; Finlay, J, & Reaney, J. (2004). Dyslexia and learning computer programming. SIGCSE Bulletin, 36(3), 242; 242-242. doi:10.1145/1026487.1008072
- RehaSoft. (2012). DiTres. Disponible en: <http://www.rehasoft.com/dislexia/ditres/>
- Rello, L; & Baeza-Yates, R. (2012). The presence of English and Spanish dyslexia in the web. New Review of Hypermedia and Multimedia, 18(3), 131-158. doi:10.1080/13614568.2012.721401
- Rello, L; Kanvinde, G; & Baeza-Yates, R. (2012). Layout guidelines for web text and a web service to improve accessibility for dyslexics. Paper presented at the Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, Lyon, France. 36:1-36:9. doi:10.1145/2207016.2207048 Disponible en: <http://doi.acm.org/10.1145/2207016.2207048>
- Rello, L; Williams, K; Ali, A; White, N. & Bigham, J. (2016). Dytective: Towards detecting dyslexia across languages using an online game. New York, NY 10036-9998 USA: Assoc Computing Machinery. doi:10.1145/2899475.2899491
- Salge, C. A. d. L. & Berente, N. (2016). *Pair programming vs. solo programming: What do we know after 15 years of research?* doi:10.1109/HICSS.2016.66
- Saunders, Ken. (2017). Theme Font Size Changer: Change the font size and font family in Firefox. Disponible en: <https://www.accessfirefox.org/Theme-Font-Size-Changer.php>
- Schneider, F. (2012). On concurrent programming, Springer, New York.
- Shapiro, S. & Wilk, M. (1965). An analysis of variance test for normality (complete samples). Biometrika, 52(3-4), 591-611. doi:10.1093/biomet/52.3-4.591

- Shaywitz, S. (1996). Dyslexia. *Scientific American*, 275(5), 98-104.
- Shigley, J. & Mitchell, L. (1993). *Mechanical engineering design*. New York.: McGraw-Hill.
- Smith, D. (1975). *Pygmalion: A creative programming environment* Computer Science Department, Stanford University. Disponible en: <https://books.google.com.co/books?id=mihHAAAAIAAJ>
- SolidThinking. (2017). VisSim. Disponible en: <https://web.solidthinking.com/vissim-is-now-solidthinking-embed>
- SonarQube.org. (2017). Sonarqube, the leading product for continuous code quality. Disponible en : <https://www.sonarqube.org/>
- Stencyl, L. (2017). Stencyl. Disponible en: <http://www.stencyl.com/help/toc/>
- Ingram.T (1964). "The dyslexic child", the practitioner. *192*, 503 - 516.
- TIOBE Software BV. (2019). Índice TIOBE de junio de 2019. Disponible en: <https://www.tiobe.com/tiobe-index/>
- UN, (2006). Convention on the rights of persons with disabilities. Disponible en: <http://www.un.org/disabilities/default.asp?id=150>
- W3C. (2005). Web accessibility initiative (WAI). Disponible en: <https://www.w3.org/WAI/intro/accessibility.php>
- Wagner, R. (1973). Rudolf berlin: Originator of the term dyslexia. *Annals of Dyslexia*, 1(23), 57-7.
- Wilensky, U. (1999). NetLogo.
- Wizcomtech. (2014). Reading Pents. Disponible en: <https://www.wizcomtech.com/>
- World Health Organization (Ed.). (2011). *ICD-10 international statistical classification of Diseases and Related Health Problems*. (10th ed.) Malta: WHO publications, 2011

10 Anexo A. Encuesta experimento 1.

Encuesta para la captura de datos. Experimento 1

La dislexia es un trastorno de aprendizaje donde las personas cambian o confunden las palabras o letras, y a veces dicen las palabras o frases al revés. No todas las personas con dislexia presentan los mismos síntomas, pero normalmente evidencian dificultades en la lectoescritura, mientras que en otras áreas pueden destacar positivamente. La dislexia va asociada a un coeficiente intelectual normal o por encima de la media.

Encuesta para captura de información

Agradecemos tu colaboración diligenciando la presente encuesta, la cual tiene como objetivo determinar las diferencias de desempeño en la solución de un problema dado, en dos entornos de programación diferentes, mediante el desarrollo de programas básicos en los lenguajes Alice y Java, en un tiempo determinado, estableciendo niveles de satisfacción, fortalezas y debilidades de éstos a la hora de trabajar en cada uno de los lenguajes seleccionados.

***Obligatorio**

I. Información Académica

1. Indique el nivel de conocimientos en el manejo del lenguaje de programación Java *

Marca solo un óvalo.

☐ Bajo

☐ Medio

☐ Alto

2. Indique el nivel de conocimientos en el manejo del lenguaje de programación Alice *

Marca solo un óvalo.

☐ Bajo

☐ Medio

☐ Alto

3. Indique el nivel de formación

Marca solo un óvalo.

☐ Estudiante de pregrado

☐

Título de pregrado

II. Identificación de Dislexia

4. ¿Crees tener dislexia? *

Marca solo un óvalo.

☐ Sí

☐ No

III. Desarrollo del programa

8. Cuando tiempo (minutos y segundos) se tardó en finalizar el ejercicio desarrollado bajo el lenguaje visual *

☐ Minutos

☐ Segundos

9. Cuando tiempo (minutos y segundos) se tardó en finalizar el ejercicio desarrollado bajo el lenguaje textual *

Marca solo un óvalo.

☐ Minutos

☐ Segundos

10. Usted terminó completamente el ejercicio asignado a desarrollar bajo el lenguaje visual *

Marca solo un óvalo.

☐ Si. Continúe con la pregunta 11

☐ No Continúe con la pregunta 12

11. Escriba el porcentaje de los casos de prueba realizados al programa desarrollado en el lenguaje de programación visual.

12. Usted terminó completamente el ejercicio asignado a desarrollar bajo el lenguaje textual *

Marca solo un óvalo.

☐ Si. Continúe con la pregunta 13

☐ No Continúe con la pregunta 14

13. Escriba el porcentaje de los casos de prueba realizados al programa desarrollado en el lenguaje de programación visual

VI. Pruebas de Usabilidad

14. Considera que le gustaría utilizar lenguaje de programación Alice con frecuencia.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

15. Considera que encontró que el lenguaje de programación Alice es demasiado complejo para programar.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

16. Considera que el lenguaje de programación Alice fue fácil de usar.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

17. Considera que necesita el apoyo de una persona técnica para poder utilizar el lenguaje de programación Alice

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

18. Considera que descubrió que las diversas funciones del lenguaje de programación Alice estaban bien integradas.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

19. Considera que había demasiadas inconsistencias en el uso del lenguaje de programación Alice

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

20. Se imagina que la mayoría de la gente aprendería a utilizar el lenguaje de programación Alice rápidamente.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

21. El lenguaje de programación Alice le pareció muy complicado de usar.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

22. Se sintió muy seguro usando el lenguaje programación Alice

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

23. Considera que necesitó aprender muchas cosas, antes de poderse poner en marcha con el lenguaje de programación Alice

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

24. Considera que le gustaría utilizar lenguaje de programación Java con frecuencia.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo

De acuerdo

☐ ☐ ☐ ☐ ☐

25. Considera que encontré que el lenguaje de programación Java es demasiado complejo para programar.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo

De acuerdo

☐ ☐ ☐ ☐ ☐

26. Considera que el lenguaje de programación JAVA fue fácil de usar.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo

De acuerdo

☐ ☐ ☐ ☐ ☐

27. Considera que necesita el apoyo de una persona técnica para poder utilizar el lenguaje de programación Java

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo

De acuerdo

☐ ☐ ☐ ☐ ☐

28. Considera que descubrí que las diversas funciones del lenguaje de programación Java estaban bien integradas.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo

De acuerdo

☐ ☐ ☐ ☐ ☐

29. Considera que había demasiadas inconsistencias en el uso del lenguaje de programación Java.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

30. Se imagina que la mayoría de la gente aprendería a utilizar el lenguaje de programación Java rápidamente.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

31. El lenguaje de programación Java ¿le pareció muy complicado de usar?

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

32. ¿ Se sintió muy seguro usando el lenguaje programación Java?

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

33. ¿Considera que necesitó aprender muchas cosas, antes de poderse poner en marcha con el lenguaje de programación Alice?

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

11 Anexo B. Encuesta experimento 2.

Encuesta para la captura de datos. Experimento 2

Agradecemos tu colaboración respondiendo a la presente encuesta, la cual tiene como objetivo determinar las experiencias de desempeño, y niveles satisfacción experimentados en la solución de ejercicios en2 entornos de programación diferentes, mediante el desarrollo de programas básicos en los lenguajes Alice y Java, en un tiempo determinado.

***Obligatorio**

I. Identificación

1. Indique su número de documento de identidad * _____

II. Desarrollo del programa

2. ¿Cuánto tiempo (en minutos) se tardó en finalizar el ejercicio desarrollado bajo el lenguaje visual?*

Minutos

3. ¿Cuánto tiempo (en minutos) se tardó en finalizar el ejercicio desarrollado bajo el lenguaje textual?*

Minutos

4. ¿Terminó completamente el ejercicio asignado a desarrollar bajo el lenguaje visual?*

☐ Sí.

☐ No (Continúa en la pregunta 9)

5. Escriba el porcentaje de exactitud asignado por el jurado según los casos de prueba realizados al ejercicio desarrollado en el lenguaje de programación visual.

6. ¿Cuántos errores reporta el compilador de Java luego de la transformación del código Alice a Java?

7. ¿Terminó completamente el ejercicio asignado a desarrollar bajo el lenguaje textual?*

Marque solo un óvalo.

☐ Sí.

☐ No (Continúa en la pregunta 10)

8. Escriba el porcentaje de exactitud asignado por el jurado según los casos de prueba realizados al ejercicio desarrollado en el lenguaje de programación visual

9. ¿Cuántos errores reporta el compilador de Java luego de la transformación del código Alice a Java?

_____ (Continúa en pregunta 11)

10. ¿Cuántos errores reporta el compilador de Java luego de la transformación del código Alice a Java?

III Experiencias comportamentales

Valore de 1 a 5 las siguientes afirmaciones, donde 1 significa “estoy en completo desacuerdo”, 3 “me es indiferente o no aplica” y 5 “estoy totalmente de acuerdo”.

11. Cuando programo en Java con frecuencia me siento angustiado*

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

12. Cuando programo en Java sufro de baja concentración *

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

13. Cuando programo en Java cometo muchos errores de sintaxis*

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

14. Cuando programo en Alice con frecuencia me siento angustiado*

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

15. Cuando programo en Alice sufro de baja concentración *

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

16. Cuando programo en Alice cometo muchos errores de sintaxis*

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

IV Experiencias para medir la satisfacción

17. Considera que le gustaría utilizar lenguaje de programación Alice con frecuencia.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

18. Considera que encontré que el lenguaje de programación Alice es demasiado complejo para programar.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

19. Considera que el lenguaje de programación Alice fue fácil de usar.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

20. Considera que necesita el apoyo de una persona técnica para poder utilizar el lenguaje de programación Alice

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

21. Considera que descubrió que las diversas funciones del lenguaje de programación Alice estaban bien integradas.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

22. Considera que había demasiadas inconsistencias en el uso del lenguaje de programación Alice

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

23. Se imagina que la mayoría de la gente aprendería a utilizar el lenguaje de programación Alice rápidamente.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

24. El lenguaje de programación Alice le pareció muy complicado de usar.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

25. Se sintió muy seguro usando el lenguaje programación Alice

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

26. Considera que necesitó aprender muchas cosas, antes de poderse poner en marcha con el lenguaje de programación Alice

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

27. Considera que le gustaría utilizar lenguaje de programación Java con frecuencia.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

28. Considera que encontró que el lenguaje de programación Java es demasiado complejo para programar.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

29. Considera que el lenguaje de programación JAVA fue fácil de usar.

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

30. Considera que necesita el apoyo de una persona técnica para poder utilizar el lenguaje de programación Java

Marque solo un óvalo.

	1	2	3	4	5	
Desacuerdo	☐	☐	☐	☐	☐	De acuerdo

31. Considera que descubrió que las diversas funciones del lenguaje de programación Java estaban bien integradas.

Marque solo un óvalo.

1	2	3	4	5
---	---	---	---	---

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

32. Considera que había demasiadas inconsistencias en el uso del lenguaje de programación Java.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

33. Se imagina que la mayoría de la gente aprendería a utilizar el lenguaje de programación Java rápidamente.

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

34. El lenguaje de programación Java ¿le pareció muy complicado de usar?

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

35. ¿ Se sintió muy seguro usando el lenguaje programación Java?

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

36. ¿Considera que necesitó aprender muchas cosas, antes de poderse poner en marcha con el lenguaje de programación Alice?

Marque solo un óvalo.

1 2 3 4 5

Desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo

12 Anexo C. Compendio de resultados arrojados por el software SPSS, estudio para la definición del perfil de los programadores con dislexia

Los datos arrojados por el software SPSS después de realizar los diferentes análisis se presentan a continuación.

12.1 Variables de comportamiento (Emocionales)

A continuación, en las tablas 46 a la 57, se pueden observar las frecuencias relacionadas con cada una de las variables, frente a las afirmaciones presentadas previamente.

Afirmación: Cuando programo, me considero una persona ansiosa.

Tabla 45. Comportamiento Emocional (Ansiedad_P)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	45	16.3	20	13.8	24	21.1
	2	51	18.5	35	24.1	14	12.3
	3	84	30.4	47	32.4	29	25.4
	4	67	24.3	36	24.8	29	25.4
	5	29	10.5	7	4.8	18	15.8
	Totales	276	100	145	100	114	100
SI	1	3	7.7	1	10	2	6.9
	2	2	5.1	1	10	1	3.4
	3	11	28.2	1	10	10	34.5
	4	17	43.6	6	60	11	37.9
	5	6	15.4	1	10	5	17.2
	Totales	39	100	10	100	29	100

Afirmación Cuando programo, con frecuencia me siento angustiado(a)

Tabla 46. Comportamiento Emocional (Angustia)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	55	19.9	21	14.5	32	28.1
	2	81	29.3	46	31.7	28	24.6
	3	63	22.8	34	23.4	28	24.6
	4	56	20.3	27	18.6	24	21.1
	5	21	7.6	17	11.7	2	1.8
	Totales	276	100	145	100	114	100
SI	1	5	12.8	2	20	3	10
	2	11	28.2	2	20	9	31
	3	6	15.4	2	20	4	14
	4	14	35.9	4	40	10	34
	5	3	7.7	10	100	3	10
	Totales	39	100	21	14.5	29	100

Afirmación Cuando programo, me da rabia con frecuencia.

Tabla 47. Comportamiento Emocional (Rabia)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	67	24.3	25	17.2	41	36
	2	65	23.6	35	24.1	24	21
	3	54	19.6	32	22.1	20	18
	4	71	25.7	45	31	19	17
	5	19	6.9	8	5.5	10	9
	Totales	276	100	145	100	114	100
SI	1	6	15.4	1	10	5	17
	2	13	33.3	4	40	9	31
	3	6	15.4	1	10	5	17
	4	12	30.8	4	40	8	28
	5	2	5.1	10	100	2	7
	Totales	39	100	25	17.2	29	100

Afirmación: Cuando programo sufro de baja concentración.

Tabla 48. Comportamiento Emocional (Concentración)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	100	36.2%	43	29.7%	47	41
	2	84	30.4%	56	38.6%	25	22
	3	42	15.2%	21	14.5%	20	18
	4	32	11.6%	16	11%	14	12
	5	18	6.5%	9	6.2%	8	7
	Totales	276	100%	145	100%	114	100
SI	1	11	28.2%	4	40%	7	24
	2	9	23.1%	4	40%	5	17
	3	6	15.4%	1	10%	5	17
	4	7	17.9%	1	10%	7	24
	5	6	15.4%	10	100%	5	17
	Totales	39	100%	43	29.7%	29	100

Afirmación: Cuando estudio me disperso fácilmente

Tabla 49. Comportamiento Emocional (Dispersión)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	45	16.3	11	7.6	22	19%
	2	51	18.5	32	22.1	19	17%
	3	84	30.4	29	20	30	26%
	4	67	24.3	46	31.7	29	25%
	5	29	10.5	27	18.6	14	12%
	Totales	276	100	145	100	114	100%
SI	1	3	7.7	2	20	1	3%
	2	2	5.1	2	20	4	14%
	3	11	28.2	3	30	5	17%
	4	17	43.6	3	30	13	45%
	5	6	15.4	10	100	6	21%
	Totales	39	100	11	7.6	29	100%

Afirmación: Presento con frecuencia cambios de humor

Tabla 50. Comportamiento Emocional (Humor)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia			Frecuencia
NO	1	100	36.2	54	NO	1	100
	2	67	24.3	35	22.1	2	67
	3	48	17.4	24	20	3	48
	4	43	15.6	23	31.7	4	43
	5	18	6.5	9	18.6	5	18
	Totales	276	100	145	100	Totales	276
SI	1	10	25.6	1	SI	1	10
	2	11	28.2	4	20	2	11
	3	5	12.8	1	30	3	5
	4	11	28.2	3	30	4	11
	5	2	5.1	1	100	5	2
	Totales	39	100	10	7.6	Totales	39

Afirmación: Sufro con frecuencia de depresión

Tabla 51. Comportamiento Emocional (Depresión)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	165	59.8	90	62.1	68	60%
	2	46	16.7	24	16.6	17	15%
	3	38	13.8	20	13.8	17	15%
	4	19	6.9	9	6.2	9	8%
	5	8	2.9	2	1.4	3	3%
	Totales	276	100	145	100	114	100%
SI	1	15	38.5	6	60	9	31%
	2	14	35.9	3	30	11	38%
	3	7	17.9	1	10	6	21%
	4	1	2.6	10	100	1	3%
	5	2	5.1	90	62.1	2	7%
	Totales	39	100	24	16.6	29	100%

Afirmación: Con frecuencia me siento triste

Tabla 52. Comportamiento Emocional (Tristeza)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	134	48.6	69	47.6	61	54%
	2	63	22.8	35	24.1	21	18.4%
	3	43	15.6	23	15.9	20	17.5%
	4	25	9.1	13	9	9	9%
	5	11	4	5	3.4	3	3%
	Totales	276	100	145	100	114	100%
SI	1	12	30.8	2	20	10	34%
	2	11	28.2	3	30	8	28%
	3	11	28.2	4	40	7	24%
	4	3	7.7	1	10	2	7%
	5	2	5.1	10	100	2	7%
	Totales	39	100	69	47.6	29	100%

Afirmación: Con frecuencia sufro de episodios de irritabilidad

Tabla 53. Comportamiento Emocional (Irritabilidad)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	106	38.4	49	33.8	49	43%
	2	70	25.4	40	27.6	25	22%
	3	53	19.2	28	19.3	24	21%
	4	34	12.3	23	15.9	10	9%
	5	13	4.7	5	3.4	6	5%
	Totales	276	100	145	100	114	100%
SI	1	9	23.1	4	40	9	31%
	2	10	25.6	2	20	6	21%
	3	10	25.6	3	30	8	28%
	4	8	20.5	1	10	5	17%
	5	2	5.1	10	100	1	3%
	Totales	39	100	49	33.8	29	100%

Afirmación: Ante ciertas circunstancias de la vida me considero indefenso(a)

Tabla 54. Comportamiento Emocional (Indefenso)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	79	28.6	39	26.9	35	31%
	2	75	27.2	41	28.3	30	26%
	3	65	23.6	34	23.4	29	25%
	4	44	15.9	23	15.9	17	15%
	5	13	4.7	8	5.5	3	5%
	Totales	276	100	145	100	114	100%
SI	1	10	25.6	1	10	9	31%
	2	9	23.1	3	30	6	21%
	3	13	33.3	4	40	9	31%
	4	6	15.4	2	20	4	14%
	5	1	2.6	10	100	1	3%
	Totales	39	100	39	26.9	29	100%

Afirmación: Poseo baja autoestima

Tabla 55. Comportamiento Emocional (Autoestima)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	121	43.8	45	31	72	63%
	2	70	25.4	43	29.7	20	18%
	3	45	16.3	32	22.1	8	7%
	4	30	10.9	22	15.2	8	7%
	5	10	3.6	3	2.1	6	5%
	Totales	276	100	145	100	114	100%
SI	1	14	35.9	2	20	12	41%
	2	10	25.6	4	40	6	21%
	3	9	23.1	2	20	7	24%
	4	4	10.3	2	20	2	7%
	5	2	5.1			2	7%
	Totales	39	100	10	100%	29	100%

Afirmación: Me considero una persona desconfiada

Tabla 56. Comportamiento Emocional (Desconfianza)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	51	18.5	27	18.6	22	19%
	2	64	23.2	38	26.2	18	16%
	3	67	24.3	32	22.1	31	27%
	4	64	23.2	33	22.8	28	25%
	5	30	10.9	15	10.3	15	13%
	Totales	276	100	145	100	114	100%
SI	1	7	17.9	2	20	5	17%
	2	7	17.9	3	30	4	14%
	3	10	25.6	1	10	9	31%
	4	12	30.8	4	40	8	28%
	5	3	7.7			3	10%
	Totales	39	100	10	100	29	100%

12.2 Variables de comportamiento (conductuales)

A continuación, en las tablas 58 a la 74, se pueden observar las frecuencias relacionadas con cada una de las variables, frente a las afirmaciones presentadas previamente.

Afirmación: En mi etapa de colegio con frecuencia recurría a diferentes excusas para no asistir a clases

Tabla 57. Comportamiento Conductual (Excusa_E)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	195	70.7	102	70.3	84	74%
	2	42	15.2	23	15.9	15	13%
	3	23	8.3	12	8.3	8	7%
	4	9	3.3	5	3.4	4	4%
	5	7	2.5	3	2.1	3	3%
	Totales	276	100	145	100	114	100%
SI	1	27	69.2	7	70	20	69%
	2	3	7.7	1	10	2	7%
	3	3	7.7			3	10%
	4	1	2.6			1	3%
	5	5	12.8	2	20	3	10%
	Totales	39	100	10	100	29	100%

Afirmación: Actualmente utilizo con frecuencia excusas para no asistir al trabajo o universidad

Tabla 58. Comportamiento Conductual (Excusa_T)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	186	67.4	90	62.1	89	78%
	2	54	19.6	31	21.4	17	15%
	3	23	8.3	16	11	6	5%
	4	8	2.9	5	3.4	2	2%
	5	5	1.8	3	2.1		
	Totales	276	100	145	100	114	100%
SI	1	27	69.2	6	60	21	72%
	2	6	15.4	2	20	4	14%
	3	4	10.3	1	10	3	10%
	4	1	2.6			1	3%
	5	1	2.6	1	10		
	Totales	39	100	10	100	29	100%

Afirmación: Con frecuencia utilizo conductas para llamar la atención

Tabla 59. Comportamiento Conductual (Atención)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	178	64.5	88	60.7	79	69%
	2	58	21	36	24.8	20	18%

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
	3	24	8.7	15	10.3	9	8%
	4	13	4.7	4	2.8	5	4%
	5	3	1.1	2	1.4	1	1%
	Totales	276	100	145	100	114	100%
SI	1	24	61.5	4	40	20	69%
	2	9	23.1	3	30	6	21%
	3	3	7.7			3	10%
	4	3	7.7	3	30		
	5						
	Totales	39	100	10	100	29	100%

Afirmación: Con frecuencia siento síntomas de hipoactividad, no tener ganas de hacer nada

Tabla 60. Comportamiento Conductual (Hipoactividad)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	68	24.6	33	22.8	33	29%
	2	55	19.9	32	22.1	19	17%
	3	75	27.2	41	28.3	32	28%
	4	55	19.9	26	17.9	23	20%
	5	23	8.3	13	9	7	6%
	Totales	276	100	145	100	114	100%
SI	1	8	20.5	2	20	6	21%
	2	7	17.9	2	20	5	17%
	3	7	17.9	1	10	6	21%
	4	10	25.6	3	30	7	24%
	5	7	17.9	2	20	5	17%
	Totales	39	100	10	100	29	100%

Afirmación: Mi hipoactividad me afecta a la hora de programar

Tabla 61. Comportamiento Conductual (Hipoactividad_P)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	12	4.3	7	4.8	3	10%
	2	15	5.4	6	4.1	8	27%
	3	12	4.3	8	5.5	2	7%
	4	26	9.4	12	8.3	12	40%
	5	12	4.3	6	4.1	5	17%
	Subtotales	77	27.9	39	26.9	30	100%
	Perdido del Sistema	199	72.1	106	73.1		
	Totales	276	100	145	100		
SI	2	1	2.6			1	3%
	3	1	2.6			1	3%
	4	5	12.8	1	10	4	13%
	5	10	25.6	4	40	6	29%
	Subtotales	17	43.6	5	50	12	40%
	Perdido del Sistema	22	56.4	5	50		
	Totales	39	100	10	100		

Afirmación: Me considero una persona agresiva

Tabla 62. Comportamiento Conductual (Agresividad)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	158	57.2	80	55.2	67	59%
	2	65	23.6	41	28.3	21	18%
	3	37	13.4	18	12.4	17	15%
	4	12	4.3	4	2.8	8	7%
	5	4	1.4	2	1.4	1	1%
	Totales	276	100	145	100	114	100%
SI	1	24	61.5	6	60	18	62%
	2	10	25.6	2	20	8	28%
	3	2	5.1			2	7%
	4	3	7.7	2	20	1	3%
	5						
	Totales	39	100	10	100	29	100%

Afirmación: Mi agresividad me afecta a la hora de programar

Tabla 63. Comportamiento Conductual (Agresividad_P)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	4	1.4	1	0.7	3	33.3%
	2	4	1.4			4	44.4%
	3	3	1.1	2	1.4	1	11.1%
	4	2	0.7	2	1.4		
	5	3	1.1	1	0.7	1	11.1%
	Subtotales	16	5.8	6	4.1	9	100%
	Perdido del Sistema	260	94.2	139	95.9		
	Totales	276	100	145	100		
SI	1						
	2	2	5.1	1	10	1	100%
	3						
	4	1	2.6	1	10		
	5						
	Subtotales	3	7.7	2	20	1	100%
	Perdido del Sistema	36	92.3	8	80		
	Totales	39	100	10	100		

Afirmación: Me considero una persona retraída

Tabla 64. Comportamiento Conductual (Retraído)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	112	40.6	49	33.8	58	51%
	2	77	27.9	49	33.8	24	21%
	3	40	14.5	22	15.2	16	14%
	4	33	12	18	12.4	10	9%
	5	14	5.1	7	4.8	6	5%
	Totales	276	100	145	100	114	100%
SI	1	18	46.2	3	30	15	52%
	2	8	20.5	6	60	2	7%
	3	5	12.8			5	17%
	4	6	15.4	1	10	5	17%
	5	2	5.1			2	7%
	Totales	39	100	10	100	29	100%

Afirmación: El ser retraído me afecta a la hora de programar

Tabla 65. Comportamiento Conductual (Retraído_P)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	23	8.3	12	8.3	9	56%
	2	8	2.9	6	4.1	1	6%
	3	7	2.5	2	1.4	4	25%
	4	8	2.9	4	2.8	2	13%
	5	1	0.4	1	0.7		
	Subtotales	47	17	25	17.2	16	100%
	Perdido del Sistema	229	83	120	82.8		
	Totales	276	100	145	100		
SI	1	2	5.1			2	29%
	2	2	5.1	1	10	1	14%
	3	1	2.6			1	14%
	4	2	5.1			2	29%
	5	1	2.6			1	14%
	Subtotales	8	20.5			7	100%
	Perdido del Sistema	31	79.5	9	90		
	Totales	39	100	10	100		

Afirmación: Me considero compulsivo

Tabla 66. Comportamiento Conductual (Compulsivo)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	101	36.6	42	29	54	47
	2	74	26.8	38	26.2	32	28%
	3	56	20.3	37	25.5	15	13%
	4	38	13.8	25	17.2	9	8%
	5	7	2.5	3	2.1	4	4%
	Totales	276	100	145	100	114	100%
SI	1	10	25.6	1	10	9	31%
	2	11	28.2	4	40	7	24%
	3	11	28.2	1	10	10	32%
	4	4	10.3	2	20	2	7%
	5	3	7.7	2	20	1	3%
	Totales	39	100	10	100	29	100%

Afirmación: El ser compulsivo me afecta a la hora de programar

Tabla 67. Comportamiento Conductual (Compulsivo_P)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	6	2.2	5	3.4	1	8%
	2	9	3.3	6	4.1	3	25%

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
	3	14	5.1	9	6.2	5	42%
	4	12	4.3	6	4.1	2	17%
	5	3	1.1	2	1.4	1	8%
	Subtotales	44	15.9	28	19.3	12	100%
	Perdido del Sistema	232	84.1	117	80.7		
	Totales	276	100	145	100		
SI	1	1	2.6			1	33%
	2	1	2.6			1	33%
	3	2	5.1	1	10	1	33%
	4	1	2.6	1	10		
	5	2	5.1	2	20		
	Subtotales	7	17.9	4	40	3	100%
	Perdido del Sistema	32	82.1	6	60		
	Totales	39	100	10	100		

Afirmación: Me gusta estar aislado de los demás

Tabla 68. Comportamiento Conductual (Aislamiento)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	100	36.2	52	35.9	44	39%
	2	59	21.4	38	26.2	18	16%
	3	53	19.2	25	17.2	23	20%
	4	53	19.2	28	19.3	21	18%
	5	11	4	2	1.4	8	7%
	Totales	276	100	145	100	114	100%
SI	1	10	25.6	3	30	7	24%
	2	8	20.5	3	30	5	17%
	3	7	17.9	3	30	4	14%
	4	12	30.8	1	10	11	38%
	5	2	5.1			2	7%
	Totales	39	100	10	100	29	100%

Afirmación: El querer estar aislado de los demás me afecta a la hora de programar

Tabla 69. Comportamiento Conductual (Aislamiento_P)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	21	7.6	9	6.2	11	38%
	2	18	6.5	9	6.2	8	28%
	3	5	1.8	2	1.4	2	7%
	4	9	3.3	5	3.4	3	10%

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
	5	11	4	5	3.4	5	17%
	Subtotales	64	23.2	30	20.7	29	100%
	Perdido del Sistema	212	76.8	115	79.3		
	Totales	276	100	145	100		
SI	1	6	15.4			6	46%
	2	2	5.1			2	15%
	3	1	2.6	1	10		
	4	4	10.3			4	31%
	5	1	2.6			1	8%
	Subtotales	14	35.9			13	100%
	Perdido del Sistema	25	64.1	9	10		
	Totales	39	100	10	100		

Afirmación: Me considero una persona hiperactiva

Tabla 70. Comportamiento Conductual (Hiperactividad)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	102	37	49	33.8	49	43%
	2	75	27.2	45	31	24	21%
	3	52	18.8	30	20.7	19	17%
	4	31	11.2	17	11.7	11	10%
	5	16	5.8	4	2.8	11	10%
	Totales	276	100	145	100	114	100%
SI	1	11	28.2	1	10	10	34%
	2	10	25.6	3	30	7	24%
	3	7	17.9	1	10	6	21%
	4	9	23.1	3	30	6	21%
	5	2	5.1	2	20		
	Totales	39	100	10	100	29	100%

Afirmación: Mi hiperactividad me afecta a la hora de programar

Tabla 71. Comportamiento Conductual (Hiperactividad_P)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	6	2.2	1	0.7	5	23%
	2	10	3.6	4	2.8	6	27%
	3	8	2.9	5	3.4	2	9%
	4	13	4.7	8	5.5	4	18%
	5	9	3.3	3	2.1	5	23%
	Subtotales	46	16.7	21	14.5	22	100%

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
	Perdido del Sistema	230	83.3	124	85.5		
	Totales	276	100	145	100		
SI	1	1	2.6			1	17%
	2	4	10.3	1	10	3	50%
	3	1	2.6	1	10		
	4	3	7.7	1	10	2	33%
	5	2	5.1	2	20		
	Subtotales	11	28.2	5	50	6	100%
	Perdido del Sistema	28	71.8	5	50		
	Totales	39	100	10	100		

Afirmación: Me considero una persona ansiosa

Tabla 72. Comportamiento Conductual (Ansiedad_G)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	76	27.5	45	31	29	25%
	2	66	23.9	38	26.2	23	20%
	3	54	19.6	28	19.3	23	20%
	4	60	21.7	28	19.3	27	24%
	5	20	7.2	6	4.1	12	11%
	Totales	276	100	145	100	114	100%
SI	1	4	10.3			4	14%
	2	11	28.2	4	40	7	24%
	3	9	23.1			9	31%
	4	12	30.8	6	60	6	21%
	5	3	7.7			3	10%
	Totales	39	100	10	100	29	100%

Afirmación: La ansiedad me causa hiperactividad

Tabla 73. Comportamiento Conductual (Ansiedad_H)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	13	4.7	6	4.1	7	18%
	2	19	6.9	7	4.8	10	26%
	3	16	5.8	7	4.8	7	18%
	4	19	6.9	10	6.9	8	21%
	5	12	4.3	4	2.8	7	18%
	Subtotales	79	28.6	34	23.4	39	100%
	Perdido del Sistema	197	71.4	111	76.6		
	Totales	276	100	145	100		

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
SI	1						
	2	2	5.1	2	20		
	3	4	10.3			4	44%
	4	4	10.3	2	20	2	22%
	5	5	12.8	2	20	3	33%
	Subtotales	15	38.5	6	60	9	100%
	Perdido del Sistema	24	61.5	4	40		
	Totales	39	100	10	100		

12.3 Variables de comportamiento (Físicas)

En las tablas 75 a la 79 se pueden observar las frecuencias relacionadas con cada una de las variables, frente a las afirmaciones presentadas previamente.

Afirmación: Con frecuencia padezco de insomnio

Tabla 74. Comportamiento Física (Insomnio)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	114	41.3	61	42.1	49	43%
	2	62	22.5	38	26.2	18	16%
	3	37	13.4	18	12.4	18	16%
	4	34	12.3	17	11.7	16	14%
	5	29	10.5	11	7.6	13	11%
	Totales	276	100	145	100	114	100%
SI	1	9	23.1	3	30	6	21%
	2	11	28.2	5	50	6	21%
	3	6	15.4	1	10	5	17%
	4	7	17.9	1	10	6	21%
	5	6	15.4			6	21%
	Totales	39	100	10	100	29	100%

Afirmación: Tengo dificultad para levantarme por las mañanas

Tabla 75. Comportamiento Física (Madrugar)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	46	16.7	19	13.1	26	23%
	2	43	15.6	24	16.6	18	16%
	3	43	15.6	22	15.2	18	16%
	4	67	24.3	43	29.7	21	18%
	5	77	27.9	37	25.5	31	27%
	Totales	276	100	145	100	114	100%
SI	1	6	15.4	2	20	4	14%
	2	10	25.6	2	20	8	28%
	3	3	7.7	1	10	2	7%
	4	8	20.5	3	30	5	17%
	5	12	30.8	2	20	10	34%
	Totales	39	100	10	100	29	100%

Afirmación: Sufro de continuos dolores de cabeza – migrañas

Tabla 76. Comportamiento Física (Migrañas)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	163	59.1	90	62.1	63	55%
	2	45	16.3	24	16.6	17	15%
	3	27	9.8	13	9	14	12%
	4	26	9.4	13	9	12	11%
	5	15	5.4	5	3.4	8	7%
	Totales	276	100	145	100	114	100%
SI	1	18	46.2	4	40	14	48%
	2	9	23.1	3	30	6	21%
	3	3	7.7	2	20	1	3%
	4	4	10.3	1	10	3	10%
	5	5	12.8			5	17%
	Totales	39	100	10	100	29	100%

Afirmación: Padezco de visión desordenada, (identifico figuras y/o letras al revés)

Tabla 77. Comportamiento Física (Visión_D)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	225	81.5	127	87.6	83	73%
	2	30	10.9	12	8.3	17	15%
	3	15	5.4	4	2.8	11	10%
	4	4	1.4	2	1.4	2	2%
	5	2	0.7			1	1%
	Totales	276	100	145	100	114	100%
SI	1	12	30.8	3	30	9	31%
	2	10	25.6	4	40	6	21%
	3	4	10.3			4	14%
	4	11	28.2	2	20	9	31%
	5	2	5.1	1	10	1	3%
	Totales	39	100	10	100	29	100%

Afirmación: Con frecuencia me siento fatigado

Tabla 78. Comportamiento Física (Fatiga)

Dislexia	Puntuación	General		España		Colombia	
		Frecuencia	%	Frecuencia	%	Frecuencia	%
NO	1	87	31.5	50	34.5	33	29%
	2	75	27.2	46	31.7	25	22%
	3	56	20.3	27	18.6	26	23%
	4	41	14.9	17	11.7	22	19%
	5	17	6.2	5	3.4	8	7%
	Totales	276	100	145	100	114	100
SI	1	5	12.8	2	20	3	10%
	2	7	17.9	2	20	5	17%
	3	9	23.1	2	20	7	24%
	4	14	35.9	4	40	10	34%
	5	4	10.3			4	14%
	Totales	39	100	10	100	29	100%

13 Anexo D. Compendio de resultados arrojados por el software SPSS, Experimento 1.

A continuación, se muestran todas las tablas arrojadas por SPSS acorde con los diferentes análisis estadísticos realizados.

13.1 Análisis descriptivo

Tabla 79. Frecuencias. Género

Género	Frecuencia	Porcentaje
Femenino	7	17,9
Masculino	32	82,1
Total	39	100,0

Tabla 80. Frecuencias. Nivel_Educación

Nivel_Educación	Frecuencia	Porcentaje
Estudiante de pregrado	30	76,9
Profesional	9	23,1
Total	39	100,0

Tabla 81. Frecuencias. Dyetective

Dyetective	Frecuencia	Porcentaje
Con Riesgo	10	25,6
Sin Riesgo	29	74,4
Total	39	100,0

Tabla 82. Frecuencias. Nivel_J

¿Cuál es tu nivel de conocimientos en el manejo del lenguaje de programación Java?	Frecuencia	Porcentaje
Alto	3	7,7
Bajo	5	12,8
Medio	31	79,5
Total	39	100,0

Tabla 83. Frecuencias. Termina_J

¿Terminaste el programa en Java, antes de finalizar el tiempo asignado para éste?	Frecuencia	Porcentaje
No	12	30,8
Si	27	69,2
Total	39	100,0

Tabla 84. Frecuencias. Conocimiento_J

¿Cuál es tu nivel de conocimientos en el manejo de Alice?	Frecuencia	Porcentaje
Alto	3	7,7
Bajo	8	20,5
Medio	28	71,8
Total	39	100,0

Tabla 85. Frecuencias. Termino_A

¿Terminaste el programa en Alice, antes de finalizar el tiempo asignado para éste?	Frecuencia	Porcentaje
No	11	28,2
Si	1	2,6
Si	27	69,2
Total	39	100,0

Tabla 86. Frecuencias. Precisión_J

Según las muestras de casos de prueba, indica el porcentaje de precisión de los resultados obtenidos en el programa Java.		Frecuencia	Porcentaje
Valido	3	2	5,1
	4	7	17,9
	5	16	41,0
	Total	25	64,1
Perdido	Sistema	14	35,9
Total		39	100,0

Tabla 87. Frecuencias. Nro_Errores_Java_1

¿Cuántos errores se evidencian en el programa Java que no terminaste?	Frecuencia	Porcentaje	
Válido	2	1	2,6
	3	2	5,1
	4	2	5,1
	5	1	2,6
	10	1	2,6
	11	1	2,6
	14	1	2,6
	15	1	2,6
	16	1	2,6
	18	1	2,6
	Total	12	30,8
Perdidos	Sistema	27	69,2
Total	39	100,0	

Tabla 88. Frecuencias. Precisión_Alice

Según las muestras de casos de prueba, indica el porcentaje de precisión de los resultados obtenidos en el programa realizado en Alice.		Frecuencia	Porcentaje
Válido	3	4	10,3
	4	4	10,3
	5	20	51,3
	Total	28	71,8
Perdidos	Sistema	11	28,2
Total		39	100,0

Tabla 89. Frecuencias. Nro_Errores_Alice_1

¿Cuántos errores se evidencian en el programa que no terminaste bajo Alice?		Frecuencia	Porcentaje
Válido	1	1	2,6
	2	2	5,1
	3	3	7,7
	4	1	2,6
	5	3	7,7
	10	1	2,6
	Total	11	28,2
Perdidos	Sistema	28	71,8
Total		39	100,0

Tabla 90. Medidas de tendencia central. Edad

Edad	N	39
	Media	26,62
	Mediana	23,00
	Desviación estándar	10,127
	Rango	36
	Mínimo	17
	Máximo	53
	N	39

Tabla 91. Medida de tendencia central. Nro_Errores_Alice

¿Cuántos errores se evidencian en el programa que no terminaste bajo Alice?	Valido	11
	Mínimo	1
	Máximo	10
	Rango	9

Tabla 92. Medidas de tendencia central. Tiempo_Desarrollo_J

¿Cuánto tiempo (en minutos), tardó desarrollando la solución en Java, para el problema planteado?	Media		78,26
	95% de intervalo de confianza para la media	Límite inferior	68,72
		Límite superior	87,80
	Mediana		73,00
	Desviación estándar		24,114
	Mínimo		30
	Máximo		120
	Rango Inter cuartil		42

Tabla 93. Medida de tendencia central. T_Desarrollo_Alice

¿Cuánto tiempo (en minutos) tardaste desarrollando la solución en Alice, para el problema planteado?	Media		74,79
	95% de intervalo de confianza para la media	Límite inferior	66,23
		Límite superior	83,35
	Mediana		72,00
	Desviación estándar		22,076
	Mínimo		37
	Máximo		115
	Rango intercuartil		27

Tabla 94. Medida de tendencia central. Satisfacción_J

TOTAL USABILIDAD JAVA	Media		27,13
	95% de intervalo de confianza para la media	Límite inferior	26,24
		Límite superior	28,01
	Mediana		27,00
	Desviación estándar		2,736
	Mínimo		21
	Máximo		33
	Rango intercuartil		2

Tabla 95. Medida de tendencia central. Satisfacción_A

TOTAL USABILIDAD ALICE	Media		25,41
	95% de intervalo de confianza para la media	Límite inferior	24,38
		Límite superior	26,45
	Mediana		26,00
	Desviación estándar		3,193
	Mínimo		19
	Máximo		32
	Rango intercuartil		5

13.2 Análisis cruzado

Tabla 96. Resultados. Angustia_J

		Resultado de la prueba DyTECTIVE		Total
		Con Riesgo	Sin Riesgo	
Cuando programo en Java con frecuencia me siento angustiado(a)	1	1	2	3
	2	4	5	9
	3	4	12	16
	4	1	4	5
	5	0	6	6
Total		10	29	39

Tabla 97. Resultados. Concentración_J

		Resultado de la prueba Dyteactive		Total
		Con Riesgo	Sin Riesgo	
Cuando programo en Java sufro de baja concentración	1	2	1	3
	2	3	3	6
	3	4	14	18
	4	0	4	4
	5	1	7	8
Total		10	29	39

Tabla 98. Resultados. Sintaxis_J

		Resultado de la prueba Dyteactive		Total
		Con Riesgo	Sin Riesgo	
Cuando programo en Java cometo muchos errores de sintaxis	1	4	3	7
	2	4	13	17
	3	2	5	7
	4	0	5	5
	5	0	3	3
Total		10	29	39

Tabla 99. Resultados. Angustia_A

		Resultado de la prueba Dyteactive		Total
		Con Riesgo	Sin Riesgo	
Al programar en Alice me senti angustiado(a)	1	1	1	2
	2	1	2	3
	3	0	6	6
	4	3	5	8
	5	5	15	20
Total		10	29	39

Tabla 100. Resultados. Concentración_A

		Resultado de la prueba Dyteactive		Total
		Con Riesgo	Sin Riesgo	
Al programar en Alice manifesté baja concentración	2	0	1	1
	3	0	6	6
	4	1	3	4
	5	9	19	28
Total		10	29	39

Tabla 101. Resultados. Sintaxis_A

		Resultado de la prueba Dyteactive		Total
		Con Riesgo	Sin Riesgo	
Cuando programo en Alice cometo muchos errores de sintaxis	2	1	1	2
	3	0	5	5
	4	2	7	9
	5	7	16	23
Total		10	29	39

14 Anexo E. Compendio de resultados arrojados por el software SPSS, Experimento 2.

14.1 Tablas cruzadas

Tabla 102. Tabla cruzada Dislexia - Angustia_J

			Angustia_J				
			1,00	2,00	3,00	4,00	5,00
Dislexia	Sin Riesgo	Recuento	0	0	9	8	9
		% dentro de Dislexia	0,0%	0,0%	34,6%	30,8%	34,6%
	Con Riesgo	Recuento	3	5	1	0	0
		% dentro de Dislexia	33,3%	55,6%	11,1%	0,0%	0,0%
Total		Recuento	3	5	10	8	9
		% dentro de Dislexia	8,6%	14,3%	28,6%	22,9%	25,7%

Tabla 103. Tabla cruzada Dislexia - Angustia_A

			Angustia A				Total
			2,00	3,00	4,00	5,00	
Dislexia	Sin Riesgo	Recuento	1	6	12	7	26
		% dentro de Dislexia	3,8%	23,1%	46,2%	26,9%	100,0%
	Con Riesgo	Recuento	0	0	4	5	9
		% dentro de Dislexia	0,0%	0,0%	44,4%	55,6%	100,0%
Total		Recuento	1	6	16	12	35
		% dentro de Dislexia	2,9%	17,1%	45,7%	34,3%	100,0%

Tabla 104. Tabla cruzada Dislexia - Concentración Java

			Concentración Java				
			1,00	2,00	3,00	4,00	5,00
Dislexia	Sin Riesgo	Recuento	0	0	13	11	2
		% dentro de Dislexia	0,0%	0,0%	50,0%	42,3%	7,7%
	Con Riesgo	Recuento	4	4	1	0	0
		% dentro de Dislexia	44,4%	44,4%	11,1%	0,0%	0,0%
Total		Recuento	4	4	14	11	2
		% dentro de Dislexia	11.4%	11.4%	40.0%	31.4%	5.7%

Tabla 105. Tabla cruzada Dislexia – Resumen Concentración Java

Dislexia	Sin Riesgo	Recuento	26
		% dentro de Dislexia	100,0%
	Con Riesgo	Recuento	9
		% dentro de Dislexia	100,0%
Total		Recuento	35
		% dentro de Dislexia	100.0%

Tabla 106. Tabla cruzada Dislexia - Concentración Alice

			Concentración Alice			Total
			3,00	4,00	5,00	
Dislexia	Sin Riesgo	Recuento	9	12	5	26
		% dentro de Dislexia	34,6%	46,2%	19,2%	100,0%
	Con Riesgo	Recuento	0	1	8	9
		% dentro de Dislexia	0,0%	11,1%	88,9%	100,0%
Total		Recuento	9	13	13	35
		% dentro de Dislexia	25.7%	37.1%	37.1%	100.0%

Tabla 107. Tabla cruzada Dislexia - Sintaxis_J

			Sintaxis Java				
			1,00	2,00	3,00	4,00	5,00
Dislexia	Sin Riesgo	Recuento	1	5	14	4	2
		% dentro de Dislexia	3,8%	19,2%	53,8%	15,4%	7,7%
	Con Riesgo	Recuento	5	2	2	0	0
		% dentro de Dislexia	55,6%	22,2%	22,2%	0,0%	0,0%
Total		Recuento	6	7	16	4	2
		% dentro de Dislexia	17,1%	20,0%	45,7%	11,4%	5,7%

Tabla 108. Tabla cruzada Dislexia - Sintaxis_A

			Sintaxis Alice			Total
			3,00	4,00	5,00	
Dislexia	Sin Riesgo	Recuento	1	5	20	26
		% dentro de Dislexia	3,8%	19,2%	76,9%	100,0%
	Con Riesgo	Recuento	0	0	9	9
		% dentro de Dislexia	0,0%	0,0%	100,0%	100,0%
Total		Recuento	1	5	29	35
		% dentro de Dislexia	2,9%	14,3%	82,9%	100,0%

14.2 Pruebas no paramétricas. Tablas cruzadas

Tabla 109. Prueba no paramétrica Angustia_J

Tabla cruzada						
			Angustia_J			
			1,00	2,00	3,00	4,00
Dislexia	Sin Riesgo	Recuento	0	0	9	8
		% dentro de Angustia_J	0,0%	0,0%	90,0%	100,0%
	Con Riesgo	Recuento	3	5	1	0
		% dentro de Angustia_J	100,0%	100,0%	10,0%	0,0%
Total		Recuento	3	5	10	8
		% dentro de Angustia_J	100,0%	100,0%	100,0%	100,0%

Tabla 110. Recuento prueba no paramétrica Angustia_J

Tabla cruzada				
			Angustia_J	Total
			5,00	
Dislexia	Sin Riesgo	Recuento	9	26
		% dentro de Angustia_J	100,0%	74,3%
	Con Riesgo	Recuento	0	9
		% dentro de Angustia_J	0,0%	25,7%
Total		Recuento	9	35
		% dentro de Angustia_J	100.0%	100.0%

Tabla 111. Pruebas de chi-cuadrado Angustia_J

	Valor	Df	Significación asintótica (bilateral)
Chi-cuadrado de Pearson	30,288 ^a	4	,000
Razón de verosimilitud	33,402	4	,000
N de casos válidos	35		

Tabla 112. Prueba no paramétrica Angustia_A

Tabla cruzada						
			Angustia_A			
			2,00	3,00	4,00	5,00
Dislexia	Sin Riesgo	Recuento	1	6	12	7
		% dentro de Angustia_A	100,0%	100,0%	75,0%	58,3%
	Con Riesgo	Recuento	0	0	4	5
		% dentro de Angustia_A	0,0%	0,0%	25,0%	41,7%
Total		Recuento	1	6	16	12
		% dentro de Angustia_A	100,0%	100,0%	100,0%	100,0%

Tabla 113. Recuento prueba no paramétrica Angustia_A

Dislexia	Sin Riesgo	Recuento	26
		% dentro de Angustia_A	74,3%
	Con Riesgo	Recuento	9
		% dentro de Angustia_A	25,7%
Total		Recuento	35
		% dentro de Angustia_A	100.0%

Tabla 114. Pruebas de chi-cuadrado Angustia_A

Pruebas de chi-cuadrado			
	Valor	Df	Significación asintótica (bilateral)
Chi-cuadrado de Pearson	4,026 ^a	3	,259
Razón de verosimilitud	5,608	3	,132
N de casos válidos	35		

Tabla 115. Prueba no paramétrica Concentracion_J

			1,00	2,00	3,00	4,00
Dislexia	Sin Riesgo	Recuento	0	0	13	11
		% dentro de Concentración Java	0,0%	0,0%	92,9%	100,0%
	Con Riesgo	Recuento	4	4	1	0
		% dentro de Concentración Java	100,0%	100,0%	7,1%	0,0%
Total		Recuento	4	4	14	11
		% dentro de Concentración Java	100,0%	100,0%	100,0%	100,0%

Tabla 116. Resumen prueba no paramétrica Concentración_J

			Concentración Java	Total
			5,00	
Dislexia	Sin Riesgo	Recuento	2	26
		% dentro de Concentración Java	100,0%	74,3%
	Con Riesgo	Recuento	0	9
		% dentro de Concentración Java	0,0%	25,7%
Total		Recuento	2	35
		% dentro de Concentración Java	100,0%	100,0%

Tabla 117. Pruebas de chi-cuadrado Concentración_J

Pruebas de chi-cuadrado			
	Valor	df	Significación asintótica (bilateral)
Chi-cuadrado de Pearson	30,139 ^a	4	,000
Razón de verosimilitud	32,698	4	,000
N de casos válidos	35		

Tabla 118. Prueba no paramétrica Concentracion_A

			Concentración Alice			Total
			3,00	4,00	5,00	
Dislexia	Sin Riesgo	Recuento	9	12	5	26
		% dentro de Concentración Alice	100,0%	92,3%	38,5%	74,3%
	Con Riesgo	Recuento	0	1	8	9
		% dentro de Concentración Alice	0,0%	7,7%	61,5%	25,7%
Total		Recuento	9	13	13	35
		% dentro de Concentración Alice	100,0%	100,0%	100,0%	100,0%

Tabla 119. Pruebas de chi-cuadrado Concentración_A

	Valor	df	Significación asintótica (bilateral)
Chi-cuadrado de Pearson	14,060a	2	,001
Razón de verosimilitud	15,529	2	,000
N de casos válidos	35		

Tabla 120. Resumen prueba no paramétrica Concentración_A

			Sintaxis Java			
			1,00	2,00	3,00	4,00
Dislexia	Sin Riesgo	Recuento	1	5	14	4
		% dentro de Sintaxis Java	16,7%	71,4%	87,5%	100,0%
	Con Riesgo	Recuento	5	2	2	0
		% dentro de Sintaxis Java	83,3%	28,6%	12,5%	0,0%
Total		Recuento	6	7	16	4
		% dentro de Sintaxis Java	100,0%	100,0%	100,0%	100,0%

Tabla 121. Pruebas de chi-cuadrado Sintaxis_J

			Sintaxis Java	Total
			5,00	
Dislexia	Sin Riesgo	Recuento	2	26
		% dentro de Sintaxis Java	100,0%	74,3%
	Con Riesgo	Recuento	0	9
		% dentro de Sintaxis Java	0,0%	25,7%
Total		Recuento	2	35
		% dentro de Sintaxis Java	100,0%	100,0%

Tabla 122. Resumen Prueba de chi-cuadrado Sintaxis_J

Varones			Mujeres	
Afección de la Dislexia	Frecuencia	Porcentaje	Frecuencia	Porcentaje
Sin Dislexia	224	87,8%	52	86,7%
No	15	5,9%	6	10,0%
Sí	16	6,3%	2	3,3%
Total	255	100%	60	100%

Resumen Pruebas de chi-cuadrado Sintaxis_J

	Valor	Df	Significación asintótica (bilateral)
Chi-cuadrado de Pearson	13,998a	4	,007
Razón de verosimilitud	14,064	4	,007
N de casos válidos	35		

Tabla 123. Prueba no paramétrica Sintaxis_A

			Sintaxis Alice			Total
			3,00	4,00	5,00	
Dislexia	Sin Riesgo	Recuento	1	5	20	26
		% dentro de Sintaxis Alice	100,0%	100,0%	69,0%	74,3%
	Con Riesgo	Recuento	0	0	9	9
		% dentro de Sintaxis Alice	0,0%	0,0%	31,0%	25,7%
Total		Recuento	1	5	29	35
		% dentro de Sintaxis Alice	100.0%	100.0%	100.0%	100.0%

Tabla 124. Pruebas de chi-cuadrado Sintaxis_A

	Valor	Df	Significación asintótica (bilateral)
Chi-cuadrado de Pearson	2,507 ^a	2	,286
Razón de verosimilitud	3,979	2	,137
N de casos válidos	35		