



TESIS DOCTORAL

MEDIE_GEDILEC: propuesta de metodología para la creación de compañeros de aprendizaje para la enseñanza de la programación en Educación Primaria.

Autor:

José Miguel Ocaña Chiluisa

Directores:

Diana Rosario Pérez Marín

María Celeste Pizarro Romero

**Programa de Doctorado en Tecnologías de la Información y las
Comunicaciones**

Escuela Internacional de Doctorado

2021



MEDIE_GEDILEC: propuesta de metodología para la creación de compañeros de aprendizaje para la enseñanza de la programación en Educación Primaria.

José Miguel Ocaña Chiluisa

Índice de contenidos

Agradecimientos	XV
Resumen	XVII
Abstract.....	XIX
Capítulo 1. Introducción	1
1.1 Motivación	1
1.2 Objetivos	3
1.3 Visión global.....	3
1.3.1 Interacción Persona Ordenador	4
1.3.1.1 Diseño centrado en el usuario.....	4
1.3.1.2 Co-diseño.....	4
1.3.2 Compañeros de aprendizaje	4
1.3.2.1 Definición	4
1.3.2.2 Integración	5
1.3.2.3 Taxonomía.....	5
1.3.2.4 Comparativa	5
1.3.2.5 Nivel educativo.....	5
1.3.3 Gestor de Diálogo.....	5
1.3.3.1 Interfaz de entrada-salida.....	5
1.3.3.2 Algoritmo	5
1.3.3.3 Análisis	5
1.3.4 Programación	5
1.3.4.1 Entrada-salida	5
1.3.4.2 Condicionales	6
1.3.4.3 Bucles	6
1.3.4.4 Metodología.....	6
1.3.4.5 Lenguaje de programación	6
1.3.4.6 Depuración	6
1.4 Escenario.....	6
1.5 Publicaciones	8
1.6 Estructura	10
Capítulo 2. Marco teórico.....	12
2.1 Constructivismo	12
2.1.1 Cognitivo.....	14

2.1.2 Socio-cultural	15
2.2 Construccionismo	17
2.2.1 Ideas principales	17
2.2.2 Aplicaciones prácticas	19
2.2.2.1 LOGO	19
2.2.2.2 Smalltalk	20
2.2.2.3 Pascal Básico	21
2.2.2.4 Scheme	21
2.3 Construccionismo social	22
Capítulo 3. Estado del arte.....	24
3.1 Enseñanza de la programación.....	24
3.1.1 Motivación	25
3.1.2 Metodología	27
3.1.2.1 MECOPROG	27
3.1.2.2 Gamificación	28
3.1.3 Tecnología.....	30
3.1.3.1 Robots.....	30
3.1.3.2 Scratch	32
3.1.3.3 Code.org	33
3.1.3.4 PRIMM.....	34
3.1.3.5 Modelo de Bloques.....	35
3.1.3.6 Lectura de código vs escritura de código	36
3.1.3.7 Usar-modificar-crear	37
3.2 Agentes conversacionales Pedagógicos.....	38
3.2.1 Definición y ejemplos	38
3.2.1.1 El insecto Herman	40
3.2.1.2 Stev	40
3.2.1.3 Guilly.....	41
3.2.1.4 Sam	41
3.2.1.5 Autotutor.....	42
3.2.1.6 Sara	43
3.2.1.7 MentorChat.....	43
3.2.1.8 Prof. Watson	44
3.2.2 Taxonomía.....	45
3.2.3 Metodología de diseño (MEDIE) y co-diseño (IPO)	47

3.2.3.1 MEDIE	47
3.2.3.1 Co-diseño.....	48
3.2.4 Compañeros de aprendizaje para el aprendizaje de programación	50
3.2.4.1 J-LATTE.....	52
3.2.4.2 CPP-TUTOR	53
3.2.4.3 ASK-ELLE	55
3.2.4.4 ANNETE	56
3.2.4.5 JEPPY	57
Capítulo 4. Propuesta metodológica	60
4.1 Fase 1: Comunicación con docentes, estudiantes y co-diseño.....	61
4.1.1 Comunicación con docentes y estudiantes	61
4.1.2 Co-diseño	63
4.2 Fase 2: Análisis de diálogos.....	65
4.2.1 Storytelling	65
4.3 Fase 3: Gestor de diálogo.....	68
4.3.1 API Text-to-Speech.....	68
4.3.2 AIML.....	69
4.4 Fase 4: Editor de código	71
4.4.1 Lenguaje de programación	71
4.4.1.1 Entrada-salida	71
4.4.1.2 Condicionales	72
4.4.1.3 Bucles	72
4.4.2 Ejemplos compiladores / intérpretes	73
4.4.2.1 Ideone	73
4.4.2.2 CodePad.....	73
4.4.2.3 Python.....	74
4.4.2.4 WritePhpOnline	74
4.4.2.5 Browxy	75
4.4.3 Proceso de ejecución	76
4.4.4 Proceso de depuración.....	76
4.5 Fase 5: Validación de la interfaz y de la funcionalidad	78
4.5.1 Validación de la interfaz	78
4.5.2 Validación de la funcionalidad.....	79
4.6 Fase 6: Sesiones prácticas	80
4.6.1 Sesiones presenciales	81

4.6.2 Sesiones virtuales	81
4.6.3 Interfaces para la integración	82
4.6.3.1 Inicio de sesión	82
4.6.3.2 Opciones generales	82
4.6.3.3 Tutoriales	83
4.6.3.4 Editor de programación	84
4.6.3.5 Encuestas de satisfacción.....	85
Capítulo 5. Validación de la metodología	87
5.1 Fase 1: Comunicación y co-diseño: docentes y estudiantes.	88
5.1.1 Comunicación con docentes.....	88
5.1.2 Co-diseño con niños	90
5.2 Análisis de diálogos	94
5.2.1 Storytelling	94
5.2.1.1 Storytelling aplicado a estudiantes de cuarto año.....	94
5.2.1.2 Storytelling, aplicado a estudiantes de quinto año	96
5.2.2 Revisión de diálogos.	97
5.3 Gestor de diálogo	99
5.3.1 API Text-to-Speech.....	99
5.3.2 Desarrollo del gestor de diálogo.....	100
5.4 Editor de código.....	103
5.4.1 Lenguaje de programación	103
5.4.1.1 Entrada-salida	104
5.4.1.2 Condicionales	105
5.4.1.3 Bucles	106
5.4.2 Intérprete del compañero de aprendizaje	106
5.4.3 Proceso de ejecución	107
5.4.3.1 Análisis léxico	107
5.4.3.2 Análisis sintáctico.....	108
5.4.3.3 Análisis semántico	110
5.4.4 Proceso de depuración.....	111
5.5 Validación de la interfaz y de la funcionalidad	114
5.5.1 Experimento 1.	114
5.5.2.1 Resultados.....	116
5.5.2 Experimento 2.	117
5.5.3.1 Ganancias de aprendizaje	125

5.5.3.2 Amenazas a la validez	136
5.6 Fase 6: Sesiones prácticas	137
5.6.1 Sesiones presenciales	138
4.6.2 Sesiones virtuales	139
5.6.3 Interfaces para la integración	140
5.6.3.1 Inicio de sesión	140
5.6.3.2 Perfil de usuario	141
5.6.3.3 Opciones generales	142
5.6.3.4 Chat del gestor diálogo	143
5.6.3.5 Tutoriales	145
5.6.3.6 Recetas	145
5.6.3.7 Medallero	147
5.6.3.8 Editor de programación	148
5.6.3.9 Encuestas de satisfacción	152
Capítulo 6. Conclusiones	154
6.1 Objetivos cumplidos	154
6.2 Contribuciones	156
6.3 Trabajo futuro	158
Referencias Web	181
Anexo	182

Índice de figuras

Figura 1.1 Visión global del marco de investigación de este trabajo (fuente: propia)	4
Figura 1.2 Representación del escenario presencial (fuente: propia)	7
Figura 1.3 Representación del escenario en línea	7
Figura 2.1 Interfaz de Mindstorms – LOGO (fuente: Papert, 1991)	20
Figura 2.2 Smalltalk (fuente: Goldberg y Kay, 1976)	21
Figura 2.3 Snap! (fuente: Resnick et al., 2009)	22
Figura 3.1 Principios básicos de la gamificación (fuente: Marache y Brangier, 2013) ..	29
Figura 3.2 Interfaz de Scratch (fuente: Resnick et al., 2009)	33
Figura 3.3 Interfaz de Code.org (fuente: Code.org, 2019)	34
Figura 3.4 Framework para el diseño de lección de programación (fuente: Sentance et al., 2019).....	35
Figura 3.5 Progresión del aprendizaje, Usar-Modificar-Crear (fuente: Lee et al., 2011)	38
Figura 3.6 Interfaz del agente Herman (fuente: Lester et al., 1997).....	40
Figura 3.7 Interfaz del agente Steve (fuente: Rickel y Johnson, 1999).....	41
Figura 3.8 Interfaz del agente Guilly (fuente: Nunes et al., 2002)	41
Figura 3.9 Interfaz del agente Sam (fuente: Ryokai, 2003).....	42
Figura 3.10 Interfaz del agente Autotutor (fuente: Landauer et al., 1998).....	42
Figura 3.11 Interfaz de MonetorChat (fuente: Tegos et al., 2015).....	44
Figura 3.12 Conversación entre un estudiante y el profesor Watson (fuente: Yeves-Martínez y Pérez-Marín, 2019)	45
Figura 3.13 MEDIE ciclo iterativo e incremental (fuente: Tamayo-Moreno y Pérez-Marín 2017).....	47
Figura 3.14 Proceso iterativo del (DCU), según ISO 1340	49
Figura 3.15 Interfaz de J-LATTE (fuente: Holland et al., 2009)	53
Figura 3.16 Interfaz de CPP-Tutor (fuente: Abu-Naser, 2009).....	55
Figura 3.17 Interfaz de Ask-Elle (fuente: Gardes, 2016)	56
Figura 3.18 Interfaz de Annete (fuente: Day et al., 2019).....	57
Figura 3.19 Interfaz de JEPPY (fuente: Pérez et al., 2020).....	58
Figura 4.1 MEDIE_GEDILEC ciclo iterativo e incremental (fuente: propia)	60
Figura 4.2 Fases de MEDIE_GEDILEC y cómo extiende de MEDIE (fuente: propia) .	61
Figura 4.3 Relación Diseñador-Usuario (fuente: Escobar, 2010)	62
Figura 4.4 Prototipo en papel (fuente: Fitzgerald, 2013)	63
Figura 4.5 Ejemplo de Storytelling (fuente: Liang et al., 2017).....	66
Figura 4.6 Funcionamiento de la API Text-to-Speech (fuente: Google Cloud, 2020)....	69

Figura 4.7 Funcionamiento de un gestor de diálogo con AIML (fuente: Crespo y Cabrera, 2020).....	69
Figura 4.8 Formato AIML (fuente: Crespo y Cabrera, 2020)	70
Figura 4.9 Esquema de funcionalidad del Gestor de diálogo (fuente: propia)	71
Figura 4.10 Ejemplo de entrada-salida (fuente: Mathieu, 2014).....	72
Figura 4.11 Ejemplo de condicionales (fuente: Mathieu, 2014)	72
Figura 4.12 Ejemplo de bucles (fuente: Mathieu, 2014)	72
Figura 4.13 Interfaz de Ideone (fuente: HTTP1).....	73
Figura 4.14 Interfaz de CodePad (fuente: HTTP2)	74
Figura 4.15 Interfaz del intérprete Python (fuente: HTTP3).....	74
Figura 4.16 Interfaz de WritePhpOnline (fuente: HTTP4).....	75
Figura 4.17 Interfaz de Browxy (fuente: HTTP5).....	75
Figura 4.18 Proceso de ejecución del código (fuente: propia)	76
Figura 4.19 Proceso de depuración (fuente: propia).....	77
Figura 4.20 Mensajes de depuración de Browxy (fuente: HTTP5).....	77
Figura 4.21 Adaptación en la interfaz de requisitos de usuarios y co-diseño (fuente: propia).....	78
Figura 4.22 Entorno de simulación, para la validación de funcionalidad (fuente: propia)	79
Figura 4.23 Parámetros de sesiones prácticas (fuente: propia)	81
Figura 4.24 Ingreso de usuarios (fuente: propia).....	82
Figura 4.25 Interfaz general del compañero de aprendizaje (fuente: propia).....	83
Figura 4.26 Tutoriales de aprendizaje (fuente: propia)	83
Figura 4.27 Pasos principales de la Gamificación (fuente: propia).....	84
Figura 4.28 Cuestionarios (fuente: propia).....	85
Figura 4.29 Encuestas de satisfacción (fuente: propia)	86
Figura 5.1 Aplicación de MEDIE_GEDILEC en compañero de aprendizaje Alcody (fuente: propia)	87
Figura 5.2 Sistema interactivo educativo del compañero de aprendizaje Alcody (fuente: propia).....	88
Figura 5.3 Diseño de Alcody en base a las primeras entrevistas con docentes (fuente: Morales et al., 2019).....	88
Figura 5.4 Encuesta en línea para el co-diseño con niños (fuente: propia).....	91
Figura 5.5 Encuesta en papel para el co-diseño con niños (fuente: propia)	91
Figura 5.6 Compañero de aprendizaje, Alcody (Morales et al., 2019).....	92
Figura 5.7 Interfaz general de Alcody (fuente: propia)	93
Figura 5.8 Stortelling, diálogo con Alcody, estudiante cuarto año (Ocaña et al., 2019).	95

Figura 5.9 Stortelling, diálogo con Alcody, estudiante quinto año (Ocaña et al., 2019)	96
Figura 5.10 Adaptación de prototipos a la interfaz del gestor de diálogo (fuente: propia)	99
Figura 5.11 Diseño de la interfaz del gestor de diálogo en base a prototipos (fuente: propia)	100
Figura 5.12 Funcionamiento del gestor de diálogo de Alcody (fuente: propia)	102
Figura 5.13 Esquema de funcionalidad del Gestor de diálogo (AlcodyGDiag) (fuente: propia)	102
Figura 5.14 Editor de código de Alcody (fuente: propia)	103
Figura 5.15 Ejemplo de resolución del Ejercicio 05 (fuente: propia)	105
Figura 5.16 Ejemplo de resolución del Ejercicio 10 (fuente: propia)	105
Figura 5.17 Ejemplo de resolución del Ejercicio 15 (fuente: propia)	106
Figura 5.18 Esquema de ejecución del código fuente (AlcodyGEjec) (fuente: propia)	107
Figura 5.19 Esquema de funcionamiento del Análisis léxico en Alcody (fuente: propia)	108
Figura 5.20 Ejemplo de árbol de sintaxis (fuente: propia)	109
Figura 5.21 Esquema de funcionamiento del Análisis sintáctico en Alcody (fuente: propia)	110
Figura 5.22 Esquema de funcionamiento del Análisis semántico en Alcody (fuente: propia)	111
Figura 5.23 Esquema de ejecución y depuración (AlcodyGProg) (fuente: propia)	113
Figura 5.24 Preguntas analizadas en cada prueba (Ocaña et al., 2020)	121
Figura 5.25 Pre-test y post-test diagramas de caja para todos los conceptos (Ocaña et al., 2020)	125
Figura 5.26 Diagramas de caja en la pre-test y post-test en el concepto de condicionales se enseña en las sesiones en línea durante la pandemia (Ocaña et al., 2020)	127
Figura 5.27 Los estudiantes acceden al sistema interactivo educativo, Alcody (fuente: propia)	138
Figura 5.28 El docente supervisa el proceso de aprendizaje (fuente: propia)	139
Figura 5.29 El docente presenta el tema de clase (fuente: propia)	139
Figura 5.30 Resolución de ejercicios propuestos (fuente: propia)	140
Figura 5.31 Ingreso de usuarios en Alcody (fuente: propia)	141
Figura 5.32 Editar perfil de usuario en Alcody (fuente: propia)	142
Figura 5.33 Interfaz de opciones generales del sistema interactivo educativo, Alcody (fuente: propia)	143
Figura 5.34 Chat del gestor de diálogo del sistema interactivo educativo, Alcody (fuente: propia)	144
Figura 5.35 Invocación a tutoriales en el gestor de diálogo (fuente: propia)	144

Figura 5.36 Tutoriales de aprendizaje de programación (fuente: propia)	145
Figura 5.37 Gamificación de MECOPROG, Alcodey (fuente: propia)	146
Figura 5.38 Interfaz de las recetas en Alcodey (fuente: propia)	147
Figura 5.39 Medallero de actividades en Alcodey (fuente: propia)	148
Figura 5.40 Editor de código en Alcodey (fuente: propia)	149
Figura 5.41 Proceso de ejecución en Alcodey (fuente: propia)	149
Figura 5.42 Mensajes predictivos del depurador de Alcodey (fuente: propia)	150
Figura 5.43 Mensajes del depurador de Alcodey (fuente: propia)	150
Figura 5.44 Listado cuestionarios en Alcodey (fuente: propia)	151
Figura 5.45 Resolución y calificación de preguntas (fuente: propia)	152
Figura 5.46 Ejemplo de encuesta en Alcodey (fuente: www.alcodey.site/resultados)	153
Figura 6.1 Contribuciones del trabajo de investigación (fuente: propia)	156

Índice de tablas

Tabla 1.1 Fases de la metodología MEDIE_GEDILEC por publicaciones (fuente: propia)	10
Tabla 3.1 Ejemplos de metáforas (fuente: Pérez-Marín et al., 2020).....	28
Tabla 3.2 Unidades curriculares (fuente: Witherspoon et al., 2017).....	31
Tabla 3.3 Modelo por Bloques (fuente: Schulte, 2008)	36
Tabla 3.4. Compatibilidad entre estrategias de instrucción y tácticas de instrucción (fuente: Sentanc et al., 2019).....	37
Tabla 3.5 Características (fuente: Crow et al., 2018)	59
Tabla 4.1 Esquema de análisis del diálogo (fuente: Tegos, 2015)	67
Tabla 4.2 Parámetros para analizar los diálogos agente-alumno.....	68
Tabla 5.1 Características y requisitos en escuela de Educación Primaria – España (fuente: propia).....	89
Tabla 5.2 Características y requisitos en escuela de Educación Primaria – Ecuador (fuente: propia)	90
Tabla 5.3 Rasgos de características del compañero de aprendizaje (Morales et al., 2019)	92
Tabla 5.4 Parámetros del análisis de diálogos (Ocaña et al., 2019)	94
Tabla 5.5 Parámetros de análisis de los diálogos Alcody – estudiante (fuente: propia) .	97
Tabla 5.6 Directrices para los parámetros del análisis de diálogos (fuente: propia).....	98
Tabla 5.7 Ejemplo de intenciones (fuente: propia).....	100
Tabla 5.8 Ejemplo de entidades, para la intención –Saludar- (fuente: propia)	101
Tabla 5.9 Ejemplo de respuestas de Alcody.....	101
Tabla 5.10 Instrucciones de pseudocódigo (fuente: propia).....	103
Tabla 5.11 Tabla de símbolos (fuente: propia).....	104
Tabla 5.12 Listado de instrucciones de la gramática (fuente: propia).....	107
Tabla 5.13 Listado de sentencias ejecutables del intérprete (fuente: propia)	110
Tabla 5.14 Listado colores para instrucciones de pseudocódigo (fuente: propia).....	112
Tabla 5.15 Ejemplos de escritura errónea de instrucciones (fuente: propia).....	112
Tabla 5.16 Ejemplos de omisiones a la gramática formal (fuente: propia).....	113
Tabla 5.17 Rúbrica para el test de programación (Morales et al., 2020).....	115
Tabla 5.18 Parámetros estadísticos del experimento (Morales et al., 2020)	116
Tabla 5.19 Tiempo de enseñanza con Alcody (Ocaña et al., 2020)	119
Tabla 5.20 Resumen de variables, tipo (dv- variable dependiente, iv - variable independiente), nombres y descripción (Ocaña et al., 2020).....	119
Tabla 5.21 Rúbrica para la prueba (Ocaña et al., 2020)	122

Tabla 5.22 Cuestionario de pandemia pre-covid-19 (Ocaña et al., 2020).....	123
Tabla 5.23 Cuestionario durante la pandemia covid-19 (Ocaña et al., 2020)	123
Tabla 5.24 Técnicas de recopilación de datos (Ocaña et al., 2020).	124
Tabla 5.25 Análisis descriptivo simple (Ocaña et al., 2020).....	125
Tabla 5.26 Anova unidireccional (Ocaña et al., 2020).....	126
Tabla 5.27 Análisis descriptivo de la muestra para el concepto de condicionales, durante la pandemia (Ocaña et al., 2020)	127
Tabla 5.28 Resultados de la prueba t para muestras pareadas entre pre-test y post-test en concepto condicional (Ocaña et al., 2020)	127
Tabla 5.29 Análisis de los resultados del cuestionario pre-covid (Ocaña et al., 2020)	129
Tabla 5.30 Análisis de los resultados del cuestionario durante Covid-19 (Ocaña et al., 2020).....	129
Tabla 5.31 Comparación de resultados antes y durante Covid-19 (Ocaña et al., 2020)	130
Tabla 6.1 Características de compañeros de aprendizaje (fuente: adaptación Crow et al., 2018).....	157

Doctora Doña Diana Pérez Marín, Profesora Titular Doctora del Departamento de Ciencias de la Computación, Arquitectura de Computadores, Lenguajes y Sistemas Informáticos, Estadística e Investigación Operativa de la Universidad Rey Juan Carlos, y la Doctora Doña María Celeste Pizarro Romero, Profesora Titular Doctora del Departamento de Matemática Aplicada, Ciencia e Ingeniería de los Materiales y Tecnología Electrónica, Estadística e Investigación Operativa de la Universidad Rey Juan Carlos.

Certifican:

Que el presente trabajo de investigación titulado: “MEDIE_GEDILEC: propuesta de metodología para la creación de compañeros de aprendizaje para la enseñanza de la programación en Educación Primaria” constituye la memoria que presenta D. José Miguel Ocaña Chiluisa para optar al grado de Doctor en Tecnologías de la Información y las Comunicaciones, bajo su dirección.

Y para que así conste, firman el presente certificado en Móstoles a 02 de junio de 2021.

Fdo: Diana Rosario Pérez Marín

Fdo: María Celeste Pizarro Romero

A mis padres y a mis hijas.

Agradecimientos

Ingrid Bergman decía "El éxito es conseguir lo que quieres, la felicidad es querer lo que consigues".

Este trabajo de investigación ha sido una experiencia enriquecedora en muchos sentidos, no solo en la parte profesional y académica, sino la considero gratificante en mi superación personal.

Durante este tiempo he contado con personas que han colaborado de una u otra manera para la consecución del objetivo, en este sentido, quiero aprovechar y dar las gracias a todas ellas.

De manera inicial quiero resaltar mi agradecimiento a mi directora de tesis, Diana Pérez, quien desde el primer día mostró un compromiso total con el proyecto de investigación, su orientación y apoyo ha sido fundamental para lograr esta meta académica.

De igual manera a mi codirectora de tesis, Celeste Pizarro, quien además de fortalecer el proceso investigativo, dio una orientación clara y precisa en la parte estadística del experimento.

Debo destacar en Diana y Celeste, su parte humana y agradecer profundamente por concederme el compartir conmigo sus conocimientos, experiencia y en especial su tiempo.

Un agradecimiento a Silvia Tamayo, quien colaboró al inicio del proyecto y me facilitó la información de la metodología MEDIE de la cual se deriva el presente trabajo de investigación.

A Elizabeth, mi esposa, por acompañarme durante este proceso que nos permite dejar un ejemplo de esfuerzo y constancia a nuestras hijas.

A María Sol y Victoria Valentina, mis hijas, por su comprensión en el tiempo que el proyecto de investigación ha durado y que no me ha permitido compartir muchas cosas con ellas.

A mis padres, por ser mi ejemplo de trabajo diario y apoyo constante, sin su formación en valores no hubiera logrado superar este reto profesional, agradecer de igual manera su comprensión por haberme perdido de muchos momentos especiales en familia.

A mis hermanos y sobrinos por colaborarme en los momentos que no me fue posible estar con mis padres y velar por su bienestar.

A mis suegros Gilberto y Katy, a mis cuñadas por su apoyo incondicional y amor hacia mis hijas, gracias por compartir cada tiempo libre con ellas y llenar en gran parte el vacío que por este tiempo en varias ocasiones no lo he podido estar a su lado.

A Freepik Company, plataforma Web en la el proyecto Alcodey desde su inicio cuenta con un licenciamiento Premium, mismo que ha permitido adaptar sus imágenes en el diseño de ilustraciones del presente trabajo de investigación sin fines de lucro.

Al Ejército ecuatoriano, por las facilidades brindadas en la flexibilidad de mi horario de trabajo que me ha permitido dar el tiempo de dedicación necesario al proyecto de investigación.

A la Universidad Rey Juan Carlos de España, por los conocimientos adquiridos durante el Programa de Doctorado en Tecnologías de la Información y las Comunicaciones. Consultar si puedo colocar el financiamiento para las publicaciones.

Finalmente agradecer a las autoridades, docentes y especialmente a los niños que colaboraron en el experimento de Alcodey en las escuelas de Educación Primaria de España y Ecuador.

Resumen

En los últimos años el interés en estudiar cómo enseñar programación en Educación Primaria ha ido en aumento. Se están investigando múltiples enfoques usando Scratch, robots, lenguajes textuales, o ejercicios sin tecnología para que el mayor número de estudiantes pueda aprender a programar en edades tempranas.

Los compañeros de aprendizaje son sistemas interactivos con énfasis en el rol de acompañamiento al estudiante. El objetivo no es tanto enseñar o aprender de él sino acompañar su aprendizaje. El uso de compañeros de aprendizaje en varios dominios ha sido positivo. Sin embargo, en la revisión del estado del arte, se ha detectado una carencia en la falta de compañeros diseñados para la enseñanza de la programación a niños y la falta de una metodología que oriente su diseño, desarrollo e integración en el aula.

En esta tesis, se contribuye principalmente al campo de Interacción Persona-Ordenador con la metodología MEDIE_GEDILEC para crear compañeros de aprendizaje para enseñar programación a niños en Educación Primaria. Esta metodología extiende la metodología MEDIE para crear Agentes Conversacionales Pedagógicos para la enseñanza en distintos niveles educativos, basada a su vez en el Diseño Centrado en el Usuario.

Dada la complejidad para diseñar compañeros de aprendizaje, MEDIE_GEDILEC se centra en cómo diseñar el diálogo agente-estudiante para la enseñanza de programación tanto en modalidad presencial como en línea, y se puede complementar con otra metodología, MEDIE_LECOE, para integrar también aspectos emocionales y recomendaciones.

MEDIE_GEDILEC consta de seis fases: 1) comunicación y co-diseño con docentes y estudiantes; 2) análisis de diálogos; 3) gestor de diálogo; 4) editor de código; 5) validación de la interfaz y de la funcionalidad; y, 6) sesiones prácticas. Esta metodología proporciona el algoritmo AlcodyGDiag, que orienta el desarrollo del gestor de diálogo, y los algoritmos AlcodyGEjec y AlcodyGProg que guían el proceso de edición, ejecución, y depuración de programas escritos en pseudocódigo.

La metodología MEDIE_GEDILEC se ha validado con la implementación de Alcody, un nuevo compañero de aprendizaje para enseñar programación a niños cuyo nombre viene de ALgoritmos+CODIgo. Inicialmente, se pidió a 66 estudiantes de 8 a 10 años de edad y a su profesor (fase 1 de comunicación y co-diseño con docentes y estudiantes), de una escuela de Educación Primaria en España que escribieran un diálogo entre ellos y el ordenador. Tenía como objetivo describir cómo les gustaría que se les enseñara la programación para poder analizar cómo serían los diálogos compañero-estudiante (fase 2). Con esta información se pudo crear el gestor de diálogo (fase 3) y el editor de código en pseudocódigo (fase 4).

El entorno creado se probó en una escuela de Educación Primaria en Ecuador con 137 estudiantes de entre 10 y 12 años que usaron el compañero de aprendizaje en la asignatura de Programación Inicial durante diez meses en el curso 2019/2020 validando la interfaz y la funcionalidad (fase 5). En las sesiones con Alcodey tanto en modalidad presencial como en línea, los estudiantes pudieron aumentar significativamente sus puntuaciones de aprendizaje: de una puntuación media en un pre-test de programación de 0,88 en una escala de 0 (mínimo conocimiento) a 10 (máximo conocimiento) en septiembre de 2019, a una puntuación media de 7,56 en un primer post-test realizado en febrero de 2020 y alcanzando una puntuación media de 8,01 en el segundo post-test durante la pandemia de COVID-19 en junio de 2020 (todos con la misma escala). Estos cuestionarios cubrían los conceptos de entrada/salida, condicionales y bucles. Los resultados se complementan con los obtenidos en cuestionarios sobre la satisfacción y motivación de los estudiantes, en los que se registraron valores superiores al 90% tanto durante el tiempo previo como durante la pandemia en las siguientes sesiones realizadas (fase 6).

El trabajo de esta tesis se ha publicado en 9 artículos: 2 de conferencia, 2 capítulos de libro, y 5 en revista, uno de ellos titulado “Can a learning companion be used to continue teaching programming to children even during the COVID-19 pandemic” JCR Q1 del año 2020.

Abstract

Interest in studying how to teach programming in Primary Education has increased in recent years. Technological approaches – using Scratch, robots and textual language – and non-technological approaches have been researched with the purpose of teaching the greatest number of students how to program from an early age.

Learning companions are interactive systems that accompany students while they learn. Their objective is not so much to teach or for students to learn from them, but to accompany students while learning. The use of learning companions in different areas has been positive. Despite this, we reviewed the current situation of this area and found that there is a lack of companions specifically designed to teach programming to children as well as a lack of a methodology to guide their design, development and integration in the classroom.

The results of this thesis mainly contribute to the field of Human-Computer Interaction using the MEDIE_GEDILEC methodology to create learning companions to teach programming to children in Primary Education. This methodology furthers the MEDIE methodology to create Pedagogical Conversational Agents that can teach at different educational levels, based on User Centred Designs.

Given the complexity of designing learning companions, MEDIE_GEDILEC focuses on how to design the agent-student dialogue for teaching programming in both face-to-face and online modes, and can be complemented with another methodology, MEDIE_LECOE, to further integrate emotional aspects and recommendations.

MEDIE_GEDILEC consists of six stages: 1) communication and co-design with teachers and students; 2) dialogue analysis; 3) dialogue management; 4) code editor; 5) interface and functionality validation; and, 6) practical sessions. This methodology provides the AlcodeyGDiag algorithm, which guides the development of the dialogue manager, and the AlcodeyGEjec and AlcodeyGProg algorithms that guide the editing, executing, and debugging processes of the programs written in pseudocode.

The MEDIE_GEDILEC methodology has been validated with the implementation of Alcodey – a portmanteau of ALgorithms+CODe –, a new learning companion that teaches programming to children. Sixty-six students between the ages of 8 and 10 and their teacher were asked to write a dialogue between themselves and the computer (stage 1 of communication and co-design with teachers and students); the sample was from a Primary Education school in Spain. The purpose was to describe how they would like to be taught programming in order to analyse what the companion-student dialogues would be like

(stage 2). We used this information to create the dialogue manager (stage 3) and the pseudocode editor (stage 4).

The created environment was tested in a Primary Education school in Ecuador with 137 students aged between 10 and 12 who used the learning companion in an Initial Programming course that lasted ten months during the 2019/2020 academic year, validating the interface and functionality (phase 5). Students who learned with Alcodey in both face-to-face and online modes, significantly increased their learning scores: from a mean score on a programming pre-test of 0,88 on a scale of 0 (minimum knowledge) to 10 (maximum knowledge) in September 2019, to a mean score of 7,56 in a first post-test carried out in February 2020 and reaching an average score of 8,01 in the second post-test during the COVID-19 pandemic in June 2020 (measured using the same scale). These quizzes covered the concepts of input/output, conditionals, and loops. The results were complemented with those obtained in a questionnaire completed by the sample regarding their satisfaction and motivation, in which values higher than 90% were given by students; this was for both the pre- and post-pandemic learning (stage 6).

The work of this thesis has been published in nine articles: two conference articles, two book chapters, and five journal articles, one of them entitled “Can a learning companion be used to continue teaching programming to children even during the COVID-19 pandemic” JCR Q1, 2020

Capítulo 1. Introducción

1.1 Motivación

La informática en la Educación Primaria en los últimos años ha tomado una gran importancia dentro de la sociedad a nivel mundial. Pese a esto la cantidad de estudios específicos en el área de programación aún resultan limitados.

Hemos sido testigos de una discusión activa en torno al papel de la programación y la informática para todos, donde varios países la han implementado como asignatura dentro del currículo de su Educación Primaria.

En este sentido, países como: Australia, Inglaterra, Estonia, Finlandia, Suecia y Corea del Sur, Polonia incluyen a la programación en su currículo de Educación Primaria de manera obligatoria. Por otra parte, países como: Nueva Zelanda, Noruega y Estados Unidos incluyen a la programación en su currículo de Educación Secundaria de manera opcional (Heintz et al., 2016).

La programación puede definirse como un medio de comunicación con una máquina usando un lenguaje que "entiende", y realice algo para resolver algún problema (Papert, 1980).

Un motivo por el cual la enseñanza de programación resulta compleja para su investigación es que algunos lenguajes de programación son muy técnicos en su sintaxis y carecen de poder expresivo.

El esfuerzo de programar es recompensado, en razón que los estudiantes se vuelven activos constructores de sus propios programas y les ayuda a mejorar su comprensión del mundo digital (Papert, 1980; Pérez-Marín et al., 2020).

Para el estudio de programación no es necesario que los estudiantes en un futuro quieran convertirse en programadores, pero con el avance y los requisitos digitales actuales y futuros, pueden aprovechar el pensar como un ingeniero informático en sus vidas.

Dada la dificultad de enseñar programación, se han probado varios enfoques como: entorno visual basado en bloques con un lenguaje multimedia como Scratch, programar robots como Lego Mindstorms, o crear programas utilizando programación textual.

Por otra parte, en las últimas décadas se han realizado importantes investigaciones del uso de compañeros de aprendizaje. A éstos se los puede definir como personajes digitales animados o no que funcionan para simular la interacción humana-similar, que podría

brindar una oportunidad para simular dicha interacción social en el aprendizaje basado en computadora. (Kim y Baylor, 2006).

En este sentido, la presente propuesta plantea incorporar un compañero de aprendizaje en un sistema interactivo educativo co-diseñado con los niños para seguir aprendiendo a programar.

Sin embargo, en el estado del arte actual no se ha encontrado una metodología para crear compañeros de aprendizaje para enseñar a programar, y son pocos compañeros de aprendizaje creados sin seguir una metodología como por ejemplo Jeppy (Pérez et al., 2020).

Por tanto, en este trabajo se quiere contribuir al campo de Interacción Persona-Ordenador con una metodología para el diseño, desarrollo e integración en el aula de compañeros de aprendizaje para la enseñanza de programación en Educación Primaria. Esta metodología se valida con el diseño y desarrollo de un nuevo sistema con el compañero de aprendizaje Alcody.

Alcody es el resultado de un co-diseño con 66 niños, en su desarrollo se ha aplicado la Metodología (MEDIE_GEDILEC), con la premisa de contar con un entorno motivador y agradable, necesario para tener una experiencia de aprendizaje fructífera.

Dado que el habla constituye la forma más natural de comunicación entre las personas, en el compañero de aprendizaje se integró un gestor de diálogo para aumentar la efectividad y facilidad de aprendizaje.

Es por ello que en la presente investigación se pretende establecer las pautas a seguir en la gestión del diálogo de Alcody para la enseñanza de programación, basado en las características de diálogo en español, establecidas por los niños participantes.

Por otra parte, la gamificación es una técnica de enseñanza que introduce contenido de forma lúdica al proceso educativo, permitiendo de esta manera captar la atención del estudiante y mejorando los niveles de aprendizaje (Deterding et al., 2011).

La gamificación es una técnica de aprendizaje que traslada la mecánica de los juegos al ámbito educativo-profesional con el fin de conseguir mejores resultados, ya sea para absorber mejor algunos conocimientos, mejorar alguna habilidad, o bien recompensar acciones concretas, entre otros muchos objetivos.

Alcody también sigue un enfoque de gamificación, con actividades y medallas como recompensas a desafíos planteados.

Además, Alcody sigue la teoría del construccionismo de Papert (1980), la forma de enseñar, en este caso la programación, es "crear pensando". Con una idea central de que los niños necesitan pensar para poder crear programas con un objetivo determinado.

Los resultados de un experimento realizado en el curso 2019-2020 con 137 estudiantes de entre 10 y 12 años en Ecuador, registran una mejora significativa en las puntuaciones de los estudiantes junto con una alta satisfacción incluso durante la pandemia de COVID-19 cuando usaron Alcodey para continuar aprendiendo a programar, validando de esta forma la metodología MEDIE_GEDILEC para el diseño y desarrollo del compañero de aprendizaje Alcodey.

1.2 Objetivos

Este trabajo describe una propuesta de metodología de diseño, integración y evaluación de un compañero de aprendizaje (MEDIE_GEDILEC), que permita guiar a docentes o investigadores sugiriendo pasos para su diseño, desarrollo de herramientas de soporte, que mejoren el aprendizaje y motivación de niños en programación, sus objetivos son:

- Diseñar una metodología para la gestión del diálogo de los compañeros de aprendizaje que enseñan programación en Educación Primaria.
- Proponer un algoritmo que guíe la interacción y el diálogo para la enseñanza de conceptos de programación inicial.
- Proponer algoritmos que permitan apoyar a la metodología propuesta, permitiendo la edición, ejecución y depuración del pseudocódigo.
- Contribuir al área de la enseñanza de programación en Educación Primaria.
- Incrementar los niveles de aprendizaje, satisfacción y motivación en los niños en el momento de aprender a programar.

1.3 Visión global

La Figura 1.1 ofrece una visión general del enfoque propuesto. El objetivo es caracterizar el contexto del presente trabajo, y las diferentes disciplinas involucradas, acentuando cuatro áreas principales de investigación: Interacción Persona Ordenador, Compañeros de aprendizaje, Gestor de Diálogo y Programación.

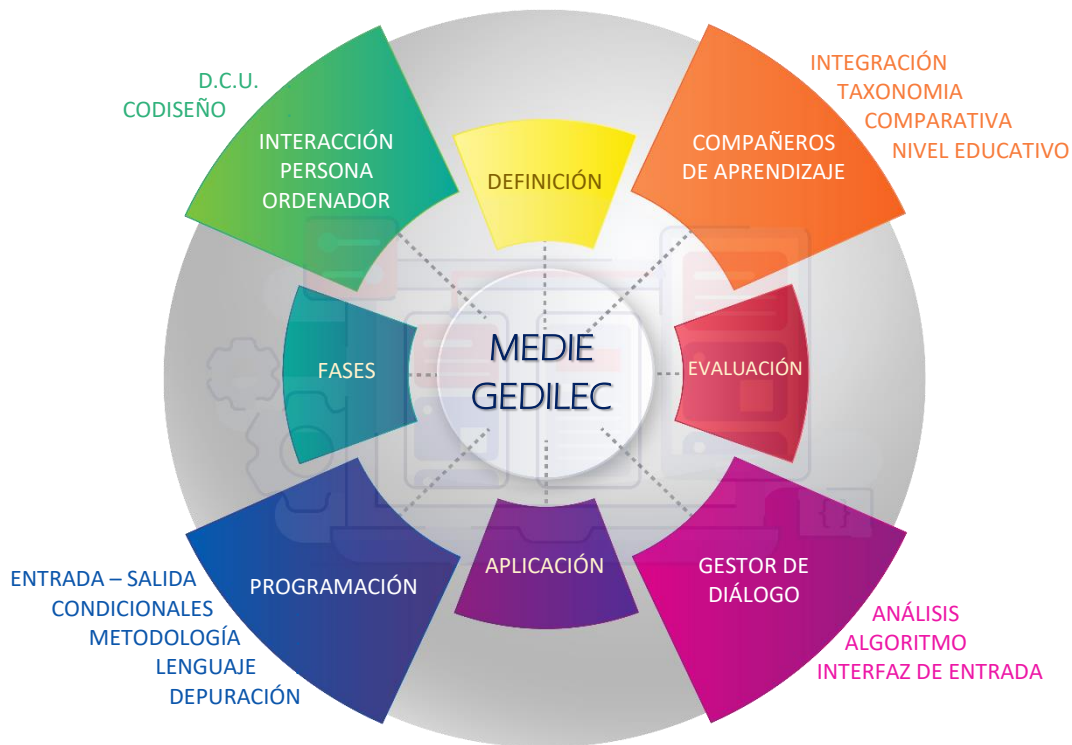


Figura 1.1 Visión global del marco de investigación de este trabajo (fuente: propia)

1.3.1 Interacción Persona Ordenador

1.3.1.1 Diseño centrado en el usuario: DCU es una aproximación al diseño aplicaciones que sitúa al usuario en el centro del proceso (Norman, 2002), al tiempo de planificar, gestionar y llevar a cabo proyectos de creación, mejora e implementación de productos interactivos.

En este contexto el Diseño Centrado en el Usuario (DCU) de sistemas interactivos conlleva realizar un diseño pensando en y para el usuario, convirtiéndole en el punto central del desarrollo (Granollers et al., 2002).

1.3.1.2 Co-diseño: Es un método que permite el desarrollo de productos por medio de la experimentación, acciones basadas en la participación activa de los sujetos de estudio para el éxito del diseño (Andersen et al., 2015).

1.3.2 Compañeros de aprendizaje

1.3.2.1 Definición: Los compañeros de aprendizaje son personajes digitales animados o no que funcionan para simular la interacción humana-similar, que podría brindar una oportunidad para simular dicha interacción social en el aprendizaje basado en computadora (Kim y Baylor, 2006).

1.3.2.2 Integración: Son técnicas y aspectos a considerar en la incorporación de las Tecnologías de la Información y Comunicación (TIC) en el ámbito educativo, con los Agentes Conversacionales Pedagógicos (ACP) en las aulas (Kim y Baylor, 2006).

1.3.2.3 Taxonomía: Se encuentra estructurada por diez criterios principales: Rol y actitud, modalidad de interacción, tipo de animación, posibilidades afectivas, tipo de personaje virtual, evolución adaptativa, ubicuidad, dominio, idioma, edad (Tamayo-Moreno y Pérez-Marín, 2014).

1.3.2.4 Comparativa: Los compañeros de aprendizaje pueden compararse según su objetivo de aprendizaje, características del lenguaje, contenidos, ejecución y retroalimentación.

1.3.2.5 Nivel educativo: Etapa educativa (Infantil, Primaria, Secundaria y/o Universitaria) que se encuentra focalizado el proceso educativo en programación, el presente trabajo se aplica en Educación Primaria.

1.3.3 Gestor de Diálogo: Herramienta que permite la comunicación que incluyen en su diseño fuentes de información, tales como: resultados del procesamiento del lenguaje natural, o de consultas a bases de datos, conocimiento acerca del dominio de aplicación, conocimiento acerca de los usuarios y la historia del diálogo (Morales-Rodríguez y Domínguez-Martínez, 2011). En este contexto, este apartado comprendería:

1.3.3.1 Interfaz de entrada-salida: Permite el manejo de entrada-salida: el ingreso por texto y/o por voz. Incluyendo el manejo de bibliotecas para el procesamiento de entradas de voz, mediante la comunicación segura con https.

1.3.3.2 Algoritmo: Secuencia de pasos que guía el diálogo, para establecer la interacción en lenguaje natural entre el acompañante y el usuario. En este sentido permite cumplir el objetivo de aprendizaje.

1.3.3.3 Análisis: Revisión de la interacción de los diálogos, identificando opciones de mejora automática o no, que permite crear nuevos contenidos, para optimizar las siguientes respuestas.

1.3.4 Programación: Es la capacidad de crear programas para conseguir objetivos con recursos informáticos (Pérez-Marín et al., 2017). Es así que el pensamiento lógico puede ser útil no solo para estudiantes o profesionales de Ciencias de la Computación, sino también para cualquier persona (Wing, 2006). Esta es la razón por la cual la enseñanza de programación se ha implementado en la Educación Primaria en varios países de todo el mundo (Heintz et al., 2016). Algunos de los conceptos que se tratan son:

1.3.4.1 Entrada-salida: Concepto de programación con sentencias básicas de programación, que permiten el ingreso de información por parte del usuario (entrada) y presentación de la información (salida) generalmente por pantalla.

1.3.4.2 Condicionales: Estructura de programación que realiza una comparación con una expresión lógica, si el valor de la expresión es verdadero ejecuta ciertas instrucciones, si el valor es falso ejecuta otras instrucciones (Mathieu, 2014).

1.3.4.3 Bucles: Estructura que brinda la posibilidad de ejecutar un grupo de instrucciones por más de una vez (Mathieu, 2014).

1.3.4.4 Metodología: MECOPROG contribuye a la enseñanza de la Programación permitiendo desarrollar el pensamiento lógico deseado en estos niveles básicos de programación mediante el uso de metáforas, como por ejemplo programa como cocinar, programa como receta y memoria como despensa (Pérez-Marín et al., 2017).

1.3.4.5 Lenguaje de programación: Forma de comunicación con el ordenador mediante una serie de instrucciones, permite escribir un conjunto de órdenes secuenciales, formando un programa. En este contexto, pueden ayudar o distraer a programadores novatos, por lo que la elección no es neutral y sus características deben analizarse cuidadosamente para fomentar un buen contexto de aprendizaje (Lodi et al., 2018).

1.3.4.6 Depuración: Consiste en mensajes de error pueden marcar la diferencia en la experiencia de programadores novatos, en el sentido de que, cada estudiante puede presentar diferencias en la solución del problema, ya sea que programe solo o con la ayuda de otra persona (Becker et al., 2016).

1.4 Escenario

Con la intención de ejemplificar el proceso de aplicación de la metodología MEDIE_GEDILEC, para integrar y validar el sistema educativo interactivo, con el compañero de aprendizaje Alcodey, se describe los elementos del escenario y las etapas de la metodología propuesta con ejemplos prácticos.

La Figura 1.2 ilustra el escenario presencial, el lugar donde se da es un laboratorio de informática de la unidad educativa que dispone de: 1 ordenador para el docente, 40 ordenadores para estudiantes, 1 proyector de video, 1 pizarra digital y conexión a Internet.

De acuerdo al horario de clase de programación inicial, los estudiantes asisten dos veces por semana al laboratorio de computación por 40 minutos, un docente y un grupo de estudiantes de (32 a 36 niños) de entre 8 a 12 años.

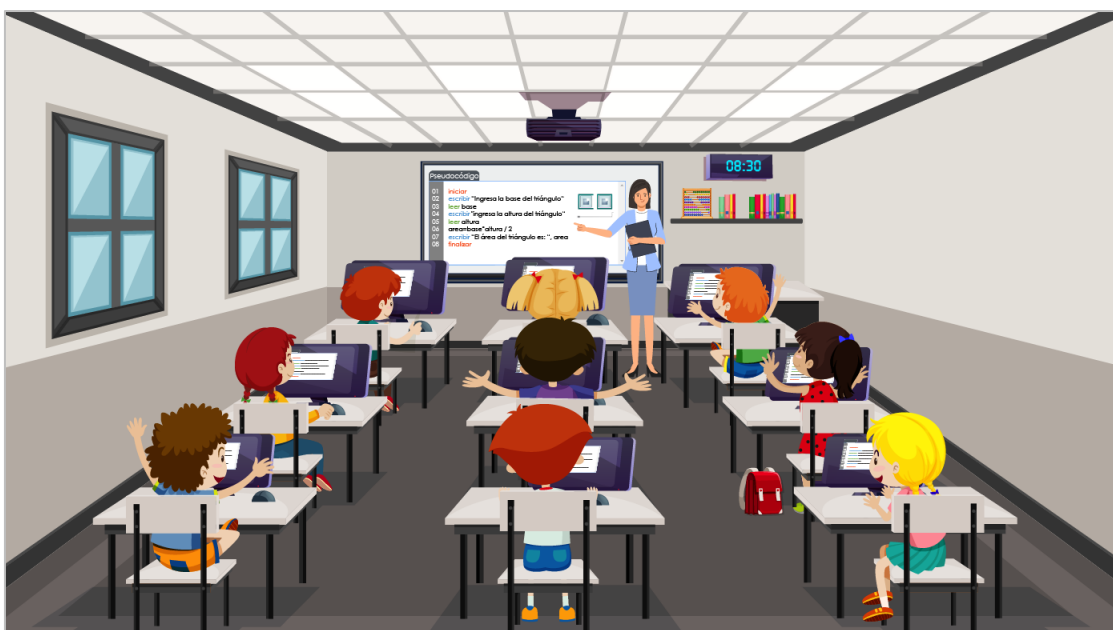


Figura 1.2 Representación del escenario presencial (fuente: propia)

La Figura 1.3 describe el escenario de la sesión en línea, donde por medio de una plataforma de videoconferencia una vez por semana se conectan por 60 minutos a la sesión de clase, el docente y los mismos grupos de estudiantes.



Figura 1.3 Representación del escenario en línea

Mantener la atención de niños de estas edades es una tarea complicada para un docente y más aún si el grupo es numeroso, el panorama se complica si la clase es de programación, por lo anterior se requiere una metodología que permita lograr los objetivos planteados.

A continuación se enlista un breve repaso de las etapas de MEDIE_GEDILEC:

Comunicación y co-diseño con docentes y estudiantes: se recomienda establecer reuniones presenciales y/o virtuales con los docentes que van a enseñar programación a

los estudiantes y con los propios estudiantes para conocer sus necesidades y conocimientos previos.

Análisis de diálogos: se analiza la forma de comunicación hablada entre los estudiantes y el compañero con técnica de prototipado Storytelling. Los niños escriben una historia sobre cómo querrían aprender los conceptos de programación. Del análisis de estas historias se puede analizar el vocabulario que emplean, la frecuencia de uso de determinados términos, y el general, el tipo de diálogo que mantendrá el compañero de aprendizaje con los estudiantes según el entorno donde se está usando.

Gestor de diálogos: se incluye una API text-to-speech que permite ingresar información por voz. También se debe escoger el lenguaje y entorno de programación para la gestión del diálogo estudiante-compañero de aprendizaje.

Se podrían usar lenguajes de marcado como AIML y gestores ya existentes como Lex de Amazon o Dr. Watson de IBM que ya ha sido usado en otros agentes para enseñar programación en Primaria con buenos resultados.

Editor de código: se debe elegir el lenguaje de programación que enseñará el compañero de aprendizaje según las necesidades identificadas se selecciona pseudocódigo, que puede ser particularmente útil para la enseñanza inicial de programación.

Según el lenguaje escogido se deberá usar o desarrollar un intérprete o compilador adecuado para permitir la ejecución y depuración del código en la interacción con el estudiante.

Validación de la interfaz y funcionalidad: se pide a los estudiantes que prueben el entorno resultante con el compañero para validar estadísticamente su correcto funcionamiento y eficacia.

Sesiones prácticas: una vez validada tanto la interfaz como la funcionalidad es el momento de integrar dentro de la planificación didáctica y curricular del centro la enseñanza de la programación con el compañero de aprendizaje en el mayor número posible de sesiones en clases presenciales y/o virtuales.

1.5 Publicaciones

La presente investigación tiene como principal aporte determinar una metodología MEDIE_GEDILEC, para el: análisis, diseño, desarrollo e integración en el aula de un sistema interactivo educativo con un compañero de aprendizaje para la enseñanza de programación inicial en Educación Primaria.

Este compañero de aprendizaje llamado Alcody, se llevó a las aulas de dos escuelas de Educación Primaria una de España y otra de Ecuador, las mismas que permitieron realizar cada una de las etapas de la metodología.

El cambio de modalidad de presencial a en línea debido a la pandemia, permitió validar su interfaz y funcionalidad, en ambos escenarios. Producto de este trabajo se difundieron aportes y resultados alcanzados en las siguientes publicaciones:

- **Publicación 01:** Urrutia, E. K. M., Ocaña, J. M., Pérez-Marín, D., & Tamayo, S. (2017, Octubre). A first proposal of Pedagogic Conversational Agents to develop Computational Thinking in children. *In Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality ACM*, 2, 1–6. <https://doi.org/10.1145/3144826.3145350>
- **Publicación 02:** Ocaña, J. M., Morales-Urrutia, E., Pérez-Marín, D., & Tamayo, S. (2019). Gestión del diálogo de un Agente Conversacional Pedagógico para aprender a programar. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E19), 239-251.
- **Publicación 03:** Morales-Urrutia, E., Ocaña, J. M. C., Aguirre, J. L. S., & Pérez-Marín, D. (2019). Interfaz de usuario enfocado en el co-diseño con niños. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E22), 147-160.
- **Publicación 04:** Ocaña, J. M., Morales-Urrutia, E. K., Pérez-Marín, D., & Tamayo-Moreno, S. (2019). How to Create a Pedagogic Conversational Agent for Teaching Computer Science. *In Advanced Online Education and Training Technologies* (pp. 114-134). IGI Global. <https://doi.org/10.4018/978-1-5225-7010-3.ch007>
- **Publicación 05:** Morales-Urrutia, E. K., Ocaña, J. M., & Pérez-Marín, D. (2020). How to Integrate Emotions in Dialogues With Pedagogic Conversational Agents to Teach Programming to Children. *In Innovative Perspectives on Interactive Communication Systems and Technologies* (pp. 66-91). IGI Global. <https://doi.org/10.4018/978-1-7998-3355-0.ch004>
- **Publicación 06:** Morales-Urrutia, E. K., Ch, Ocaña, J. M., Pérez-Marín, D., & Pizarro-Romero, C. (2020, July). Promoting learning and satisfaction of children when interacting with an emotional companion to program. *In International Conference on Artificial Intelligence in Education* (pp. 220-223). Springer, Cham. https://doi.org/10.1007/978-3-030-52240-7_40
- **Publicación 07: JCR Q1 - Ocaña, J. M.,** Morales-Urrutia, E. K., Pérez-Marín, D., & Pizarro, C. (2020). Can a learning companion be used to continue teaching programming to children even during the COVID-19 pandemic? *IEEE Access*, 8, 157840-157861. <https://doi.org/10.1109/ACCESS.2020.3020007>
- **Publicación 08:** Morales-Urrutia, E. K., Ocaña, J. M., Pérez-Marín, D., & Pizarro, C. (2020). ¿Pueden los niños aprender a programar usando un entorno de programación basado en texto con un agente compañero? *Informática Educativa Comunicaciones*, 32.

- **Publicación 09: JCR Q1** - Morales-Urrutia, E. K., Ocaña, J. M., Pérez-Marín, D., & Pizarro, C. (2021). Can mindfulness help Primary Education students to learn how to program with an emotional learning companion? *IEEE Access*, 9, 6642-6660. <https://doi.org/10.1109/ACCESS.2021.3049187>
- **Publicación 10: Ocaña, J. M.**, Morales-Urrutia, E. K., Pérez-Marín, D., & Pizarro, C. (2021). How to create emotional learning companions to teach programming in Primary Education?, en proceso de revisión en Simposio *Internacional de Informática Educativa (SIIE)*.

La Tabla 1.1 describe los aportes presentados de las publicaciones realizadas acorde a las fases de la metodología MEDIE_GEDILEC:

Ord	Fases	Publicación
1	Comunicación y co-diseño: docentes y estudiantes	01 – 03 - 04 - 10
2	Análisis de diálogos	02 – 03 - 04 – 05 - 10
3	Gestor de diálogo	02 - 10
4	Editor de código	05 – 06 - 07 – 08 - 10
5	Validación de la interfaz y de la funcionalidad	03 – 04 - 07 - 10
6	Sesiones prácticas	04 - 10

Tabla 1.1 Fases de la metodología MEDIE_GEDILEC por publicaciones (fuente: propia)

1.6 Estructura

El presente trabajo, se encuentra organizado en seis capítulos, que se describen a continuación:

- En el Capítulo 1 se realiza una introducción, colocando el contexto general del presente trabajo y determinando la motivación de la investigación.
- En el Capítulo 2 se efectúa una revisión del marco teórico, abarcando el constructivismo (Piaget, 1973), y construccionismo (Papert, 1991).

- En el Capítulo 3 se realiza una exploración del estado del arte, justificando el trabajo de investigación en relación al ámbito de estudio de la enseñanza de programación y su impacto tecnológico de uso de agentes conversacionales.
- En el Capítulo 4 se presenta una propuesta de la metodología MEDIE_GEDILEC, que permitirá a docentes e investigadores realizar el análisis, diseño, validación e integración en el aula de un sistema interactivo educativo con un compañero de aprendizaje para la enseñanza de programación inicial en Educación Primaria.
- En el Capítulo 5 se describe cómo se aplica MEDIE_GEDILEC para el diseño, desarrollo e integración en el aula de Educación Primaria del compañero de aprendizaje Alcodey, y su validación en dos escuelas de Educación Primaria tanto en España como en Ecuador.
- En el Capítulo 6 se establecen las conclusiones obtenidas en los experimentos del proyecto y el trabajo futuro.

Capítulo 2. Marco teórico

2.1 Constructivismo

El constructivismo se originó a finales de la década de 1970 e inicios de la década de 1980. Surge como una corriente muy influyente en la educación científica (Bliss, 2002; Adey, 1999) que muestra un enfoque alejado al de dominio general para exploración de la cognición en el aprendizaje de las ciencias (Piaget, 1973).

A este respecto, varios autores mencionan que el constructivismo se convirtió en una perspectiva teórica que sustenta gran parte de la investigación, el desarrollo docente y la actividad de desarrollo curricular en la educación científica (Solomon, 1994; Gilbert, 1995; Fensham, 2004).

Así también, esta teoría se centra en la preocupación por cómo los estudiantes entendieron las diferentes conceptualizaciones en las ciencias; en otras palabras, existió un amplio interés en explorar el pensamiento de los individuos (el antes y después de la instrucción) (Driver y Easley, 1978; Driver y Erickson, 1983; Gilbert et al., 1982; Gilbert y Watts, 1983; Osborne y Wittrock, 1985).

El término constructivismo ha sido utilizado de una forma inconsistente en la literatura más amplia haciendo que éste tenga diferentes implicaciones en varios contextos (Phillips, 1995; Bickhard, 1998; Bodner et al., 2001).

Al constructivismo, en ciertas ocasiones se lo vincula con enfoques epistemológicos generales, por lo que puede estar asociado con una postura epistemológica que se encuentra en desacuerdo a lo que mencionan los supuestos científicos positivistas tradicionales en relación con la posibilidad de desarrollar un conocimiento objetivo de un mundo externo que existe independientemente de las mentes humanas (Glaserfeld, 1989).

A lo largo de la historia, se han desarrollado una serie de trabajos académicos llevados a cabo por investigadores comprometidos con el análisis del proceso de enseñanza y aprendizaje. Entre algunos defensores de este enfoque filosófico se destacan a Bruner (1966), Piaget (1929), Vygotsky (1934), Dewey (1938), Montessori (1952) y Kelly (1963). No obstante, de todos estos exponentes se destaca a Piaget como el padre del paradigma filosófico constructivista. En parte, esto se debe a que él protagonizó la formalización de la teoría del constructivismo a través de la explicación de los mecanismos que los estudiantes utilizan con el fin de construir su propio conocimiento.

Por su parte, la educación científica adoptó varias formas de constructivismo que fueron generalmente de carácter personal y psicológico, y a veces explícitamente cognitivistas.

No obstante, se basaron en los aspectos constructivistas personales propuestos por Piaget (1929) que se centran en el sujeto epistémico individual que construye sus propias estructuras de conocimiento para interpretar el mundo.

Así también, se apoyaron en la teoría del constructo personal de Kelly (1963) que propone que un individuo usualmente desarrolla un sistema único de constructos personales, que se aplican de forma un tanto implícita, al hacer discriminaciones o dar sentido a los datos perceptivos. Asimismo, el autor hizo referencia a su perspectiva como un “alternativismo constructivo”. Muestra cómo cada individuo es capaz de construir un sistema único para generar la experiencia, así como también, cómo éste llega a cambiar su sistema de construcción con el fin de interpretar sus experiencias de manera diferente (Taber, 2016).

Por otra parte, Glasersfeld (1989) considerado como un activista del constructivismo generó un impacto importante en la educación científica a través del “constructivismo radical” (Glasersfeld, 1990), que se trató de una forma de constructivismo más psicológico. El autor se encontraba más preocupado por los fenómenos mentales; es decir, de cómo la mente construye lo que ha comprendido del mundo externo, en lugar de asegurar que la realidad ya fue creada.

Bruner (1960) consideró al constructivismo como un enfoque del interaccionismo para el desarrollo del lenguaje, en aspectos relacionados con las intenciones comunicativas, expresión lingüística, uso del lenguaje en la primera infancia y el papel de los padres, entre otras.

Bruner basó su trabajo en la teoría social constructivista, mencionando que la participación en la vida social de un grupo, y el uso significativo del lenguaje implica un proceso interpersonal, intersubjetivo y colaborativo de creación del significado compartido (Aljohani, 2017).

Así también, Ausubel et al., (1968) realizó aportaciones sobre las condiciones requeridas para la existencia de un aprendizaje significativo a través de un proceso en el que el estudiante relacione la enseñanza con el material que se encuentra ya disponible en su estructura cognitiva.

En la misma línea argumental, Honebein (1996) concibe al constructivismo como un enfoque que sostiene que los individuos son los encargados de construir su propia comprensión y comprensión del mundo a partir de experiencias y reflexiones de experiencias vividas (Hein, 1991; Cashman et al., 2008).

Por lo tanto, para el constructivista, el construir significa aprender a través del espíritu de experimentación y acción (Mustafa y Kalender, 2007) y anula por completo el concepto tradicional. Es decir, si los propios estudiantes presencian la realización de un fenómeno determinado, les resulta más fácil recordarlo por medio de la actividad sensorial de ver.

En este sentido, la opción más acertada que varios filósofos constructivistas proponen es que los alumnos deben involucrarse de forma directa en el proceso de enseñanza y aprendizaje a fin de que su participación le permita descubrir de forma personal el conocimiento o la verdad (Adom et al., 2016).

Finalmente, Vygotsky (1934) puso de relevancia al papel del lenguaje como herramienta para el pensamiento y la enseñanza de conceptos ya sean científicos o académicos y cómo éstos fueron entendidos por los estudiantes en relación con sus conceptos desarrollados de forma espontánea.

Así también, promovió lo que se conoce como constructivismo social, trabajo en grupo, aprendizaje, etc., promoviendo una metodología de instrucción “de arriba hacia abajo” en lugar de la “de abajo hacia arriba”, es decir, en lugar de mostrar todos los detalles que conducen a una idea principal, los alumnos son los encargados de descubrir en primer lugar la idea principal para posteriormente derivar en los detalles (Aljohani, 2017).

2.1.1 Cognitivo

La eficiencia del enfoque cognitivo en el proceso de enseñanza y aprendizaje ha sido analizada por varios académicos, llegando a resultados concluyentes de que un proceso educativo llega a su propósito de manera exitosa a través de la aplicación de estrategias cognitivas (Olson y Land, 2007; Meang, 2006; Griffiths, 2004; Holden, 2004; Martínez, 1996).

A este respecto, Bloom (1956) describió al término cognitivo como aquel que utiliza procesos cognitivos, tales como: competencias y destrezas intelectuales que generalizan técnicas que se encuentran relacionadas con materiales o problemas.

El constructivismo cognitivo se basa en el trabajo desarrollado por Piaget. Esta teoría se estructura en dos partes principales; por un lado, una denominada “edades y etapas” que predice que los niños pueden o no entender a diferentes edades, y, por otro lado, una conocida como la teoría del desarrollo que describe cómo los niños desarrollan habilidades cognitivas (Chambliss, 1996).

La teoría del desarrollo es la base principal de los enfoques constructivistas cognitivos, a través de esta se sugiere que los seres humanos no procesan y utilizan automáticamente una información, sino que deben construir su propio conocimiento, ya sea por medio de experiencias que les permitan crear imágenes mentales en sus cabezas.

El papel que desarrolla el docente dentro de esta teoría, según Piaget es el de proporcionar a los alumnos herramientas adecuadas e interesantes a fin de animarlos a construir sus propios conocimientos.

La cognición en el constructivismo piagetiano considera que el propósito principal de la educación es educar, por lo que este enfoque asume que los estudiantes llegan a la escuela

con ideas, creencias y opiniones que requieren ser cambiadas o modificadas por un docente, que ayuda con esta alteración a través de tareas o preguntas que generan dilemas para los alumnos (Kumar, 2012).

Por otro lado, Bruner desarrolló una teoría cognitiva de la instrucción, ésta se fundamenta en que los docentes son los encargados de crear una instrucción que guíe al estudiante por medio de una secuencia de declaraciones hasta que éste domine el contenido.

Por lo tanto, el alumno comprende el material, lo organiza para transformar el concepto de una manera diferente, y, posteriormente transfiere el conocimiento a nuevas situaciones (Bruner 1966).

2.1.2 Socio-cultural

La teoría del constructivismo social se encuentra basada en el trabajo pionero de Vygotsky (1978, 1987-1999). Esta teoría ha tenido gran influencia en la psicología educativa (Berliner y Calfee, 1996; Palincsar, 1998) y del desarrollo (Parke y Clarke, 2003).

Así también, en disciplinas psicológicas, tales como: la psicología de género (Gergen y Davis, 2013), psicología cognitiva (Frawley, 1997), psicología social (Jost y Kruglanski, 2002), psicología cultural (Markus y Hamadani, 2010), psicología organizacional (Nica y Potkovaru, 2015), entre otras.

Así también, Voss et al., (1995) manifiestan que, a través de esta revolución sociocultural, se hace especial énfasis en que el aprendizaje se logra, no sólo por medio de la interacción social, sino también, en contextos extraescolares.

Del mismo modo, el autor argumenta que los estudiantes, no se encuentran en la capacidad de comprender el desarrollo cognitivo individual cuando existe una ausencia del contexto social en el que se desarrollan (Gärdenfors, 2005).

Por lo tanto, el conocimiento requiere de manera obligatoria una participación activa dentro de un grupo cerrado de interacciones, en el que el dominio del conocimiento depende directamente del sujeto y su realidad (Maturana, 1978).

Los constructivistas sociales consideran al proceso de conocer como el factor principal de la interacción social, que a su vez conduce a niveles más elevados de razonamiento y aprendizaje (O'Connor, 1998). Asimismo, se menciona que el proceso de adquisición de habilidades intelectuales involucra a otros (Jones y Brader-Araje, 2002; Glasersfeld, 1989).

En este sentido, Jaworski (1994) coincide en que la esencia del constructivismo social se encuentra en el reconocimiento del poder que tiene la interacción y negociación, el conocimiento intersubjetivo puede ser visto como un producto de ésta, en la que los

individuos que participan se encuentran de acuerdo en determinadas interpretaciones que son representadas a través de un discurso y comunicación no verbal.

Tanto Vygotsky como Piaget propusieron diferentes mecanismos para explicar el aprendizaje de acuerdo con las perspectivas del constructivismo social. Según Vygotsky la vida social es de suma importancia, mientras que, desde el punto de vista del segundo, el individuo es el que tiene mayor relevancia (Bozkurt, 2017).

En este contexto, de acuerdo a Piaget la perspectiva constructivista se basa en el constructivismo radical, que se centra en los procesos cognitivos individuales en combinación con la interacción social.

Aunque menciona que los aspectos sociales que se presentan en la interacción en el aula son importantes, destaca que son los aspectos individuales los que intervienen en la construcción del conocimiento (Ernest, 1994; Rogoff 1999; Wertsch 1985).

A este respecto, Piaget (1970) mencionó que "... ya no hay necesidad de elegir entre la primacía de lo social o la del intelecto: el intelecto colectivo es el equilibrio social resultante de la interacción de las operaciones que entran en toda cooperación".

Así también, el autor considera que las acciones que desarrollan los niños en relación con el mundo que los rodea, provocan el aprendizaje y no hay lugar para la enseñanza (Jaworski, 1996).

Por lo que es muy probable que la interacción social existente, genere distintos puntos de vista y se cree un conflicto cognitivo, lo que puede ayudar en parte al crecimiento cognitivo del individuo (Palincsar, 1998).

Por otro lado, el constructivismo social visto desde el punto de vista de Vygotsky priorizó las influencias sociales y lingüísticas existentes en el aprendizaje y la creación del significado.

En este sentido, afirmó que "La dimensión social de la conciencia es primordial en el tiempo y, de hecho, la dimensión individual de la conciencia es derivada y secundaria" (Vygotsky, 1978).

De acuerdo a Vygotsky la interacción social a través de la participación de los individuos en una serie de actividades conjuntas y el proceso de internacionalización de las influencias del trabajo colectivo brinda a los estudiantes un desarrollo intelectual y la adquisición de conocimientos es guiada por uno o más compañeros o a su vez por un adulto que se encuentra más capacitado (Rogoff, 1999).

Al respecto, Vygotsky (1978) desarrolló el concepto de Zona de Desarrollo Próximo (ZDP) a la que describió como un instrumento necesario para examinar las diferentes contribuciones de los individuos al entorno de aprendizaje, así como, el papel de la intersubjetividad en los estudiantes del proceso.

2.2 Construccinismo

En la década de los 80's surge el construccionismo como una filosofía educativa propuesta por Seymour Papert. La tesis central en la que se basa esta teoría, consiste en que para que los estudiantes sean capaces de generar la construcción del conocimiento en su mente, es necesario que, construyan estructuras externas, físicas o vitales que les permita reflexionar y compartir (Papert, 1991; Noss y Hoyles, 2017).

A este respecto, Harel y Papert (1991) mencionan que el construccionismo comparte la connotación constructivista del conocimiento, en el sentido, de construir estructuras de conocimientos cualquiera que sea el contexto de aprendizaje en el que se encuentre el alumno.

El construccionismo es considerado como una pedagogía que va más allá del constructivismo promulgado por Piaget, en el sentido, de que considera que el niño es el constructor de sus propias estructuras cognitivas en interacción con el medio que lo rodea.

Por lo tanto, el construccionismo, se opone a la idea de que la única manera, a través de la cual se puede lograr un mejor aprendizaje implica el perfeccionamiento de la instrucción (Fino, 2017). No obstante, Papert (1993) sin quitarle valor a la instrucción coincide con Piaget que cada acto de enseñanza puede restarle una oportunidad de descubrimiento.

En sentido general, se plantea la hipótesis de darle una caña y enseñarle a pescar. En otras palabras, Papert considera que la educación tradicional se encarga de codificar lo que cree que los individuos requieren saber.

El construccionismo se fundamenta en el supuesto de que a los niños les irá mejor si encuentran el conocimiento específico que necesitan por sí mismos; la educación organizada o informal en este caso puede ser de apoyo en los aspectos morales, psicológicos, materiales e intelectuales a sus esfuerzos.

2.2.1 Ideas principales

La teoría del construccionismo (Papert, 1980) tiene su base en la establecida previamente por Piaget (1968). Papert se centra en el principio básico de que el aprendizaje significativo se alcanza cuando los estudiantes construyen de forma activa un producto en el mundo real.

Asimismo, el autor hace especial referencia al arte de aprender o “aprender a aprender” y a la importancia que tiene el hacer las cosas en el proceso de aprendizaje (Papert y Harel, 1991; Ackermann, 2001; Han y Battacharya, 2001; Kafai, 2006; Hoban et al., 2010).

Por otra parte, Jha (2012) pone de manifiesto que, desde el punto de vista de los construccionistas, el conocimiento no sólo se puede construir a través de la interacción

del individuo con su contexto, sino que, también se puede generar mediante la interacción con otras personas pertenecientes a una comunidad social específica.

Al respecto, Papert realizó una reflexión en relación al significado de la palabra pedagogía señalándola como el arte de enseñar. No obstante, hace una reflexión en referencia a la inexistencia de una palabra paralela para el arte de aprender, en el sentido de que, estaba interesado en el proceso de cómo los estudiantes inician conversaciones con sus propios artefactos o con el de los demás, y cómo éstas a su vez promueven el aprendizaje auto dirigido o la construcción de nuevos conocimientos (Ackermann, 2001).

En este contexto, varios autores concuerdan que la creación y el intercambio de un producto, es una forma más eficaz cuando se desarrolla en un entorno colaborativo.

A este respecto, Papert pone de manifiesto que la idea del construccionismo va más allá del “aprender haciendo”, ya que es un concepto mucho más rico, multifacético y amplio en todas sus implicaciones (Papert y Harel, 1991).

Asimismo, explicó que el construccionismo, la palabra con N en oposición a la palabra V, comparte la visión de constructivismo del aprendizaje como la de construir estructuras de conocimientos por medio de la aceptación individual progresiva de las acciones (Ackermann, 2001).

Por lo tanto, el construccionismo, tiene por objetivo principal teorizar estrategias que especifiquen las formas en la que las personas aprenden y una vez identificadas ayudarles a aprender, a través del uso de artefactos adecuados (Taber, 2016).

En función a lo anterior, Rob y Rob (2018) manifiestan que la filosofía de Papert sobre el construccionismo se puede resumir en las siguientes dimensiones del aprendizaje:

- Experiencia pasada: Los estudiantes poseen un conocimiento intrínseco de sus experiencias pasadas.
- Nueva experiencia: En esta etapa los alumnos construyen una experiencia desde cero. Es decir, una experiencia propia a partir de hacer algo nuevo.
- Artefacto significativo: Es el resultado o producto de una actividad educativa, por lo que debe ser significativo, es decir, responder a metas u objetivos educativos propuestos como parte de la actividad. No cualquier producto finalizado corresponde a la construcción de un conocimiento.
- Producto mundo real: Cuando el producto responde o da solución a un problema real a través de la construcción de un producto, el aprendizaje se vuelve más significativo.
- Colaboración: El trabajo en conjunto es de relevancia para la creación de conocimientos, ya que el intercambio de aprendizajes fortalecen el conocimiento.

- **Compartir:** La creación de un producto por parte de un estudiante debe ser compartida con otros.
- **Herramientas:** A fin de mejorar el proceso educativo, se hace necesaria la aplicación de herramientas educativas, informáticas o de comunicación.
- **Medios de comunicación:** Existen una variedad de medios que pueden ser utilizados.
- **Contexto:** Se conoce como el entorno o escenario de aprendizaje.

Así, el construccionismo de acuerdo a Papert (1993) mira más de cerca que otros -ismos la idea de generar una construcción mental. Además, concede una importancia especial al papel de la construcción en el mundo, y, toma más en serio la idea de construir en la cabeza más de un tipo de construcciones a través de diferentes métodos y materiales utilizados.

Por lo tanto, el construccionismo hace énfasis en los objetos externos, como construcciones en el mundo, de modo que se puedan discutir, mostrar, examinar, probar y admirar. En otras palabras, compartir una creación puede resultar no solo, en su perfeccionamiento, sino que se puede obtener una comprensión más amplia de las perspectivas de otros (Nogueira, 2017).

2.2.2 Aplicaciones prácticas

La programación es intrínsecamente construccionista, ya que implica de manera obligatoria la construcción de un aparato o artefacto que puede ser mostrado o compartido para la enseñanza.

En este contexto, desde el punto de vista construccionista del aprendizaje, los lenguajes de programación cumplen un papel importante, en el sentido de que, son considerados como medios claves que permiten controlar el comportamiento de una máquina (Lodi et al., 2018).

A continuación, se detallan diferentes experimentos del entorno de la programación relacionados con el construccionismo.

2.2.2.1 LOGO fue diseñado por Papert, Feurzeig y Solomon con fines netamente educativos (Papert, 1980). Esta herramienta presenta un entorno gráfico en el que un objeto con forma de “tortuga” se mueve por toda la pantalla.

Ésta tiene por objeto brindar ayuda a los estudiantes, específicamente de sexto a octavo año, con una especie de auto identificación.

A este respecto, los movimientos que desarrolla esta tortuga son muy similares a los que realiza en el mundo real, los patrones que va dibujando sirven para que los alumnos

construyan sus conocimientos y comprensión relacionada con geometría y principios matemáticos (Abelson y DiSessa, 1986). Ver Figura 2.1 con un ejemplo de la pantalla de LOGO.

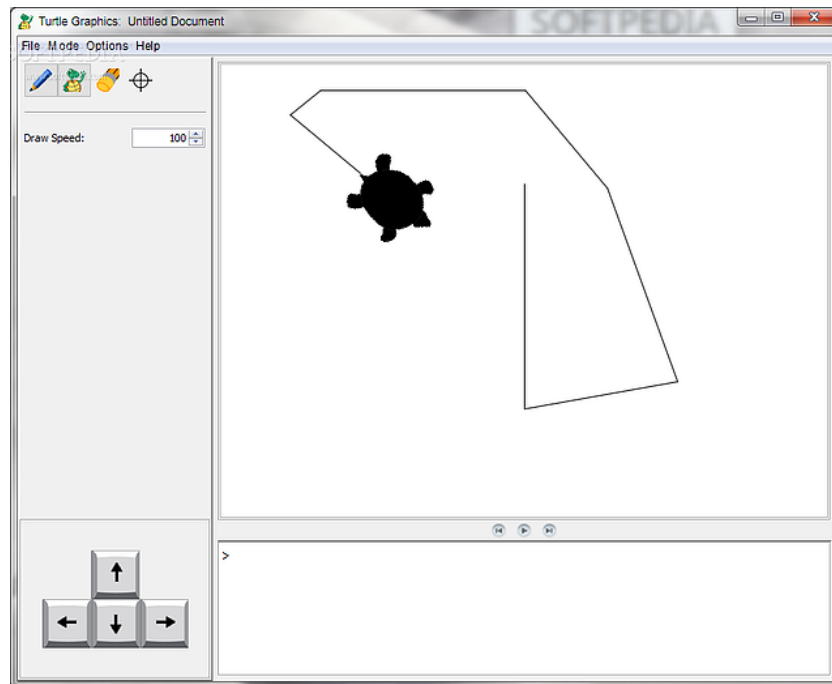


Figura 2.1 Interfaz de Mindstorms – LOGO (fuente: Papert, 1991)

2.2.2.2 Smalltalk (Goldberg y Kay, 1976) surge a inicios de la década de los setenta, cuando el Learning Research Group a través del Xerox Parc Research Center buscaba la creación de diapositivas informáticas personales.

Éstas debían contar con interfaz de usuario intuitivas y con una programabilidad explícita, esta herramienta fue diseñada para una audiencia que mostraba comodidad al utilizar la programación y los dispositivos informáticos en los entornos del aprendizaje (Kay, 1993).

En línea de la herramienta LOGO, en Smalltalk todo es un objeto que reacciona a un mensaje. Así también, introduce la idea de que todo sistema es programable, se pueden realizar todo tipo de modificaciones incluso en estado de ejecución. Puede verse una imagen del uso de Smalltalk en la Figura 2.2.

Posteriormente, se desarrolló una aplicación específica para niños conocida como Squeak Smalltalk: E-toys (Kay et al., 1997) que proporcionó un escenario de personajes divertidos que poseen movimientos de forma simultánea.

Estos juguetes electrónicos luego evolucionaron en Scratch (Resnick, 2012), herramienta en la que la programación fue reemplazada por bloques visuales.

```

to box | x y size tilt
(odraw = (@place x y turn tilt. square size.
oundraw = (@ white, SELF draw. @black)
oturn = (SELF undraw. 'tilt <- tilt + :. SELF draw)
ogrow = (SELF undraw. 'size <- size + :. SELF draw)
ISNEW = (SELF undraw. 'size <- size + :. SELF draw)

```

Figura 2.2 Smalltalk (fuente: Goldberg y Kay, 1976)

2.2.2.3 Pascal Básico ha sido un lenguaje de uso para la elaboración de proyectos personales e incluso de conectividad generalizada a internet. No obstante, la popularidad que alcanzó esta herramienta se debió a la disponibilidad de ésta en dispositivos informáticos personales y domésticos (Kurtz, 1981).

Así también, el lenguaje que se implementó fomentó los estilos de aprendizaje de forma incremental y de prueba y error que son típicos del enfoque constructorista (Brin, 2016).

2.2.2.4 Scheme o esquema raqueta (Abelson et al., 1998) es un tipo de lenguaje que se encuentra estandarizado desde el año 1999 por IEEE, contando actualmente con una amplia comunidad de usuarios.

La relevancia que tiene en la educación está relacionada con un libro “Estructura e interpretación de programas de computadora” (Abelson et al., 1996) que tuvo especial impacto en la práctica de la educación en programación.

El libro presenta ciertas particularidades, con relación a la programación como una forma de organización del pensamiento y la resolución de problemas. Scratch, Snap, Alice y otros, se trata de objetos predefinidos, programables que evolucionan en un entorno virtual genérico en que se pueden definir todo, a través, de bloques de declaración.

Al respecto de Scratch, Resnick et al., (2009) menciona que se trata de un entorno de programación basado en bloques visuales, fue creado en 2007 por el MIT Media Lab. A diferencia de algunos lenguajes de programación tradicionales existentes que requieren de declaraciones de código y reglas de sintaxis complejas, esta herramienta utiliza bloques de programación gráfica que se unen automáticamente como piezas de un puzle, cuando tienen un sentido sintáctico (Ford, 2009).

Así también, el enfoque de Scratch fue adaptado a los lenguajes de programación convencionales Alice (Dann et al., 2006) presentando los mundos de Alice en 3D que lo hace muy atractivo para los estudiantes, con el fin de que éstos se encuentren motivados a programar animaciones en 3D. Puede verse una imagen de Snap! en la Figura 2.3.

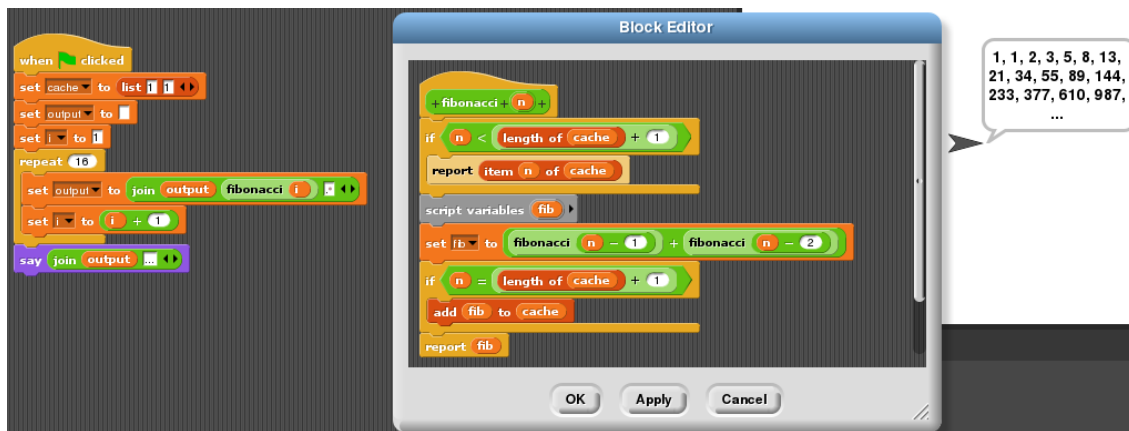


Figura 2.3 Snap! (fuente: Resnick et al., 2009)

2.3 Construccinismo social

Berger y Luckmann (1966) en su trabajo sobre la construcción social de la realidad mencionan que la sociología del conocimiento como tal, debe preocuparse de todo lo que pasa por “conocimiento” en la sociedad, y, por lo tanto, por la construcción de la realidad social de la vida cotidiana de los miembros de una sociedad.

El mundo de la vida cotidiana que se considera como significativo y coherente se mantiene de forma intersubjetiva en función de los pensamientos y acciones.

La realidad junto con la estructura espacial y temporal, no son consideradas como una problemática, ya que todo lo que se comprende y aprehende resulta de interacciones directas y recíprocas, a través, de esquemas tipificatorios que trascienden fronteras temporales y espaciales, en las que el lenguaje es necesario para este fin, dado que los signos lingüísticos consiguen un desapego máximo del aquí y ahora de la vida cotidiana.

La sociedad está conformada por una realidad tanto objetiva como subjetiva, en lo que respecta a la realidad objetiva se reconocen acciones tanto habituales como significativas que han sido institucionalizadas. A su vez, en un mundo institucional se experimenta una realidad objetiva originada en la “tipificación de las actuaciones propias y ajenas” (Berger y Luckmann, 1966).

Sin embargo, existe una pequeña cantidad de conocimientos de carácter común de significados reconocidos que pueden ser más o menos constantes y, que se transmiten de una generación a otra.

Por otro lado, desde un punto de vista subjetivo los individuos deben pertenecer a una sociedad a través de la socialización, ésta proporciona congruencia a los procesos subjetivos de aprehensión del mundo.

Por lo tanto, el aprendizaje implica interiorizar las actitudes y roles de otras personas y es considerado como un proceso tanto cognitivo como emocional que pasa por diferentes secuencias (Segre, 2016).

En cuanto al construccionismo social, Holzner (1972) menciona que el conocimiento se encuentra basado en una “red preestablecida de categorías y coordenadas”. Sin embargo, se basa en percepciones que en gran parte están construidas a nivel social, y que tienden a presentar distorsiones.

Para McKerrell (2016) el construccionismo social es un enfoque epistemológico de una base amplia que permite comprender cómo funcionan el lenguaje y el texto para la construcción de la vida social.

Los componentes de este tipo de construccionismo incluyen una postura crítica hacia los supuestos tácitos sobre la vida social, una creencia en la contingencia de los procesos sociales y un realismo crítico profundo que pretende cuestionar la concepción de lo que es o de la verdad que podría ser (Burr 2003).

Acorde al construccionismo social, la existencia de la realidad social y cultural, así como, los valores y las acciones significativas, surgen de la realización de las diferentes prácticas sociales. Por lo tanto, según Segre (2016) las convenciones sociales se crean y dan forma a lo que se considera válido y real y que tiene una existencia objetiva.

En este contexto, el principio en el que se fundamenta el enfoque construccionista social se basa en que la forma en la que se entiende o percibe el mundo, en relación con los objetos (incluidas las personas) y eventos dentro de él, no necesariamente debe reflejar la naturaleza de ese mundo, sino que es un producto de cómo se representa ese mundo a través del lenguaje.

Así también, el construccionismo social da una especial importancia a que los objetos y prácticas deben transformarse radicalmente. Es decir, no se puede realizar una investigación de tipo psicológica o social a través de viejas prácticas y suposiciones porque el enfoque en las estructuras y procesos psíquicos internos sería inadecuado.

Así también, los construccionistas sociales consideran que este enfoque mantiene una especial preferencia por los métodos de investigación cualitativos, debido a que éstos son los que más se adaptan para la recopilación de datos de carácter lingüístico y textuales, y, además, son considerados menos propensos a descontextualizar las experiencias y relatos de los encuestados (Burr et al., 2017).

Capítulo 3. Estado del arte

3.1 Enseñanza de la programación

El uso de la programación en las unidades educativas tiene sus orígenes en la década de 1960, en la que Papert desarrolló un lenguaje de comunicación conocido como LOGO, con el fin de que los niños puedan utilizar los ordenadores para crear juegos, pintar dibujos o componer música (Calao et al., 2015).

En este contexto, en los últimos años las diferentes políticas públicas a nivel mundial han reconocido la importancia de que la sociedad cada vez más tecnificada cuente con las competencias necesarias que le permita hacer uso de las tecnologías digitales desde un punto de vista más crítico y creativo (Papadakis, 2016; Redecker, 2017).

La programación en algunas ocasiones es un proceso difícil de aprender (Qian y Lehman, 2017). Esto puede deberse a que aprender programación requiere del desarrollo y dominio de la semántica del lenguaje de programación (Greg et al., 2017), herramientas de constante cambio (Ko et al., 2004), patrones comunes de computación (Lister et al., 2004); y, ciertas habilidades básicas de ingeniería de software, tales como: depuración, pruebas, diseños de programas, entre otras (Ko et al., 2011).

Del mismo modo, los estudiantes requieren otro tipo de habilidades y capacidades relacionadas con autorregulación (Falkner et al., 2014) con el fin de que aprendan a regular tanto su aprendizaje como su proceso de programación (Loksa y ko, 2016; Loksa et al., 2016).

En la misma línea, Jenkins (2002) menciona que existen diferentes fuentes de dificultades que deben superar los estudiantes al momento de programar. Algunas de las áreas problemáticas resultan, por ejemplo, de lo expresado por Perkins y Martin (1986) en relación a la existencia de cuatro tipos de conocimiento: parcial, inerte, fuera de lugar y conglomerado.

Du Boulay (1986) mencionó diferentes áreas curriculares (orientación, máquina teórica, notación, estructuras pragmáticas) que pueden generar dificultades en la programación, y, además, agregó en su investigación la existencia de errores específicos que cometen los estudiantes, tales como: mala aplicación de la analogía, interacciones y sobre generalizaciones.

Del mismo modo, Spohrer y Soloway (1986) enunciaron dos categorías de errores cometidos por los estudiantes: por un lado, problemas basados en constructos, refiriéndose a aquellos relacionados con la semántica de las construcciones de programación; y, por otro, problemas de composición de planes.

Finalmente, Sorva et al., (2013) exponen una serie de desafíos a los que están expuestos los programadores novatos, entre los cuales se mencionan a las percepciones estáticas de la programación, conceptualizaciones erróneas, dificultades para entender las computadoras y el rastreo y estado de la computadora.

Otro aspecto a destacar es que, los docentes encargados de impartir estas asignaturas, en algunos casos, no se encuentran en la capacidad de comunicar el entusiasmo o la comprensión de los estudiantes de lo que sucede al interior de una computadora para que funcione (Wilson y Moffat, 2010).

En la misma línea argumental, Fessakis et al., (2013) mencionan que el éxito que tenga la programación en las aulas no depende tan sólo de la tecnología disponible, sino también, de las actividades de aprendizaje y el material de apoyo que puede ser integrado con facilidad en la práctica escolar y, que a su vez es elaborado por docentes bien informados y preparados.

Por otro lado, Malliarakis et al., (2016) mencionan que existen alternativas o soluciones que pueden contribuir a afrontar desafíos propios de la educación en programación de computadoras. Entre éstas, Johnson et al., (2014) coinciden en que una opción alentadora representa la utilización de juegos educativos como una herramienta educativa, ya que muestra un mayor rendimiento (Papastergiou, 2009) e incremento de la motivación en los estudiantes (Connolly et al., 2012).

En la misma línea argumental, Ferguson y Garza (2011) mencionan que este tipo de juegos fomentan un mayor número de habilidades y resultados de aprendizaje. Además, optimizan el enfoque de atención de los estudiantes, así como, su capacidad de observación e identificación de objetos en sistemas tanto simples como complejos (Green y Bavelier, 2012), y, a la vez, permiten que éstos sean más receptivos a nuevas experiencias (Ventura et al., 2013).

3.1.1 Motivación

En décadas recientes, los entornos de programación han alcanzado gran relevancia en la enseñanza primaria, ya que permiten a los estudiantes experimentar, compartir y generar nuevos productos basados en códigos (Saez-López et al., 2019).

Así también, los modelos y métodos computacionales facilitan la resolución de problemas, elaboración de diseños de sistemas que a los estudiantes se le dificultaría realizar de manera individual (Freeman et al., 2017).

En la misma línea, autores como Kim et al., (2015) y Kucuk y Sisman (2017) coinciden que, la programación en las escuelas puede servir de apoyo en el aprendizaje de diferentes asignaturas como: ciencia, tecnología, matemáticas e informática.

Así también, la programación es utilizada para mejorar el aprendizaje a través del pensamiento lógico y la creatividad y la aplicación de enfoques de enseñanza, tales como: aprendizaje basado en proyectos, aprendizaje basado en problemas y estrategias de aprendizaje colaborativo con el fin último de promover mayor grado de autonomía y experiencias de aprendizaje activo (Mergendoller et al., 2006; Weng-yi Cheng et al., 2008; Grant, 2011).

La importancia de enseñar habilidades de pensamiento lógico desde una corta edad resulta un factor clave (Fletcher y Lu 2009), ya que la programación es considerada no sólo como una herramienta que permite desarrollar tareas cognitivas relacionadas con el pensamiento lógico, sino también, es una demostración de las competencias computacionales que posee el estudiante (Grover y Pea, 2013).

El desarrollo de las habilidades computacionales en la educación básica ha despertado especial interés. Es así como en una revisión de cómo Australia, Reino Unido, Estonia, Finlandia, Nueva Zelanda, Noruega, Suecia, Corea del Sur, Polonia y Estados Unidos enseñan programación en Informática en su Educación Primaria y Secundaria, revelando que normalmente la programación impartida en la asignatura de Informática es obligatoria en Educación Primaria y optativa en Educación Secundaria (Heintz et al., 2016).

El proceso de aprendizaje que implica codificar, y, a la vez desarrollar el pensamiento lógico resultan habilidades que contribuyen de manera directa al aprendizaje de los niños, en el sentido, de que éstos se encuentran en capacidad de enfrentar situaciones que se les presentan en la vida diaria, y, además les permite mejorar las relaciones de colaboración entre seres humanos y máquinas (Papadakis et al., 2016).

A este respecto, Flannery et al., (2013) coinciden que, en determinadas ocasiones, los dispositivos móviles tienden a generar un entorno de aprendizaje interactivo que resulta de gran beneficio para los estudiantes.

Así también, con el fin de motivar a los niños a incursionar en el mundo de la programación se han diseñado una serie de lenguajes de programación que pueden ser programados visualmente sin que exista la necesidad de aprender sintaxis, como es el caso de los lenguajes tradicionales (Moreno-León y Robles, 2016).

En este sentido, la programación visual permite que los estudiantes, en muchos de los casos principiantes, desarrollen el pensamiento crítico y analítico a través de la resolución de problemas (Papadakis et al., 2018).

Otros enfoques se basan en el uso de robots, ejercicios sin tecnología con sitios web como Code.org o el uso de Agentes Conversacionales Pedagógicos (ACP), son agentes virtuales autónomos y anónimos que usualmente tienen apariencia humana. Además son amigables y efectivos para los usuarios, gestionan el diálogo y generan respuestas (Ayuso y Rioja, 2005). En este contexto, el uso de los compañeros de aprendizaje en el ambiente educativo

puede tener un efecto positivo en la percepción de los estudiantes sobre la experiencia de aprendizaje: el efecto Proteus (Yee y Bailenson, 2007) ; la presencia de un agente en un entorno interactivo, aunque no sea animado, puede tener un efecto positivo en la percepción de la experiencia educativa por parte del estudiante: el efecto Persona (Lester et al., 1997); y, los estudiantes pueden llegar a hacer un esfuerzo mayor por aprender para enseñar a su avatar que para aprender ellos mismos: el efecto Protégé (Chase et al., 2009).

3.1.2 Metodología

En los últimos tiempos, el proceso de aprendizaje se ha vuelto cada vez más activo, por lo que debe presentar características más atractivas para los estudiantes. Las nuevas generaciones muestran mayor interés por el uso de tecnologías digitales modernas, ya que el aprendizaje se da por interacciones (Prensky, 2003) con metodologías diseñadas con la finalidad de mejorar la enseñanza de programación en Educación Primaria.

Estos nuevos escenarios propuestos, permiten a los estudiantes aprender a través del uso de elementos y ambientes similares a juegos o simuladores, que a su vez incrementan el interés, la motivación y el placer en el usuario (McGonigal, 2011; Zichermann y Cunningham, 2011; Yu-Kai Chou, 2016).

3.1.2.1 MECOPROG

Es una metodología didáctica utilizada para la enseñanza de conceptos básicos de programación en Educación Primaria (Pérez Marín et al., 2020). En este sentido, Pérez-Marín et al., (2019) pusieron en marcha esta metodología a través de la utilización de metáforas, tales como: recetas, programa-despensa, memoria y cajas, con el fin de despertar el interés de los estudiantes hacia la programación con aspectos conocidos y utilizados en su vida diaria.

A este respecto, los autores mencionan que un ejemplo de metáfora utilizado es el de una cocina, en el que, con una receta, los estudiantes siguen una serie de pasos hasta lograr el resultado esperado. Así también, la cantidad de tiempo que se dedique a cada grupo de metáforas, dependerá principalmente de las respuestas que den los estudiantes.

Por lo tanto, la principal regla de este tipo de metodología, es que, no se dará inicio a un nuevo grupo de metáforas hasta que el anterior haya sido asimilado con éxito siguiendo los principios del aprendizaje significativo (Ausubel, et al., 1983).

En la Tabla 3.1 se presenta un ejemplo de metáforas propuesto por los autores de la metodología.

ID	Guion	Metáfora
M1	Programa, programación, secuencia, memoria	Receta, despensa, caja
M2	Entrada/salida Instrucciones	Boca y recto (inicio y final del sistema digestivo)
M3	Condicionales	Frigorífico inteligente

Tabla 3.1 Ejemplos de metáforas (fuente: Pérez-Marín et al., 2020)

3.1.2.2 Gamificación

En los últimos años, han sido varios los académicos e investigadores que han analizado cómo ha evolucionado la visión del mundo basada en juego y cómo estos han ido adquiriendo relevancia para el desarrollo de la sociedad (Rigóczki et al., 2018).

En este sentido, Rigóczki et al., (2017) introdujo la expresión “Game Living” que se encuentra fundamentada en el hecho de que los juegos y el comportamiento similar a un juego se integran en la naturaleza humana.

Así también, Damsa y Damsa (2014) crearon un modelo con el objetivo de vincular los juegos y la gamificación como una alternativa para mejorar la comprensión del aprendizaje.

En la misma línea argumental, Passos et al., (2011) menciona que la incorporación de los juegos en herramientas que sirven como apoyo para el desarrollo de software, que ha convertido en una tendencia de actualidad.

Por su parte, McGonigal (2011) pone de manifiesto que, el tiempo que invierte un jugador en el desarrollo del juego, éste comienza a sentirse libre, generando un empoderamiento e identificación con las misiones propuestas en los juegos. Por lo tanto, el autor menciona que, el jugador desarrolla sus habilidades creativas, a través, de la resolución de problemas.

La definición de gamificación ha alcanzado crecimiento y popularidad significativa en los últimos años (Deterding et al., 2011; Zicherman y Cunningham, 2011; MacMillan, 2011; Werbach y Hunter, 2012; Németh, 2015).

Al existir una amplia diversidad de conceptualizaciones relativas a esta metodología, la mayor parte de estudiosos concuerda con el principio derivado de Pelling (2011), en el que afirma que el objetivo principal de la gamificación es integrar y hacer uso de la filosofía, los elementos, la mecánica (desafíos, recompensas, castigos, insignias o medallas).

Además, la dinámica que presentan los juegos en entornos ajenos a los juegos para inducir ciertos comportamientos en las personas. Así como fomentar la motivación y participación colaborativa en tareas específicas.

En otras palabras, la gamificación toma todas aquellas características de los juegos, que hacen que éstos sean más divertidos, atractivos e incluso un tanto adictivos, y, las aplican para generar una experiencia mejorada en los jugadores en entornos no relacionados al juego, tales como: la escuela, el lugar de trabajo, una aplicación de software, etc. (Rigóczki et al., 2018).

Así también, Formann (2017) describe a la gamificación no sólo como una metodología, sino también, como un enfoque nuevo, que requiere de una nueva era informacional. En esta era, los juegos están siendo considerados como herramientas que ayudan a los estudiantes en el proceso de enseñanza, rompiendo esquemas y generando pensamientos nuevos e innovadores.

En este sentido, Kapp (2012) menciona que en el ámbito educativo la gamificación utiliza elementos de los juegos, a fin de, involucrar a los estudiantes, motivándolos hacia la generación del aprendizaje y resolución de problemas reales.

Marache y Brangier (2013) presentaron un diseño que se encuentra centrado en el usuario, el mismo que se muestra en la Figura 3.1. Este diseño tiene como objetivo identificar los factores considerados en la gamificación (intención, situación, tarea, usuarios). Facilitando su diseño, al proporcionar medios para superar las dificultades detectadas en este proceso.

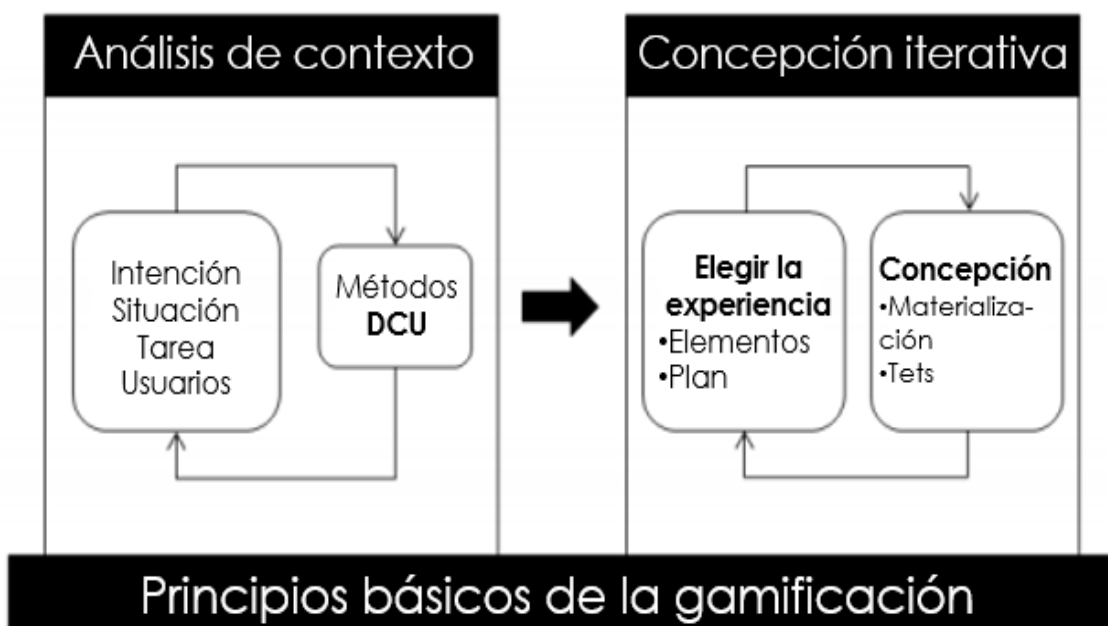


Figura 3.1 Principios básicos de la gamificación (fuente: Marache y Brangier, 2013)

3.1.3 Tecnología

3.1.3.1 Robots

En los últimos años, el rápido desarrollo de la tecnología ha hecho que el uso de las herramientas multimedia alcance importante popularidad (Toh et al., 2016).

A este respecto, los robots educativos se han convertido en una herramienta clave que permite motivar a los estudiantes y despertar su interés, a través, de diferentes actividades divertidas que sirven de apoyo para un aprendizaje interactivo (Xia y Zhong, 2018).

En este contexto, Beran et al., (2011) manifiestan que los niños tienden a utilizar cada vez más dispositivos con tecnología avanzada dentro del proceso de enseñanza.

En la misma línea, autores como Kozima y Nakagawa (2007), Wei et al., (2011), Chen et al., (2011), Kahn et al., (2012) coinciden en que la utilización de robots en la Educación Primaria tiene gran influencia en la cognición, el lenguaje, la interacción, el desarrollo social y moral de los niños y, además, permite que éstos se involucren más en sus actividades de aprendizaje.

Así también, Varney et al., (2011) ponen de manifiesto que el uso de robots fomenta el desarrollo de habilidades en equipo y el constructivismo como método de aprendizaje. De esta forma los niños dialogan, razonan, resuelven problemas y trabajan en equipo combinando sus conocimientos con el objetivo de construir robots (Hong et al., 2011).

Según Merdan et al., (2017) los sistemas robóticos son considerados como una metodología adecuada para la enseñanza de ciencia, tecnología, ingeniería y matemáticas dada su naturaleza multidisciplinaria.

En este sentido, Jojoa et al., (2010) y Chaudhary et al., (2016) concuerdan en que, el concepto de robot es de fácil entendimiento para estudiantes de todas las edades y niveles educativos.

Asimismo, Calvo et al., (2017) mencionan que los robots pueden ser utilizados para la enseñanza directa de robótica o indirecta a través de plataformas para enseñar diferentes temas, ya sean estos técnicos o no (Arvin et al., 2019).

Por otro lado, Berland y Wilensky (2015) y Hoffmann y Krämer (2013) manifiestan que existen dos tipos de robots, por un lado, el físico, y, por otro, la simulación virtual de éste.

El robot físico es la intervención robótica en el campo, así también, es considerado como una herramienta que facilita la comprensión de conceptos básicos de programación y, a su vez, consigue mejores resultados en el proceso de enseñanza que métodos tradicionales de computadoras (Barnes, 2002; Dagdilelis et al., 2005).

Además, este tipo de robots fomenta el autoaprendizaje entre los estudiantes, de manera especial en el proceso de aprendizaje de programación compleja (Fagin y Merkle, 2003).

Los robots virtuales son conocidos como robótica de simulación. Acorde al entorno virtual se pueden clasificar en robots virtuales 2D y 3D. No obstante, los efectos de observación son distintos, mientras que, los componentes son aproximadamente los mismos.

Así también, éstos resultan altamente personalizables e interactivos, de modo que, superan muchas de las limitaciones que presentan los robots físicos (Zhong et al., 2020).

Altin y Pedaste (2013) en su estudio que se centra en los robots virtuales utilizados como herramientas para enseñar (ciencia, tecnología, ingeniería, matemáticas) a niños en la escuela, mencionan que los robots virtuales se presentan como una opción económica, precisa, flexible y mejoran sus componentes en relación con los robots físicos.

También mencionan que éstos pueden ser de fácil manipulación en lo relacionado a su comportamiento y apariencia, por lo que, pueden ser mostrados en diferentes pantallas en forma simultánea, como es el caso de dispositivos móviles, ordenadores, entre otros (Hoffmann y Krämer, 2013).

En este sentido, Witherspoon et al., (2017), coinciden que la interacción de los niños mediante un computador con los robots virtuales, y su inclusión en el currículo, como se muestra en la Tabla 3.2.

Estos son beneficiosos en su nivel de aprendizaje, ya que presentan avances significativos en las puntuaciones previas y posteriores en relación a evaluaciones realizadas, con estudiantes que utilizaron un plan de estudios entornos virtuales de aprendizaje que facilitan la adquisición de competencias básicas de pensamiento lógico.

Unidad	Lecciones	Actividades	Temas tratados
Introducción a la programación	4	1	Requisitos del sistema y configuración tutorial sobre el funcionamiento de la plataforma.
Movimiento básico	6	16	Secuencias básicas de comando de movimiento hacia adelante. Secuencias básicas de comandos.
Sensores	10	19	Usos de sensores, ecuaciones con desigualdades lógica booleana. Función If - then declaraciones condicionales en programación.
Flujograma	10	18	Bucles repetidos e infinitos. If - else declaraciones condicionales en programación. Decisiones repetidas (if - else anidadas dentro de los bucles)

Tabla 3.2 Unidades curriculares (fuente: Witherspoon et al., 2017)

3.1.3.2 Scratch

Scratch fue desarrollado por la División de Educación del Instituto de Tecnología de Massachusets (MIT) con el fin de promover el conocimiento en ciencia, tecnología y otras áreas relacionadas en estudiantes de todo nivel educativo independientemente de su edad, antecedentes, o intereses (Tsur, 2017; Plaza et al., 2019).

Por lo que, se ha convertido en uno de los entornos de programación más utilizado por principiantes, específicamente por niños en contextos escolares (Benton et al., 2017).

Scratch se presenta como una herramienta tecnológica que permite programar historias interactivas, juegos, animaciones 2D interactivas, arte, simulaciones, anuncios, etc. (Papadakis et al., 2014; Feijoo y De la Rosa, 2016; Tsur, 2017; Moreno-León et al., 2017).

Esas creaciones se denominan proyectos (Barshay, 2014), considerados como un enfoque constructorista de la tecnología, en el que los niños construyen su propio conocimiento, a través de, proyectos personales en entornos abiertos, y sobre todo, en el que los estudiantes pueden tener más control sobre su proceso de aprendizaje (Tsur, 2017).

Según Resnick et al., (2009) un proyecto de Scratch consiste en un conjunto de actores que pueden presentar diferentes comportamientos, que son definidos, a través, comandos de lenguaje y actúan en un escenario o pantalla.

Así también, estos escenarios pueden ser personalizados con la inclusión de fotografías, voces, clips musicales, entre otros. El entorno que ofrece Scratch consiste en un piso muy bajo y un techo alto, además, cuenta con paredes bastante anchas a fin de que los usuarios puedan crear diferentes tipos de escenarios (Papadakis et al., 2017).

Asimismo, la interfaz que presenta Scratch es similar a un rompecabezas (Buitrago Flórez et al., 2017; Evangelopoulou y Xinogalos, 2018; Papadakis y Orfanakis, 2018) con el fin de evitar o reducir al mínimo los problemas de sintaxis, y, que éstos puedan enfocarse en construcciones programáticas fundamentales (condiciones, bucles, variables).

Al respecto, Resnick et al., (2009) consideran que este entorno ofrece a los estudiantes la oportunidad de aprender conceptos básicos; y, además brinda un espacio para desarrollar el pensamiento creativo, el razonamiento sistemático (Topalli y Cagiltay, 2018), y, el trabajo en equipo.

Al respecto, Brennan y Resnick (2012) coinciden en que los proyectos generados a través de Scratch incluyen tres dimensiones clave: (1) conceptos computacionales, (2) prácticas computacionales; y, (3) perspectivas computacionales.

Por otro lado, Vasek (2012) menciona que, los programas en Scratch se encuentran compuestos de scripts que, a su vez, controlan los sprites que luego son mostrados en un escenario.

Los estudiantes escriben los programas con el objetivo de manipular sprites. Los scripts se crean arrastrando y soltando bloques, que representan componentes del programa, tales como, expresiones, condiciones, declaraciones y variables.

Además, un sprite puede estar compuesto por varios scripts y éstos pueden ser ejecutados al mismo tiempo (Meerbaum-Salant et al., 2013). Puede verse una imagen de Scratch en la Figura 3.2.

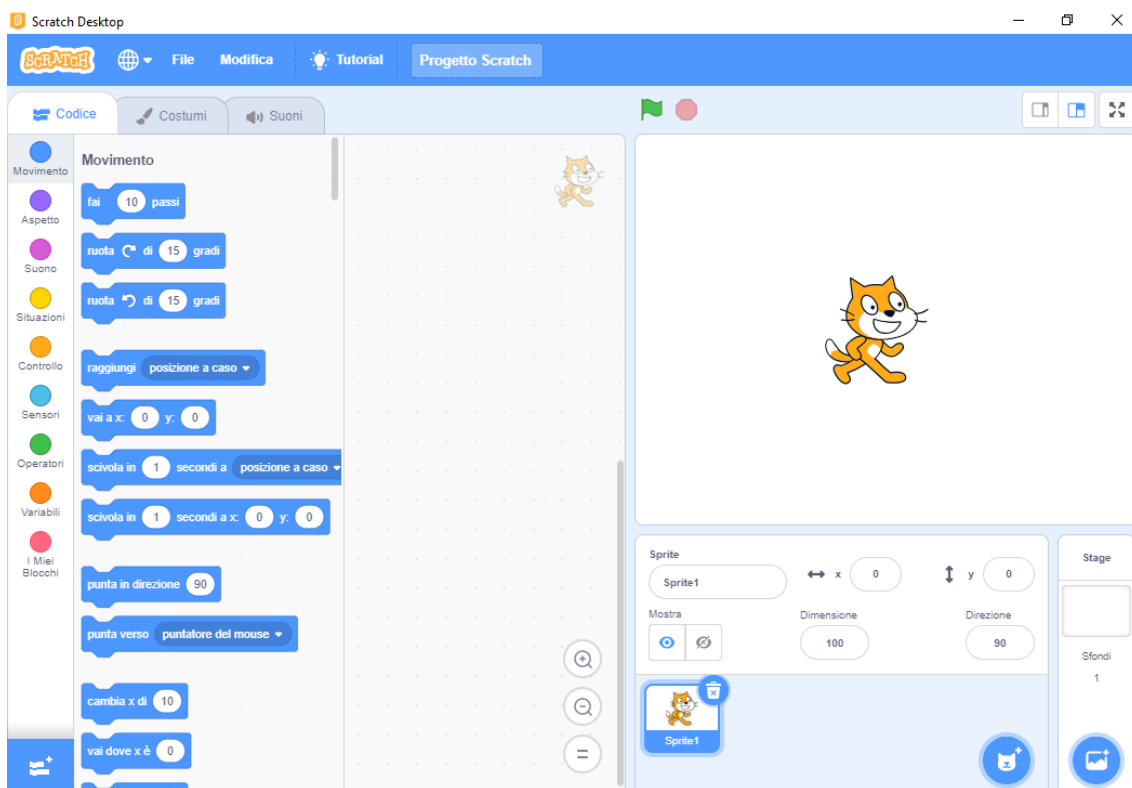


Figura 3.2 Interfaz de Scratch (fuente: Resnick et al., 2009)

3.1.3.3 Code.org

Code.org es una organización que trabaja sin fines de lucro con el fin de ampliar el acceso a la informática. Esta organización tiene como visión principal que los estudiantes de las escuelas tengan la oportunidad de aprender ciencias de la computación, del mismo modo que pueden aprender las demás asignaturas (Bučková y Dostál, 2017).

A través de la oferta de todo el contenido de los planes de estudio y cursos de forma gratuita, sin que exista la necesidad de que los usuarios generen cuentas o mantengan alguna asociación a cuentas de otras plataformas (Google, Facebook, Microsoft, etc.) (Code.org, 2019).

Así también, Code.org es considerado como un entorno de programación dirigido específicamente para niños mayores de 4 años. Esta plataforma presenta un conjunto amplio de recursos y herramientas educativas que pueden ser ejecutados en otros tipos de

plataformas, tales como teléfonos inteligentes y tabletas, lo que hace que ésta sea muy flexible y de fácil uso.

Del mismo modo, otro aspecto que hace interesante a Code.org, es que está construida como una aplicación Web que se ejecuta en un entorno de navegador. Es decir, que los estudiantes pueden realizar sus trabajos, sin que sea necesario, preinstalar o configurar los dispositivos.

Por lo que, el aprendizaje y el desarrollo del pensamiento lógico se desarrolla a través de un entorno gráfico, en el cual los estudiantes construyen sus propios algoritmos, arrastrando y soltando bloques, de modo que van resolviendo cada desafío que se presente (Code.org, 2019). Puede verse una imagen de Code.org en la Figura 3.3.

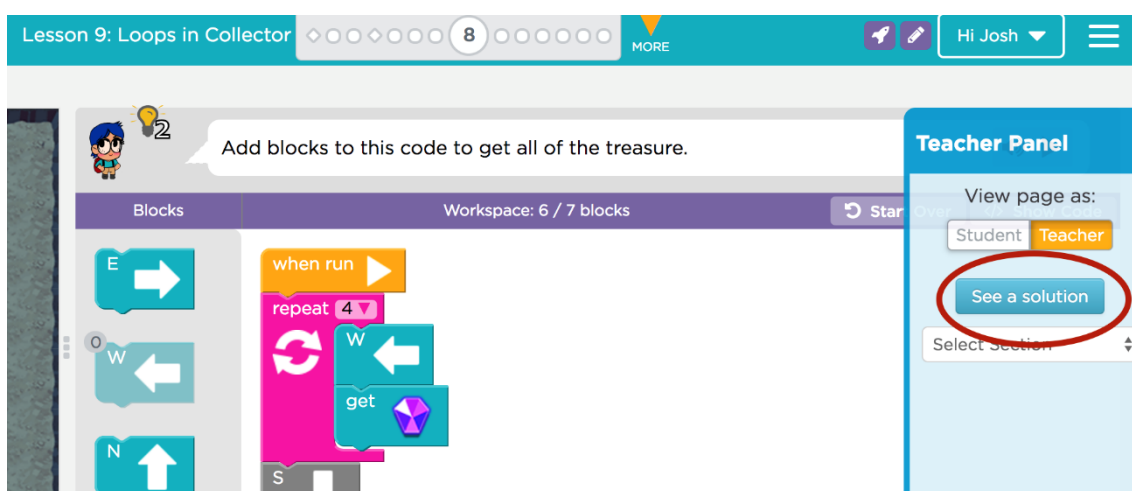


Figura 3.3 Interfaz de Code.org (fuente: Code.org, 2019)

3.1.3.4 PRIMM

Sentance et al., (2019) desarrollaron un marco para desarrollar la programación llamado PRIMM que en sus siglas en inglés significa Predict-Run-Investigate-Modify-Make puede verse en Figura 3.4.

De acuerdo a los autores, PRIMM es un método de enseñanza que permite contrarrestar el problema más conocido como el de los principiantes. Este problema consiste en que éstos escriben programas antes de ser capaces de leerlos, por lo que, centra su atención en que los estudiantes dialoguen sobre cómo y por qué funcionan los programas, previo a empezar a dar inicio la edición y redacción de sus propios programas.

En el estudio, conto con un total de 493 estudiantes de 11 a 14 años, utilizando un enfoque de métodos mixtos, se recopiló información de una prueba de referencia y exposición de 12 lecciones de programación utilizando el enfoque PRIMM. Posteriormente se realizó una prueba, y entrevistas con los docentes. Se obtuvo como resultado que, en la prueba posterior, los alumnos del grupo que utilizó PRIMM, obtuvieron mejores resultados que el grupo de control. Adicionalmente en las entrevistas, los docentes informaron que el uso

de PRIMM, contribuyo a enseñar de manera efectiva en clases de habilidades mixtas, lo que permitió que todos los estudiantes progresaran.

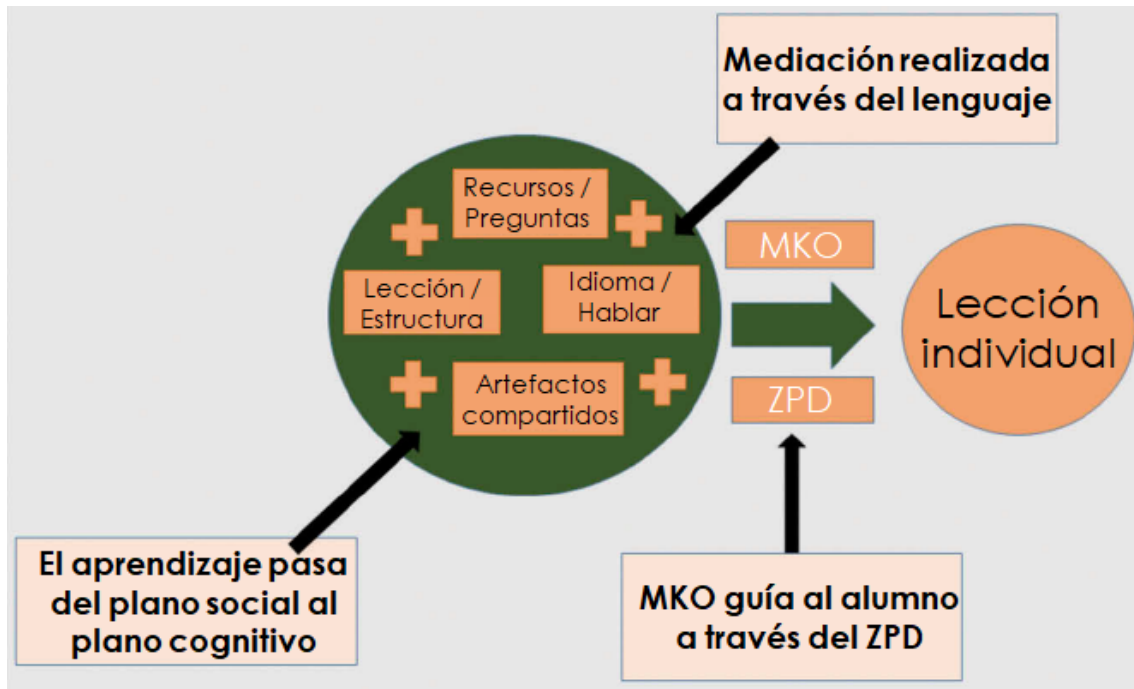


Figura 3.4 Framework para el diseño de lección de programación (fuente: Sentance et al., 2019)

3.1.3.5 Modelo de Bloques

Schulte (2008) menciona un modelo conocido como de Bloques que permite que el estudiante, visualice de mejor manera la programación. Este modelo presenta dos tipos de dimensiones. Por un lado, una horizontal de función que se encuentra dividida en superficie de texto y ejecución del programa; y, una vertical que está estructurado por niveles de átomos, bloques, relaciones y macroestructura. A este respecto, Sentance et al., (2019) consideran que la función se describe como los objetivos del programa.

A través del uso modelo de bloques, autores como Schutle et al., (2010) sugieren dos secuencias de enseñanza y aprendizaje: micro secuencias, que usualmente se enfocan en una sola actividad para implementar un algoritmo; y, macro secuencias que están compuestas por varias actividades.

En este sentido, los resultados de la experimentación, Schulte (2008) lo presenta en tres fases: datos generales (tiempo empleado, miembros del grupo, etc.), desarrollo de ideas y refinamiento de una idea.

Concluyendo que el modelo de bloques es simple, constructivo y comunicativo cuando lo usan los novatos que discuten cómo diseñar una lección sobre un programa determinado. Por otra parte, el modelo de bloques está restringido en su enfoque, lo que

parece razonable, y apoya (por medio de la simplicidad) la aplicabilidad del modelo. Este modelo puede verse en la Tabla 3.3.

Estructura macro	(Comprender la estructura general del texto del programa)	Entender el "algoritmo" del programa	Comprender el objetivo / propósito del programa (en su contexto)
Relaciones	Referencias entre bloques, p. Ej: llamadas a métodos, creación de objetos, acceso a datos.	Secuencia de llamadas de método "diagramas de secuencia de objetos"	Comprender cómo se relacionan los objetivos específicos con el objetivo general, cómo se logra la función mediante subfunciones
Bloques	'Regiones de intereses' (ROI) que construyen sintáctica o semánticamente una unidad	Operación de un bloque, un método o un ROI (como secuencia de declaraciones)	Función de un bloque, tal vez visto como subjetivo
	Elementos del lenguaje Operación de una declaración	Función de un enunciado.	Objetivo solo comprensible en contexto
	Superficie de texto	Ejecución del programa (flujo de datos y flujo de control)	Funciones (como medio o como propósito), metas del programa
Dualidad	"Estructura"		"Función"

Tabla 3.3 Modelo por Bloques (fuente: Schulte, 2008)

3.1.3.6 Lectura de código vs escritura de código

Lister et al., (2004) y Lister et al., (2009) ponen de manifiesto la importancia que tiene el leer códigos y poder rastrear lo que el estudiante hace antes de escribir un código nuevo.

Los autores al comparar las habilidades que implica el rastreo con la escritura de un código pusieron en evidencia que los principiantes necesitan una precisión del código de rastreo del 50% antes de que éste se encuentre en capacidad de escribir un código de forma independiente.

El proceso de aprendizaje de programación es secuencial y acumulativo, por lo que es necesario que los estudiantes accedan al conocimiento acumulado, para tener una idea más clara del panorama de trabajo. Por lo tanto, los estudiantes principiantes deben realizar tareas más pequeñas con elementos únicos (Teague y Lister, 2014). En este sentido, Busjahn y Schulte (2013) coinciden en la importancia que tiene el utilizar la escritura del código para entender significados.

En la década de 1980, los autores Van Merriënboer y Krammer (1987) plantearon un enfoque que permite enseñar programación basado en la lectura de códigos, considerando a éste como más efectivo que los llamados *expert* y *spiral*.

La compatibilidad entre estrategias de instrucción y tácticas de instrucción se detalla en la Tabla 3.4. Tradicionalmente se enfocaban en la escritura de códigos de arriba hacia abajo y en el diseño de programas incrementales respectivamente (Sentanc et al., 2019).

El enfoque de lectura puntúa más alto iguala con los otros grupos de estrategias, en cinco de seis tácticas de instrucción. Si todas las tácticas tienen el mismo peso, se puede concluir que el enfoque de lectura es la mejor estrategia a seguir en el diseño instruccional de cursos de introducción a la programación.

Destreza	Experto	Espiral	Lectura
	Instrucción declarativa		
Modelos de computadora	-	+	+
Planes de programación	+	+	+
Diagramas de diseño	+	-	-/+
	Instrucción procedimental		
Ejercicios resueltos	-/+	-/+	+
Habilidades básicas	-/+	+	+
Variación de tareas	-	-	+

Tabla 3.4. Compatibilidad entre estrategias de instrucción y tácticas de instrucción (fuente: Sentanc et al., 2019)

3.1.3.7 Usar-modificar-crear

Como se describe en la Figura 3.4, un enfoque muy utilizado en la enseñanza de programación es el denominado Usar-Modificar y Crear (UMC). Este enfoque tiene el objetivo apoyar la progresión en el aprendizaje de programas (Lee et al., 2011). A través de este enfoque, que se describe en la Figura 3.5, los estudiantes tienen la oportunidad de iniciar su aprendizaje utilizando, en primer lugar, un programa elaborado por otra persona

para después pasar a crear el su propio programa; es decir, pueden utilizar el trabajo existente y modificarlo para que se convierta en “suyo” (Sentance et al., 2019).

En las conclusiones de su investigación Lee et al., (2011) investigaron en la enseñanza de la programación en secundaria concluyendo que estos estudiantes pueden utilizar la abstracción, la automatización y el análisis para crear productos originales, cuando cuentan con un entorno robusto de aprendizaje y maestros capacitados.

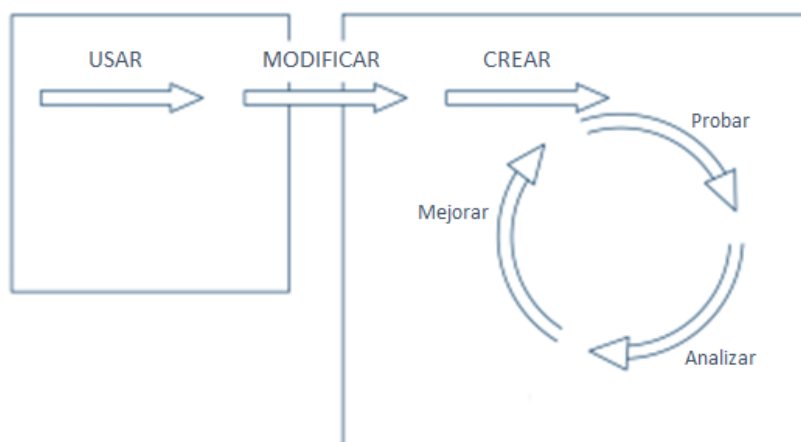


Figura 3.5 Progresión del aprendizaje, Usar-Modificar-Crear (fuente: Lee et al., 2011)

3.2 Agentes conversacionales Pedagógicos

3.2.1 Definición y ejemplos

Bickmore y Cassell (2001) define los Agentes Conversacionales Pedagógicos (ACP) como agentes virtuales autónomos y animados que interactúan con los usuarios a través de un diálogo en tiempo real.

En esta conversación el ACP reconoce tanto el habla como los gestos y hace de ésta una interacción similar a la de dos personas (Luerssen y Powers, 2011; Jacques y Vicari, 2007).

De esa manera, los usuarios no están limitados a un conjunto de palabras clave, u opciones de un menú, sino que pueden ingresar oraciones en lenguaje natural que son entendidas por la computadora, la cual, a su vez, genera más oraciones en lenguaje natural (Kelley, 1984; Zhou, 2007).

En tal sentido, Picard et al., (2004) coinciden en que la conciencia emocional; es decir, el estado afectivo de los usuarios es fundamental para que el proceso sea más productivo. Así también, Danilava et al., (2012) enfatizan en que la interacción a largo plazo con un ACP dependerá de la motivación que presente el usuario para trabajar con éste.

Por lo tanto, la conversación que mantengan estos dos, estará influenciada por la confianza, la simpatía, el vínculo emocional positivo y la utilidad.

El crecimiento de los ACP ha tenido gran impulso debido a la convergencia existente con tecnologías habilitadoras, como la Web que surgió como un canal de comunicación universal.

Al respecto, los ACP que se encuentran basados en la Web son sistemas escalables que hacen uso del Internet con el fin de ofrecer de forma simultánea servicios de diálogo con un número grande de usuarios.

El surgimiento de los ACP no se trata de un hecho relativamente nuevo, ya que su primera aparición se remonta a los años sesenta, con el agente llamado Eliza, que simulaba a un psicoanalista (Weizenbaum, 1966).

Posteriormente, en la década de 1990 se realizó el lanzamiento de la máquina de inferencia A.L.I.C.E. (Artificial Linguistic Internet Computer Entity, que fue un proyecto de código abierto, se promueve como Lenguaje de Marcado de Inteligencia Artificial (AIML) (Wallace, 1995).

Así también, en lo que respecta a ACP utilizados para fines educativos, pueden asumir varios roles, dependiendo del objetivo pedagógico que persigan: pueden ser docentes, estudiantes, acompañantes, tutores, coaches, etc. (Pérez-Marín, 2014).

Se pueden nombrar a Doroty, que instruye a los usuarios en la administración de redes informáticas (Leonhardt et al., 2005); Blaze que fue creado con el fin de mejorar las habilidades cognitivas de los estudiantes en cuanto a lo que se refiere a problemas matemáticos (Aguiar et al., 2014). Geranium que fue utilizado como una herramienta útil para la enseñanza del ecosistema urbano (Griol y Callejas, 2013); y, Mentor chat que fue desarrollado para fomentar el aprendizaje colaborativo de idiomas (Tegos et al., 2014).

En cuanto al uso de los ACP en el ámbito educativo, son diversos los beneficios que éstos traen consigo, por ejemplo, la utilización de los ACP promueve procesos de enseñanza aprendizaje eficientes.

En los que los estudiantes perciben la interacción real y natural (Tamayo-Moreno y Pérez-Marín, 2017), la existencia de una mayor interactividad, sociabilidad e incremento del uso de los entornos virtuales del aprendizaje (Kang et al., 2012; Abushawar y Atwell, 2015). Así también, facilitan el trabajo a los estudiantes, permitiendo la recuperación de información de una forma más efectiva, a través, de la utilización de preguntas frecuentes (Ghose y Barua, 2013).

Del mismo modo, Xie y Luo (2017) mencionan que, por medio del ACP, los individuos pueden mejorar sus habilidades y fomentar la finalización de las tareas a través de la motivación para comunicación sostenida en el aprendizaje (Fryer et al., 2017).

A modo de ejemplo, se detallan algunos agentes conversacionales:

3.2.1.1 El insecto Herman

En el año 1997, el agente Herman que tenía forma de gusano fue utilizado para el elaborar el diseño de una planta (Lester et al., 1997). Este agente presenta una serie de combinaciones de gestos y consejos verbales.

Con la aplicación de éste, se observaron resultados positivos en los grupos aplicados. Sin embargo, no se ha seguido utilizando este agente como una herramienta educativa. Puede verse una imagen de Herman en la Figura 3.6.



Figura 3.6 Interfaz del agente Herman (fuente: Lester et al., 1997)

3.2.1.2 Stev

En 1999, apareció el agente llamado Steve, éste presentaba una forma humana que interactuaba con los estudiantes, representados con avatares en un mundo virtual en 3D (Rickel y Johnson, 1999).

Este agente entabló un diálogo de conversación mixta con los estudiantes, es decir, cualquiera de los dos podía iniciar el diálogo, con el objetivo de ayudar al estudiante a completar las tareas. Puede verse la imagen de Steve en la Figura 3.7.



Figura 3.7 Interfaz del agente Steve (fuente: Rickel y Johnson, 1999)

3.2.1.3 Guilly

En el año 2002, aparece Guilly que tenía forma de gusano, como verse en la Figura 3.8, su objetivo principal era enseñar a los niños sobre el reciclaje (Nunes et al., 2002).

El agente interactúa con los estudiantes de forma verbal y los ayuda completar las tareas encomendadas, además, utiliza cambios en el entorno con el fin de mostrarles los beneficios del reciclaje.

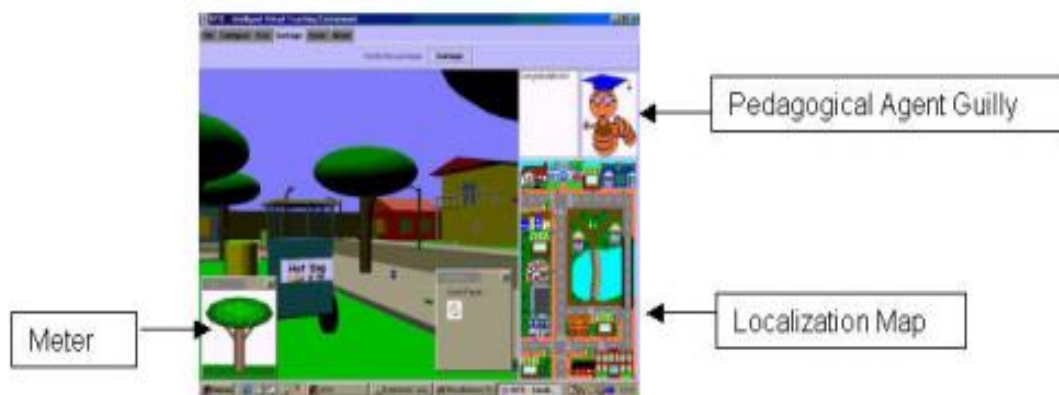


Figura 3.8 Interfaz del agente Guilly (fuente: Nunes et al., 2002)

3.2.1.4 Sam

En 2003, Sam aparece como un agente en forma de niño (Ryokai, 2003). A través de varios experimentos con este agente, se mostraron los beneficios de estas interacciones, en el sentido de que, los estudiantes están en la capacidad de desarrollar sus habilidades de narración con la ayuda de Sam. Puede verse una imagen del agente en la Figura 3.9.



Figura 3.9 Interfaz del agente Sam (fuente: Ryokai, 2003)

3.2.1.5 Autotutor

Autotutor fue diseñado en 2005, su interfaz se puede ver en la Figura 3.10, y se trataba de un agente con forma humana que utiliza el análisis semántico latente (Landauer et al., 1998), y, una base de datos de guiones circulares con preguntas y diálogos, con el objetivo de promover la conversación con los alumnos (Graesser et al., 2008).

El objetivo principal de este agente fue el de mejorar el proceso del diálogo a través de las preguntas. Los autores consideran que el funcionamiento adecuado de este agente venía del nivel de conocimiento compartido tanto de Autotutor y el estudiante.

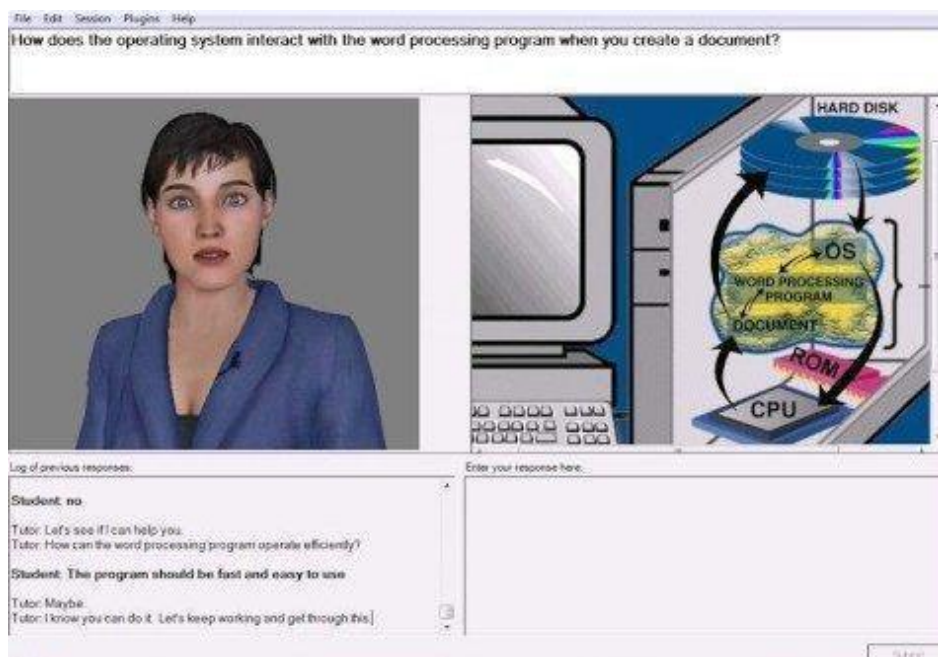


Figura 3.10 Interfaz del agente Autotutor (fuente: Landauer et al., 1998)

3.2.1.6 Sara

Sara es un agente conversacional que fue diseñado en función de tres principios: dos provenientes de la teoría del andamio (*scaffolding*) y uno de la teoría cognitiva del aprendizaje.

En cuanto al primer principio, se incluye un diálogo y sub-diálogo en la interacción, es decir, se formula siguiendo la estrategia de andamiaje denominada líneas de razonamiento dirigidas.

En otras palabras, resulta similar a lo propuesto por los docentes cuando formulan un grupo de preguntas directrices a los estudiantes con el objetivo de profundizar en un tema específico (Mills et al., 2004).

A este respecto, Sara realiza preguntas específicas a los estudiantes, después de un tiempo de haber iniciado el video, en este caso los alumnos pueden presentar tres tipos de respuestas: correctas, incorrectas y no saben la respuesta.

En este momento, si Sara detecta una respuesta correcta pasa al siguiente cuadro de diálogo, caso contrario procede a abrir un cuadro de sub-diálogo en el cual proporciona información relevante para reforzar el conocimiento del estudiante (Albacete et al., 2018).

En un segundo principio, Sara incluye métodos de diagnóstico con el fin de detectar el grado de conocimiento que presentan los alumnos a través de la aplicación de distintas preguntas en las que las respuestas pueden ser: correctas, incorrectas o el estudiante no la sabe (Graesser et al., 2017).

Finalmente, el tercer principio se encuentra basado en la teoría cognitiva del aprendizaje, en la que se debe tomar en cuenta varios canales de comunicación, ya sean de voz o de texto.

En este sentido, se debe dar el mismo peso a la utilización tanto del texto como de la voz, es decir, Sara inicia una conversación con el estudiante brindándole todas las indicaciones y consultándole si todo está claro, y, además, este mismo mensaje de voz se encuentra en forma de texto en un chat, con el fin de que el mensaje quede comprendido por completo.

3.2.1.7 MentorChat

La interfaz de MentorChat puede verse en la Figura 3.11. Es un prototipo de sistema de apoyo de diálogo (Tegos et al., 2015). Éste se asemeja a una aplicación de mensajería instantánea, a través de la cual solicita a los estudiantes que elaboren una o más tareas de aprendizaje en línea.

MentorChat presenta una interfaz de fácil administración, ya que permite a los docentes registrar a los estudiantes que van a participar y asignar grupos de trabajo, con el fin de monitorearlos y crear actividades de aprendizaje colaborativo.

Una de las actividades puede incluir tareas basadas en el diálogo, solicitando a los alumnos que discutan un tema específico y envíen sus respuestas, fomentando de esta manera el aprendizaje abierto (Tegos y Demetriadis, 2017).

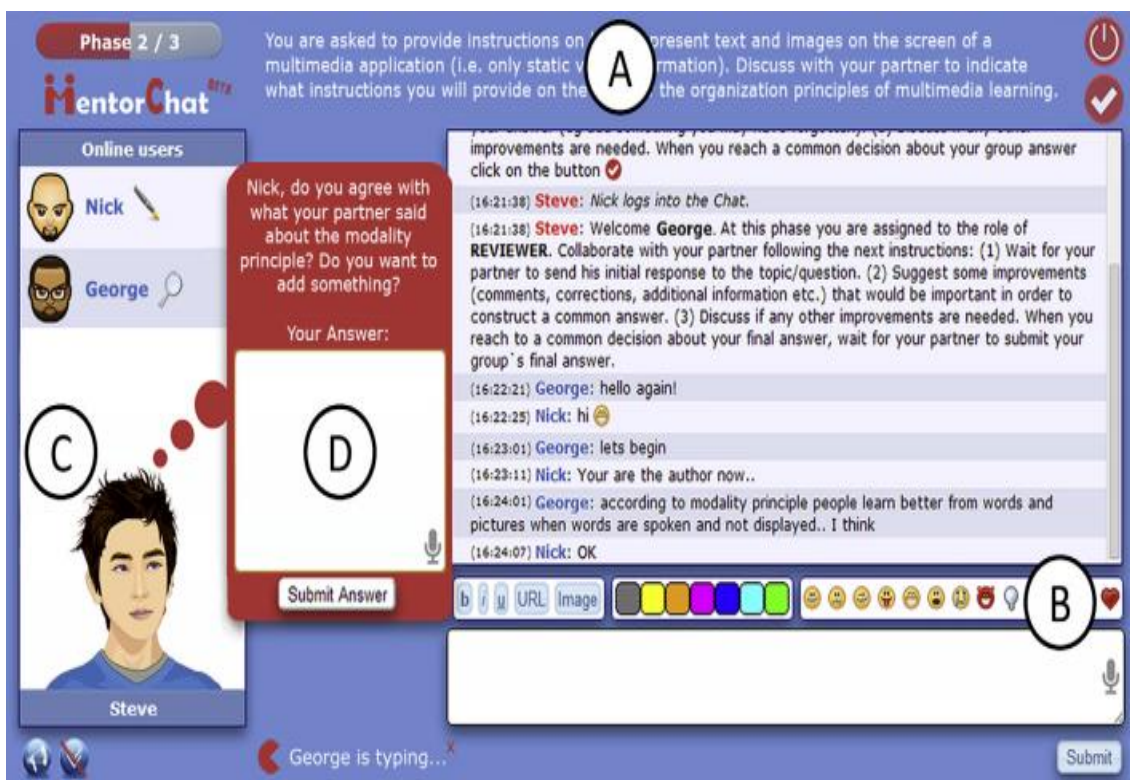


Figura 3.11 Interfaz de MoneterChat (fuente: Tegos et al., 2015)

3.2.1.8 Prof. Watson

Prof. Watson (Yeves-Martínez y Pérez-Marín, 2019) se propone como un agente que quiere dialogar con estudiantes de Educación Primaria para enseñarles programación dialogando con ellos según la metodología MECOPROG.

Sin embargo, fue nuestra creencia de que mediante el uso de agentes conversacionales pedagógicos que dialogan con los estudiantes, han pensar cómo resolver problemas dados y escribir el código para resolverlos.

En particular, se aplicó la metodología MECOPROG para diseñar el diálogo estudiante-agente en el Prof. Watson. Puede verse una imagen del diálogo del agente Prof. Watson en la Figura 3.12.

Para probar Dr. Watson se llevó a cabo un experimento con 19 estudiantes (de 11 a 12 años) que demostró la viabilidad del enfoque. Arrojando algo de luz sobre procedimientos alternativos para enseñar programación en Educación Primaria Yeves-Martínez y Pérez-Marín, D. (2019).

Hello! I'm Professor Watson! What's your name?	
Hi, Pablo! What do you want to do today?	I'm Pablo Yeves
Great! I'll show you how to create a variable. A variable consists of a name and a value. To create a variable, you can write name = value . Try to create a variable named apples with the value you like.	I'd like to do the tutorial, please
Right! Now we have a variable apples with a value of 34 , as you can see on the right column. Let's practice now functions. A function is like an action. Do you want to learn how to do them?	apples = 34
Ok, ask me to write the value of the variable apples . You can do it with write(apples) . Now it's your turn.	sure!
34 Great! Did you know that if you ask me to write between quotation marks "like this" you don't need to create a variable? For example, try to ask me to write the word you like without creating a variable.	write(apples)
Oh, you've forgot the quotation marks...	write(apples)
"apples" That's it! Let's go to conditionals now. Do you know what they are?	Write("apples")
That's easy, Pablo. A conditional is a structure like if... then... Do you want to know a bit more about conditionals?	no...

Figura 3.12 Conversación entre un estudiante y el profesor Watson (fuente: Yeves-Martínez y Pérez-Marín, 2019)

3.2.2 Taxonomía

En cuanto a la taxonomía que se utiliza para clasificar las principales características de los Agentes Pedagógicos Conversacionales APC. Pérez-Marín (2011) propuso una taxonomía que se puede aplicar, con el fin de seleccionar el agente que sea más apropiado, de acuerdo a los requisitos y objetivos que se persiguen en cada curso.

La taxonomía propuesta se encuentra estructurada por diez criterios principales (Pérez Marín, 2014):

- Rol y actitud: En este criterio se tiene que decidir, si el agente va a trabajar como docente, alumno o acompañante. Entre las posibilidades se mencionan:
 - Instructor/Lector/Tutor: Entrenador, Autoritario
 - Estudiante: Individual, Colaborativo
 - Compañero: Asistente emocional, alborotador
- Modalidad de interacción: Se presentan las siguientes opciones:
 - El agente puede iniciar la conversación
 - El alumno inicia la conversación

- Mixta, es decir, tanto el agente como el estudiante pueden iniciar la conversación.
- Tipo de animación:
 - No presenta ninguna
 - Sólo gráficos
 - Presenta una animación con gráficos y sonido
- Posibilidades afectivas:
 - Ninguna
 - Apoyo emocional
 - Empatía
- Tipo de personaje virtual
 - Puede presentar forma humana sin ninguna animación
 - Presenta forma humana con animación incluida
 - Sin forma humana
 - Forma humana o no humana sin animación
 - Forma humana o no humana con animación
- Evolución adaptativa: Se refiere a la existencia de la posibilidad de que el agente se pueda adaptar al comportamiento del estudiante, y aprenda a evolucionar para mejorar futuras interacciones:
 - Ninguna posibilidad
 - Posibilidades de adaptación
 - Posibilidades de evolución
 - Posibilidades tanto de adaptación como de evolución
- La ubicuidad: Se encuentra relacionada con la posibilidad de que el agente no sólo se utilice en el computador, sino también, en otros dispositivos móviles:
 - Puede ser utilizado sólo en computadores
 - Se utiliza en dispositivos móviles (smartphones y/o PDAs)
- El dominio: Se trata del conocimiento o competencias en que los agentes pueden involucrar al estudiante:
 - General

- Específico
- El idioma:
 - Europeo
 - No europeo
- Edad: Este aspecto depende directamente de los estudiantes a los cuales está dirigido el agente (Educación Infantil, Educación Primaria, etc.)
 - Niños
 - Adultos
 - Niños y adultos

3.2.3 Metodología de diseño (MEDIE) y co-diseño (IPO)

3.2.3.1 MEDIE

La Metodología para el Diseño, Integración y Evaluación de Agentes Conversacionales Pedagógicos conocida como MEDIE, puede verse en la Figura 3.13, consta de varias fases presentando un ciclo iterativo e incremental (Tamayo-Moreno 2017).

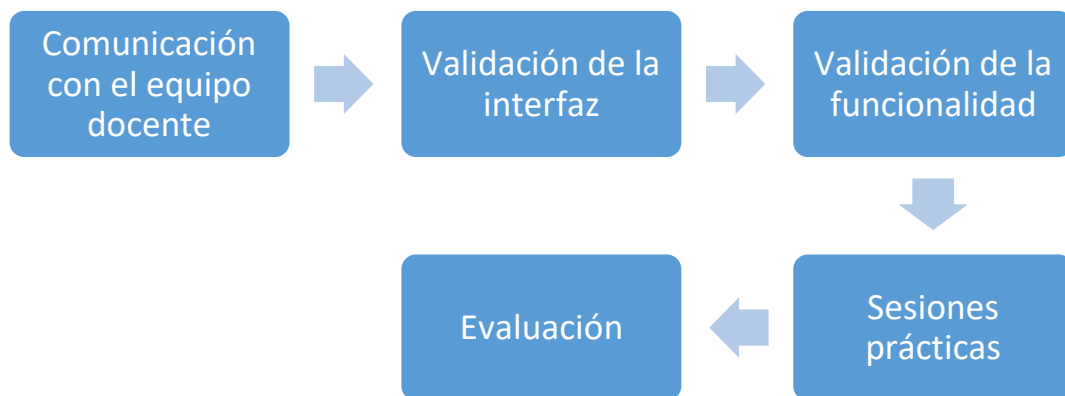


Figura 3.13 MEDIE ciclo iterativo e incremental (fuente: Tamayo-Moreno y Pérez-Marín 2017)

- En la primera fase, relacionada con la comunicación con el equipo docente, se lleva a cabo un análisis del contexto en el que se desarrollará el agente conversacional pedagógico, y qué requisitos debe cumplir, por lo que es necesario la participación de todos los especialistas de las diversas áreas involucradas en el proceso.

- En la siguiente etapa de validación de la interfaz, se procede con la aprobación y del diseño del agente y la validación de la interfaz.
- Posteriormente, en la fase de validación de la funcionalidad se determina y valida la funcionalidad del agente diseñado, por lo que es de importancia que en esta etapa se presenten diferentes diagramas que presente una simulación de lo que brindará el agente. Así también, se deben agregar y detallar diferentes aspectos necesarios, que permitan tener una idea del funcionamiento del agente.
- En lo relacionado a las sesiones prácticas, es de relevancia que se realicen el mayor número de sesiones posible, con el fin último de que el agente diseñado tenga un tiempo de exposición considerable, y, se pueda obtener evidencias significativas del diseño.
- En la etapa final, se realizará una evaluación a través de la aplicación de los métodos IPO tales como: métodos de inspección, de indagación y de test.

3.2.3.1 Co-diseño

El Co-diseño es considerado como la creatividad colectiva dentro de un proceso de diseño, por lo tanto, éste es un recurso específico de la co-creación. Así también el co-diseño se utiliza en un sentido más amplio para destacar la creatividad de los diseñadores y de personas no tan capacitadas en el diseño que trabajan de manera conjunta en el desarrollo del diseño (Sanders, 2008).

La práctica del diseño ha experimentado cambios constantes, de acuerdo a los diferentes panoramas que se han ido presentando. El enfoque centrado en el usuario tuvo su origen en los años setenta, no obstante, se generalizó en la década de 1990. Este enfoque mostró que era de gran utilidad, en especial en el diseño y en el desarrollo de productos de consumo (Sanders, 1992).

Sin embargo, en la actualidad el enfoque mencionado no está en la capacidad de abordar la complejidad de los desafíos a los cuales se enfrenta, dado que, es necesario diseñar en función de las experiencias futuras de personas, comunidades, y culturas, que poseen mayor conectividad e información (Sanders, 2008).

En este contexto, han surgido nuevas disciplinas relacionadas con el diseño, tal es el caso del diseño de interacción, que fue introducido a finales de los años ochenta por Moggridge y Verplank (2007).

Así también, el diseño de servicios que comenzó a tener importancia a partir del año 2006 en la primera conferencia de diseño. Por otra parte, está el diseño de transformación que se introdujo en el 2006 a través de un libro blanco con el mismo nombre y que fue publicado por el Consejo de Diseño del Reino Unido.

Este tipo de disciplina se basa en prácticas participativas del diseño combinadas con la participación de métodos centrados en el usuario. Además, utiliza las habilidades del diseño tradicional con el fin de abordar problemas sociales y económicos y utiliza el proceso del diseño como un medio que permite que varias disciplinas y partes interesadas colaboren (Burns et al., 2006).

Por otro lado, el diseño de interacción es definido por Preece et al., (2002) como diseñar productos de carácter interactivo con el fin de ayudar a los usuarios en su vida diaria o laboral; es decir, generar prácticas en el usuario que mejoren la forma de trabajo colaborativo.

Además, éste aborda una gama más amplia de aspectos relacionados con temas o paradigmas más tradicionales. En lo que respecta a la usabilidad, se destacan dos aspectos en la experiencia del usuario, por una parte, la efectividad, seguridad, eficiencia, utilidad, facilidad de aprendizaje y memorización; y, por otra, señala aspectos relacionados con la satisfacción y la diversión (Sanders, 2008).

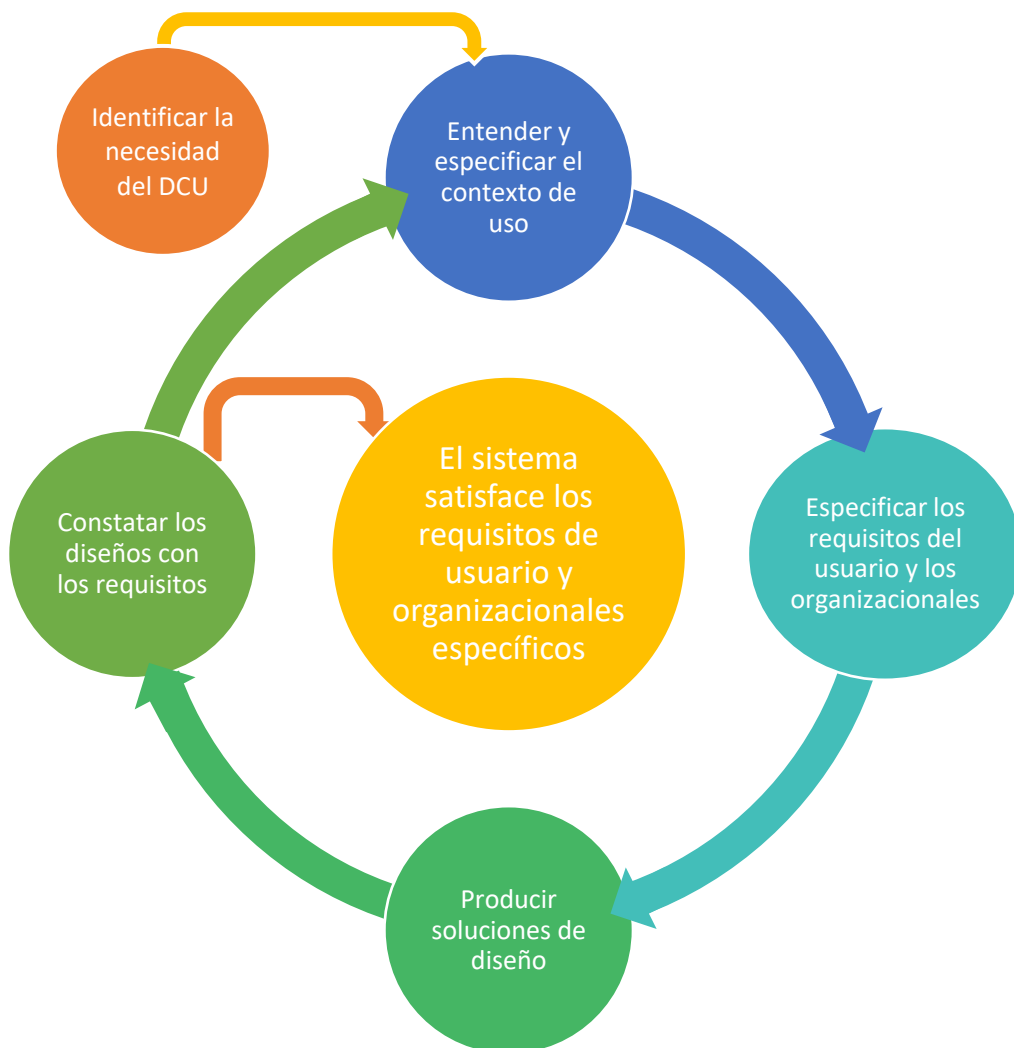


Figura 3.14 Proceso iterativo del (DCU), según ISO 1340

A modo general, las disciplinas tradicionales se centraron en los productos o tecnologías, en la que se ponían en práctica las habilidades requeridas para generar de una forma experta identidades de marca, productos, etc., mientras que las prácticas de diseño emergentes se centraron en las necesidades de los usuarios o de la sociedad.

El diseño debe basarse en el usuario, y en este sentido, tiene una serie de principios:

1. El usuario tiene que poder controlar la situación.
2. Es necesario un planteamiento directo.
3. La consistencia es fundamental.
4. Posibilitar la recuperación de los errores.
5. Retroalimentación adecuada por el sistema.
6. Cuidar la estética.
7. El diseño tiene que ser simple.
8. Es fundamental seguir una rigurosa metodología de diseño.
9. El equipo de diseño debe ser equilibrado.

3.2.4 Compañeros de aprendizaje para el aprendizaje de programación

Aprender a programar puede inducir cambios en la manera de pensar de las personas (Papert, 1980). El pensamiento lógico puede ser útil no solo para estudiantes o profesionales de Ciencias de la Computación, sino también para cualquier persona (Wing, 2006).

Esta es la razón por la cual la enseñanza de programación se han implementado en la Educación Primaria en varios países de todo el mundo (Heintz et al., 2016).

En años recientes, los agentes conversacionales pedagógicos se han convertido en excelentes herramientas que mejoran el proceso de enseñanza-aprendizaje (Schroeder et al., 2013). En el sentido de que, éstos se relacionan con los estudiantes tanto a nivel social como cognitivo, y, permiten a la vez, trabajar en función de las emociones, creencias y actitudes (Kim y Baylor, 2006).

En este sentido, Pezzullo et al., (2017) mencionan que los agentes pedagógicos se caracterizan por ser compañeros de aprendizaje virtual que comparte la experiencia de aprendizaje con los estudiantes y asume la personalidad de un compañero con conocimiento.

Así también, Kim y Baylor (2006) coinciden en que estos compañeros de aprendizaje representan un nexo con los estudiantes, adaptándose en muchos de los casos, a las necesidades afectivas y sociales de éstos, con el fin de equilibrar y apoyar en el proceso.

En la actualidad, el uso de las tecnologías de la información se ha convertido en una herramienta importante para el desarrollo del proceso de enseñanza y aprendizaje, por lo que en muchos de los países se han destinado recursos necesarios para el desarrollo de programas educativos que se encuentran orientados a la enseñanza de la programación.

Por ejemplo, en la próxima década, países como Estados Unidos requerirán de un mayor número de profesionales en áreas de informática y afines, ya que se considera que esta rama es de gran aporte para la generación de innovación y desarrollo económico.

Por lo tanto, el ámbito educativo enfrenta un reto trascendental, al ayudar a los estudiantes a fomentar y desarrollar el interés en el aprendizaje de programación y comprensiones específicas, mitigando el temor a las equivocaciones o dificultades del entorno, a través de la aplicación de técnicas o métodos acertados para este tipo de aprendizaje (Qian y Lehman, 2017).

En este contexto, la educación en programación busca involucrar, motivar y guiar a los estudiantes en el proceso, a través de la utilización de agentes conversacionales que dan soporte y ayuda a los docentes. Este agente se presenta como un personaje virtual que puede tener una forma humana, animal o de robot y que es capaz de interactuar con la utilización de la voz, texto, gestos, o una combinación de éstos.

Al respecto de este último, los docentes utilizan este tipo de herramientas con el fin de dar un acompañamiento virtual a los estudiantes en el desarrollo del proceso de aprendizaje, ya que estos agentes están diseñados con características específicas que pueden adaptarse a las necesidades individuales de los alumnos, con el fin de potenciar sus habilidades (Finch y Edwards, 2020).

Por otro lado, a pesar de que pueda existir un excelente aporte por parte de los agentes en el desarrollo del proceso, el problema surge en la automatización de herramientas de calificación, ya que no se evalúa el proceso, esfuerzo o trabajo del estudiante, sino, más bien es una calificación del resultado final (Edwards y Li, 2016). Por lo que se hace necesario la existencia de una retroalimentación que permita a los estudiantes mejorar, mientras desarrollan sus tareas, reconociendo el esfuerzo invertido y los logros alcanzados, en lugar de únicamente señalar todos los errores (Finch y Edwards, 2020).

En este sentido, se destacan aspectos como la depuración que consiste en un proceso de identificar cuál es la causa o raíz del problema y corregirlo. Ésta se contrasta con las pruebas existentes, con el fin de identificar errores iniciales. En el caso de algunos proyectos, el proceso de la depuración ocupa entre el 50% y 90% del tiempo total de desarrollo (McConnell, 2004).

El papel que tiene la depuración en la calidad de un software consiste en diagnosticar los defectos que se presentan. A este respecto, por ejemplo, los mensajes de error pueden marcar la diferencia en la usabilidad de un sistema para principiantes, en el sentido de

que, cada estudiante puede presentar diferencias en la solución del problema, ya sea solo o con la ayuda de otra persona (Becker et al., 2016).

Los errores de sintaxis suelen ser uno de los problemas principales de los estudiantes a la hora de programar. Esto ralentiza la retroalimentación puede resultar confusa, de manera especial para estudiantes nuevos, ya que aún no se encuentran tan familiarizados con la terminología utilizada en los mensajes de error típico (Finch y Edwards, 2020).

En el caso de los errores de sintaxis, resulta uno de los problemas más recurrentes para los estudiantes a la hora de programar. Por lo que ralentiza el progreso e impide recibir comentarios sobre la lógica del código. No obstante, estudios recientes revelan que el uso de comentarios mejorados de compilador proporcionan resultados poco cuantificables (Denny et al., 2014).

La depuración resulta un tanto complicada para los programadores novatos ya que requiere de la aplicación de nuevas y mejoradas habilidades que deben ser aplicadas de forma simultánea (Murphy et al., 2008).

3.2.4.1 J-LATTE

Holland et al., (2009) en su investigación con el tutor J-LATTE, menciona lo relacionado con la introducción a la programación con Java, considerando temas como: declaración, asignación, impresión, devolución, condicionales y bucles.

La interfaz de usuario se encuentra estructurada de la siguiente manera: enunciado del ejercicio (superior), editor (central), bloques de sintaxis de código (inferior) y la retroalimentación que presenta las observaciones de la ejecución del código a manera de un depurador (lateral derecha). Puede verse una imagen de J-LATTE en la Figura 3.15.

En lo referente a la interacción con el tutor, el estudiante de manera inicial se abstrae de la escritura completa en sintaxis del código, ya que, debe ir arrastrando bloques de código predefinido que considere necesario para la composición de su algoritmo.

Es importante mencionar que el estudiante puede solicitar comentarios en cualquier momento, enviando la orden ver la solución de su código, J-LATTE proporciona retroalimentación si encuentra errores de sintaxis, semántica y de estilo.

En el año 2008, la Universidad de Canterbury llevó a cabo un estudio con 26 voluntarios de un curso de introducción a la programación. Los estudiantes utilizaron J-LATTE en los laboratorios programados para el curso, previo a presentar al tutor virtual, se les aplicó un cuestionario con el fin de obtener los resultados iniciales.

Los conceptos necesarios para los ejercicios planteados en J-LATTE fueron cubiertos en conferencias en clase. Se dividió a los estudiantes en dos grupos, por un lado, el grupo experimental usó el sistema completo, mientras que, el grupo de control usó una versión

reducida, que simula la condición del aula, ya que no proporciona retroalimentación sobre los errores detectados.

Realizados los laboratorios con la interacción con J-LATTE se aplicó nuevamente el mismo cuestionario, durante el proceso estadístico se excluyeron los resultados de dos participantes que no registraron un tiempo de interacción adecuada. En este contexto, los resultados mostraron que los estudiantes del grupo de control pasaron menos tiempo con J-LATTE e intentaron más problemas que sus compañeros del grupo experimental.

Por otro lado, los estudiantes del grupo de control resolvieron significativamente más problemas, probablemente debido a la ayuda que recibieron a través de la retroalimentación de errores. Además, se pudo observar que las puntuaciones obtenidas de los dos grupos mejoraron con respecto a la prueba inicial, pero la mejora no fue significativa.

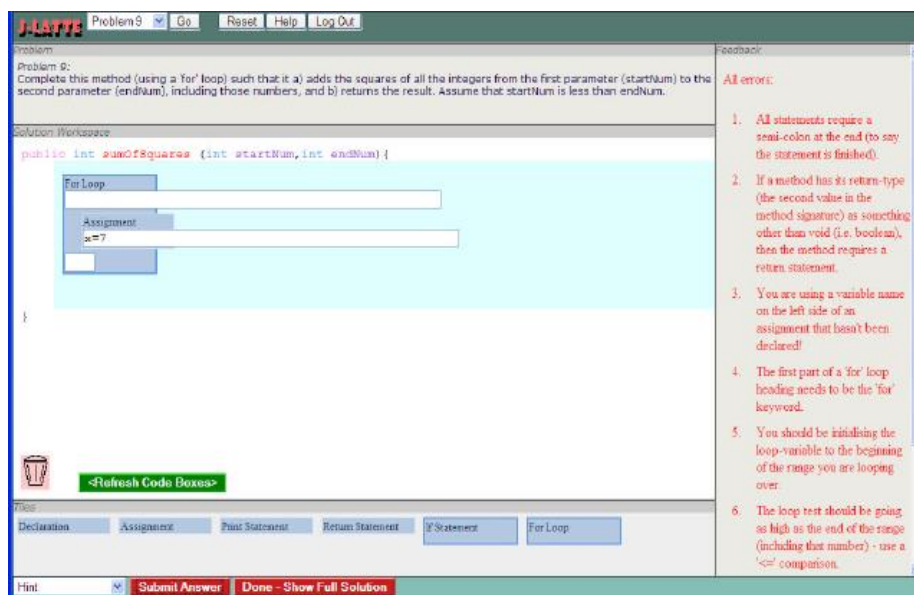


Figura 3.15 Interfaz de J-LATTE (fuente: Holland et al., 2009)

3.2.4.2 CPP-TUTOR

Abu-Naser (2009) en su investigación describe el experimento en el que midieron la efectividad de CPP-Tutor, su interfaz puede verse en la Figura 3.16, a través de una comparación del método tradicional de enseñanza (instructor y libro de texto) con el CPP-Tutor aplicado en un curso introductorio en programación C++ en la Universidad Al-Azhar en Gaza.

En cuanto a la interfaz de usuario implementada recoge características esenciales con respecto a un entorno de desarrollo integrado (IDE). Las secciones que presentó el CPP-Tutor fueron: el enunciado del ejercicio, descripción del ejercicio, el editor, una sección de botones para compilar su código y la sección destinada para ver los mensajes de error o el resultado de su código.

Asimismo, el depurador genera mensajes, detectando las intenciones del código escrito de manera errónea por el estudiante, incluso en algunos casos autocorrigiendo comandos con errores de sintaxis, al tiempo de brindarle al estudiante la posibilidad de ver la resolución del ejercicio.

Durante la aplicación del experimento se esperaba, que los módulos integrados en CPP-Tutor contribuyan como un mecanismo bien definido y dinámico, permitiendo mejorar significativamente el desempeño de los estudiantes en programación C ++.

Para esta investigación contaron con 62 estudiantes que terminaban la secundaria. La muestra de estudiantes se dividió en dos grupos coordinados para conocimientos previos similares del material C++ presentado. Al grupo de control (31 estudiantes) se le enseñó utilizando el método de enseñanza convencional (conferencia y libro de texto), mientras que al grupo test (31 estudiantes) se le enseñó utilizando sistema de tutoría inteligente Web CPP-Tutor.

La fase experimental, contó con tres etapas, en la primera se realizó una prueba inicial para evaluar los antecedentes relacionados al conocimiento de los temas que se van a presentar. Posteriormente, se siguió un periodo de entrenamiento de un mes, y finalmente se aplicó el cuestionario a todos los estudiantes para medir su nivel de retención.

Al culminar con el período de entrenamiento de un mes, se aplicó nuevamente el mismo cuestionario, y, finalmente se realizó una prueba posterior a ambos grupos, un mes después con el fin de medir su nivel de retención.

En las pruebas iniciales, el promedio de las calificaciones del grupo control fue un 10% más alto que las del grupo test, no obstante, esto cambió notablemente en los promedios de las pruebas aplicadas.

Después del período del entrenamiento, ya que el promedio de las puntuaciones del grupo test fueron un 17% más alto que el grupo control, de igual manera en el resultado alcanzado para medir la retención de los contenidos enseñados, fueron mejores en el grupo test.

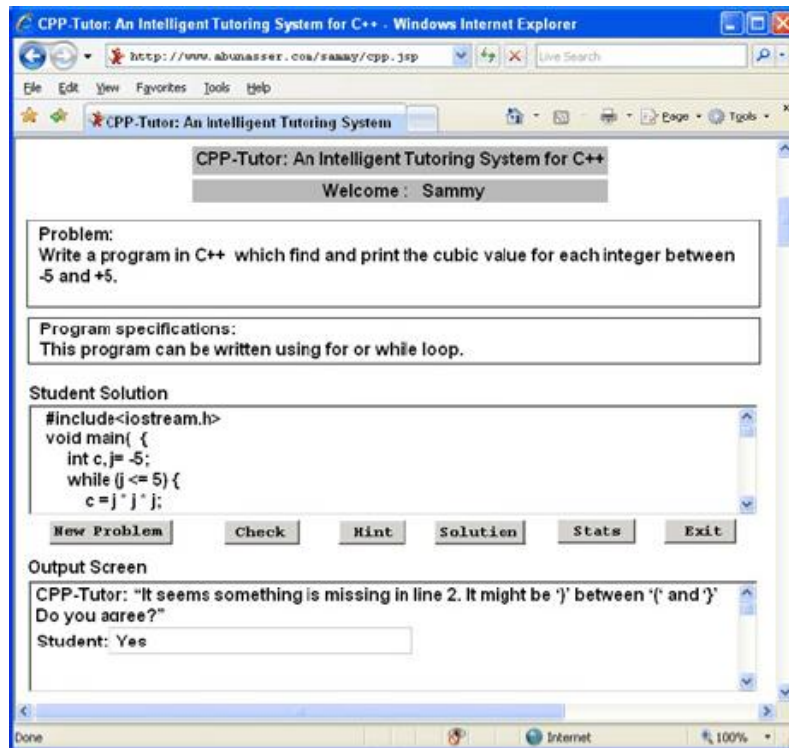


Figura 3.16 Interfaz de CPP-Tutor (fuente: Abu-Naser, 2009)

3.2.4.3 ASK-ELLE

Gardes et al., (2016) plantean un tutor llamado Ask-Elle, que apoya el aprendizaje el en desarrollo paso a paso de programas funcionales simples en Haskell, los ejercicios planteados abarcan conceptos iniciales y avanzan de manera incremental.

En cuanto a su interfaz presenta una lista de funciones (lateral izquierda), descripción del ejercicio (central superior), editor (central inferior), y retroalimentación (lateral derecha).

Las propiedades y soluciones en este modelo se presentan con mensajes de retroalimentación, brindando flexibilidad en una variedad de las soluciones suministradas a los estudiantes.

Abu-Naser (2009) considera como la principal contribución de su trabajo de investigación, que el tutor combina el desarrollo incremental de diferentes soluciones a un ejercicio de programación, con retroalimentación automatizada y que permiten mostrar soluciones y propiedades de programación específicas al estudiante, esperando cultivar en él, el criterio que un ejercicio tiene varias formas de resolución.

Durante su investigación probaron la viabilidad de este enfoque en varios experimentos, en los que analizan las soluciones de los estudiantes, tanto intermedias como finales a los ejercicios de programación, obteniendo resultados muy prometedores en cuanto al progreso tanto en el aprendizaje de los estudiantes a través de sus experimentos, como en el número de detecciones y soluciones presentadas durante el proceso.

Del mismo modo, como un trabajo futuro en Ask-Elle, se considera de manera adicional, mostrar si un programa de un estudiante es correcto o incorrecto, el incluir mensajes que un docente tradicional a veces de usa manera sutil, como por ejemplo: “esta es una buena solución, pero es mejor...”.

Adicionalmente, ya que en este fase del proyecto al encontrar un error Ask-Elle no identifica de manera precisa el lugar donde se dio el error, se considera gestionar mejoras para señalar con más precisión la parte de un programa que viola una funcionalidad. Puede verse una imagen de Ask-Elle en la Figura 3.17.

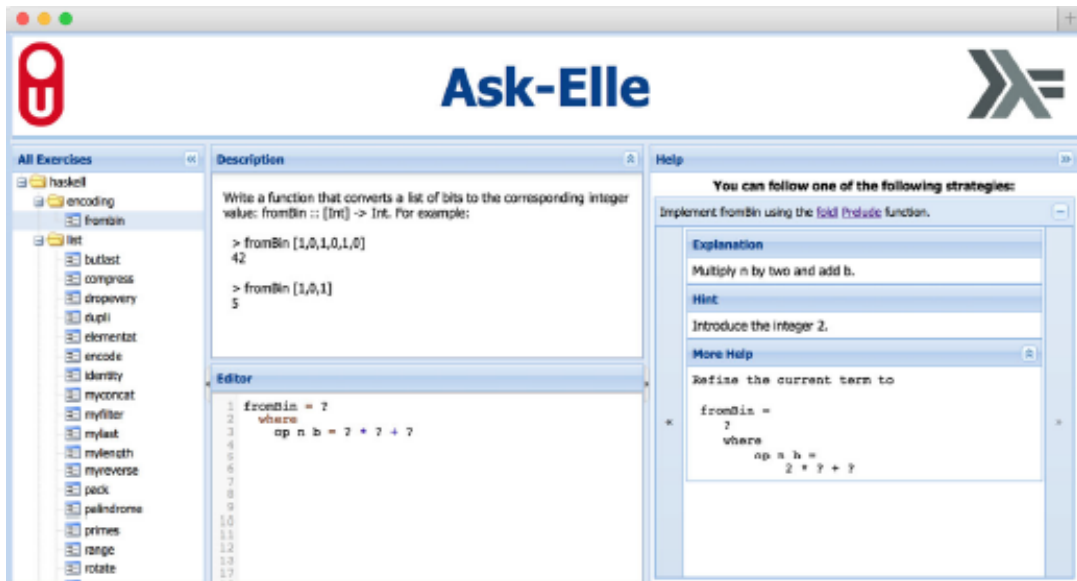


Figura 3.17 Interfaz de Ask-Elle (fuente: Gardes, 2016)

3.2.4.4 ANNETE

Day et al., (2019) se refieren a Annete, como un compañero de aprendizaje diseñado para apoyar a estudiantes universitarios en el aprendizaje de una programación en Java, consideran que muchos de los compañeros de aprendizaje carecen de una retroalimentación sólida.

La mayoría de ellos requieren que los estudiantes aprendan a utilizar una nueva herramienta, incrementando la dificultad que los estudiantes ya tienen en el aprendizaje de programación, por tanto, sería ideal que este compañero sea parte de una herramienta ya utilizada, especialmente en el ámbito universitario.

En este sentido, Annete es considerado como un módulo del IDE Eclipse, altamente utilizado por los estudiantes que forman parte de su de estudio, basándose en una investigación previa, que comprende el desarrollo de un componente para Eclipse (Penumala y González-Sánchez, 2018).

En la interacción con el usuario, se solicita una identificación de estudiante permitiéndole acceder a la “Vista de preguntas”, que se coloca en la parte inferior de la ventana, mientras que, el estudiante trabaja en el desarrollo de su algoritmo, cuenta con opciones de ejecución y depuración.

Los mensajes de la consola se capturan continuamente y se envían a la base de datos del servidor, con el fin de implementar mediante un algoritmo de aprendizaje supervisado, una estructura con los datos que entran y salen, recopilando datos del código escrito por los estudiantes, como palabras clave, mensajes de error del compilador, con esto se proporciona un mensaje de retroalimentación para mostrárselo al estudiante.

En su estudio se evaluó a 28 estudiantes, los resultados arrojaron que el 16% indicaron que el mensaje fue útil, el 26% el mensaje fue algo útil, el 41% que el mensaje no coincidía con la situación y el 17% no entendieron lo que significaba el mensaje.

Como trabajo futuro se consideró mejorar las funcionalidades de Annete, al igual que agregarlo en otro lugar del IDE, que no interfiera con el flujo de trabajo y la visualización a la que normalmente se encuentran acostumbrados los estudiantes en Eclipse, por otra parte, se consideró incorporar un avatar para Annete, con el fin de fortalecer la atracción visual en los estudiantes, como se visualiza en la Figura 3.18.

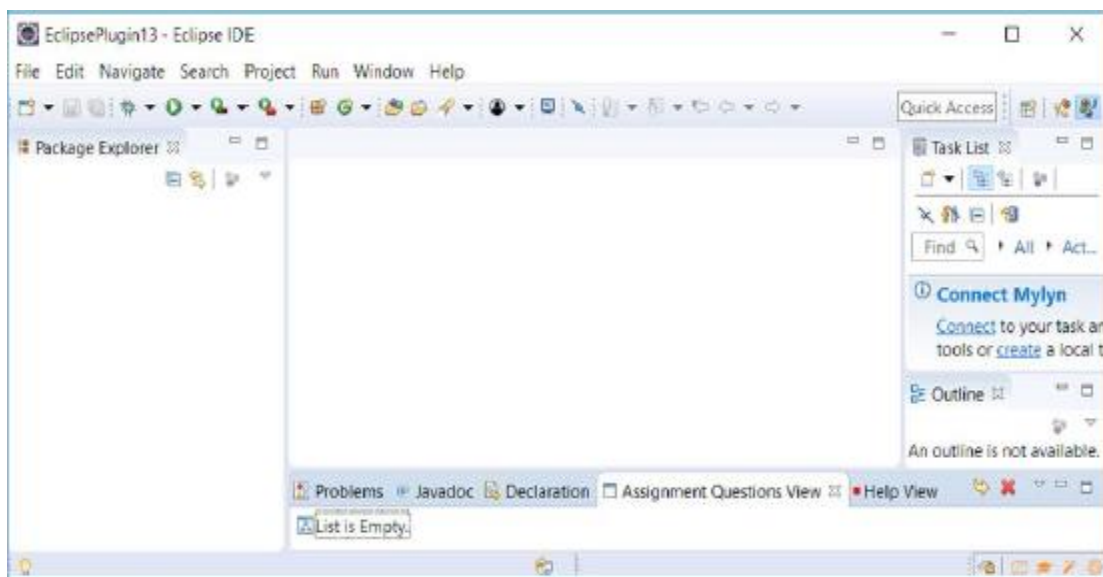


Figura 3.18 Interfaz de Annete (fuente: Day et al., 2019)

3.2.4.5 JEPPY

Pérez et al., (2020) describen el diseño e implementación de Jeppy, compañero de aprendizaje que puede ser utilizado como una herramienta para brindar soporte a programadores novatos en el lenguaje de programación C++.

Como metodología utiliza principalmente el factor pedagógico del constructivismo, permitiendo el desarrollo del pensamiento del estudiante, ya que el agente no le enseña

directamente al niño, sino que le presenta soluciones similares al ejercicio planteado, en busca de aportar en su aprendizaje. Se considera que, al no brindar demasiadas retroalimentaciones al alumno, no interfiere en el desarrollo de la actividad cuando el estudiante se encuentra detenido en un error específico.

Con relación a la interfaz del compañero de aprendizaje, puede verse en la Figura 3.19, que utiliza un dibujo animado al que lo nombraron Jeppy, que incluye mensajes y gestos afectivos a fin de captar la atención y empatizar con el estudiante.

En las pruebas de funcionamiento de los componentes de Jeppy, se trabajó con los comentarios de estudiantes que se encontraban tomando un curso introductorio de programación, en una universidad estatal.

Con el fin de verificar que eran novatos en programación, se les aplicó una encuesta, identificando a 18 estudiantes novatos en programación. Así también, se les proporcionó un código fuente que contenía errores, los cuales debían detectar de manera secuencial, sin la intervención de compañeros o docentes.

Cada vez que el alumno editaba el código, se guardaba línea por línea, y cuando enviaban a ejecutar su código Jeppy aparecía con mensajes de ser necesario, los log's de este proceso contribuyeron al proyecto.

A fin de verificar la efectividad de los mensajes del compañero de aprendizaje, registrando que, de 538 veces que apreció un mensaje de error, en tan sólo 159 ocasiones este fue utilizado por parte de los estudiantes para corregir su código. De este número, el 66,7% utilizó las pistas de solución y el 33% en ejemplos de solución al ejercicio planteado.

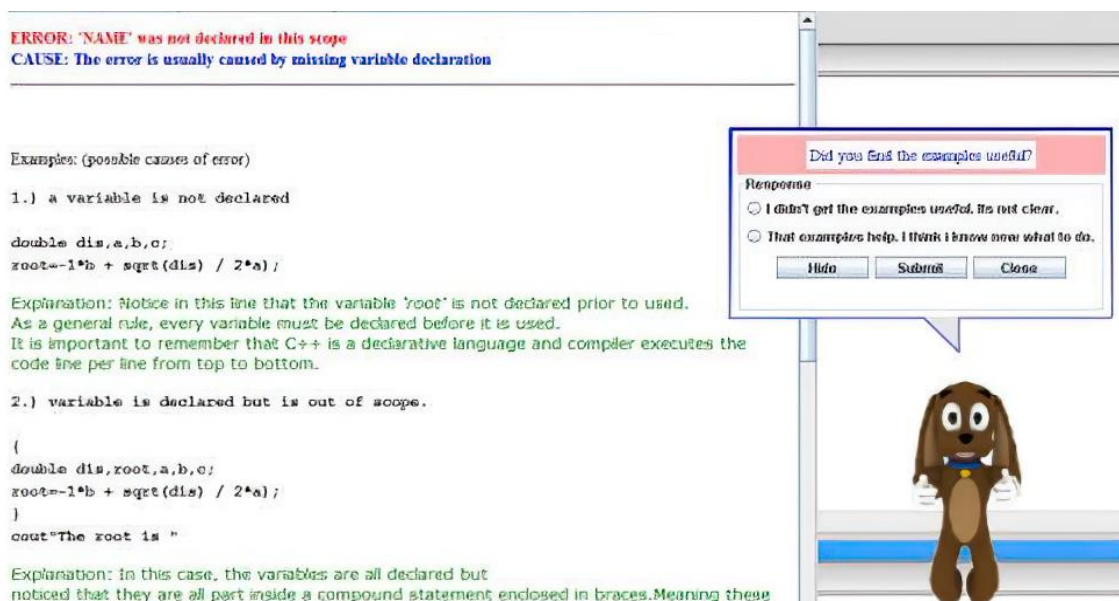


Figura 3.19 Interfaz de JEPPI (fuente: Pérez et al., 2020)

En la Tabla 3.5, se consideran cinco características relevantes, como parte de una estrategia de tutoría estructurada y que representan partes clave del proceso de aprendizaje (Crow et al., 2018). Las mismas que se detallan a continuación:

- **QU:** Los estudiantes tienen la posibilidad de contar con preguntas como: tipo cuestionario de opción múltiple, verdadero o falso, o preguntas de respuesta corta, adicional al editor para los ejercicios de programación.
- **CP:** Cuenta con un programa de planificación a manera de un guion de aprendizaje.
- **PL:** Programas prefabricados, que se presentan como ayuda a ejercicios planteados, a los pueda acceder el estudiante durante alguna etapa de aprendizaje.
- **LE:** Contenido de lección y/o material de referencia, información de referente a los contenidos de aprendizaje.
- **SO:** Soluciones de retroalimentación suministradas como recurso educativo.

COMPAÑEROS DE APRENDIZAJE	QU	CP	PL	LE	SO
J-LATTE	-	✓	-	-	-
CPP-TUTOR	-	-	-	-	✓
ASK-ELLE	-	-	-	-	-
ANNETE	-	✓	✓	-	✓
JEPPY	-	-	✓	✓	✓

Tabla 3.5 Características (fuente: Crow et al., 2018)

Capítulo 4. Propuesta metodológica

En este capítulo se presenta una propuesta de la metodología (MEDIE_GELIDEC) para la elaboración de un compañero de aprendizaje para enseñar programación a niños de Educación Primaria.

Esta metodología extiende la metodología MEDIE (Tamayo-Moreno, 2017) descrita en la Subsección 3.2.3, para el diseño e integración de agentes conversacionales en el aula. Esta metodología consta de seis fases:

1. Comunicación y co-diseño con docentes y estudiantes
2. Análisis de diálogos
3. Gestor de diálogo
4. Editor de código
5. Validación de la interfaz y de la funcionalidad
6. Sesiones prácticas

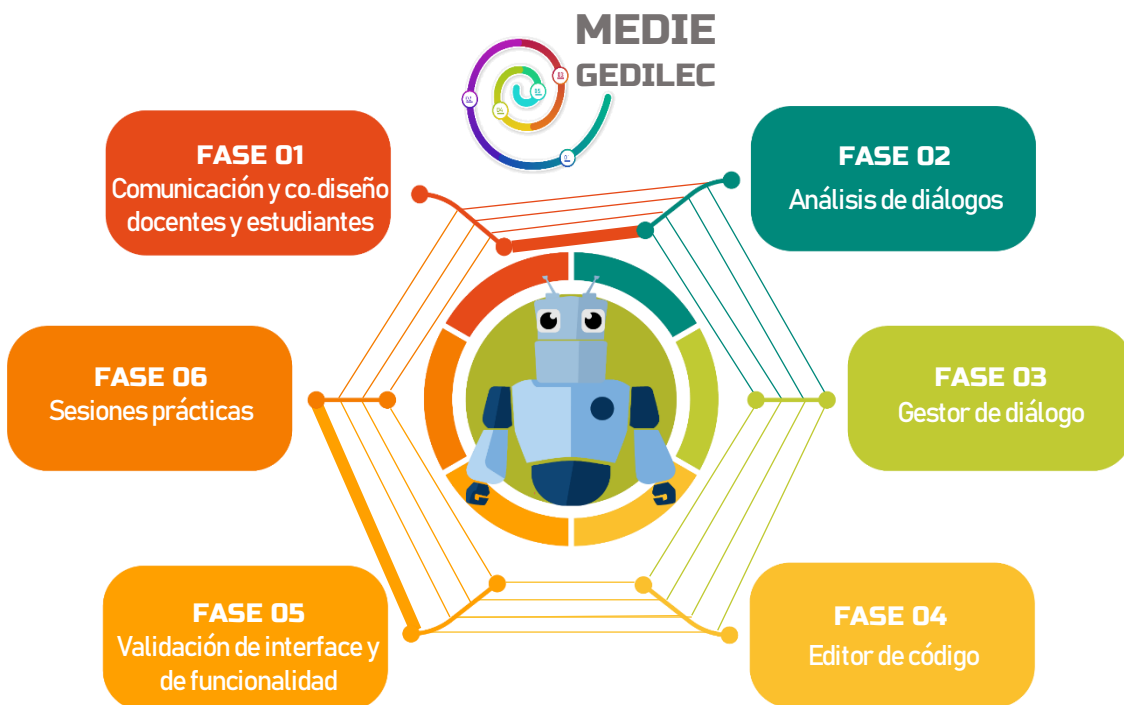


Figura 4.1 MEDIE_GEDILEC ciclo iterativo e incremental (fuente: propia)

Como se mencionó en el capítulo 2, hay autores que consideran que es posible reutilizar aquello que ya se ha diseñado, y aplicado a la práctica con éxito, especialmente la experiencia obtenida en ello, pero que es complicado establecer directrices más generales

puesto que dependerán del contexto y los objetivos de aprendizaje y grupos de alumnos (Haake, 2009).

De consideraciones como estas, radica la importancia de uso del DCU o metodologías como MEDIE, que permiten ajustarse al contexto, objetivos de aprendizaje o características de los alumnos.

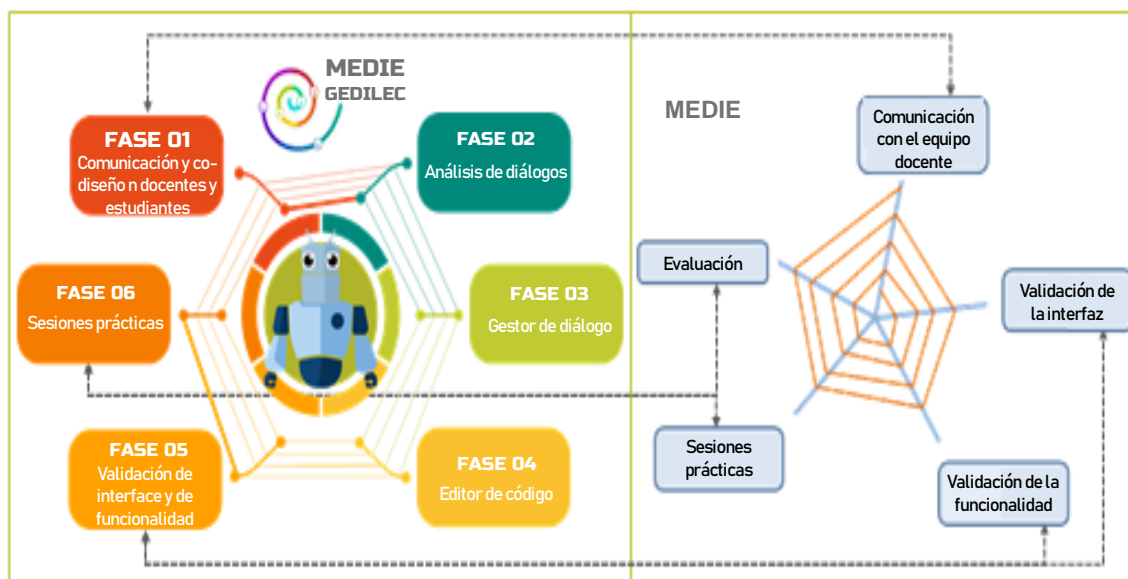


Figura 4.2 Fases de MEDIE_GEDILEC y cómo extiende de MEDIE (fuente: propia)

El ciclo que sigue MEDIE (Tamayo-Moreno, 2017) es iterativo e incremental. Como puede apreciarse en la Figura 3.1. MEDIE es una versión extendida del Diseño Centrado en el Usuario (Lorés et al., 2002).

Como se presenta en la Figura 4.2 MEDIE_GEDILEC, considera las fases de MEDIE y las extiende para el aprendizaje de programación con un compañero de aprendizaje.

4.1 Fase 1: Comunicación con docentes, estudiantes y co-diseño

4.1.1 Comunicación con docentes y estudiantes

Esta fase está orientada a realizar un análisis del contexto en el que se va a desarrollar el compañero de aprendizaje y un aprendizaje de las características de los estudiantes que serán los usuarios.

En esta fase se debe tratar de abarcar este proceso en detalle, comunicándose para ello con los expertos docentes y con los propios estudiantes.

Para obtener de primera mano la información de las necesidades de los usuarios es importante crear una relación entre el diseñador y el usuario. Observando las tareas

cotidianas del usuario, lo que permitirá identificar características que incluso el usuario puede pasar por alto.

De no considerar este proceso y su contexto, se puede sobreponer los criterios del diseñador sobre las necesidades de la actividad. En la Figura 4.3 se observa la relación Diseñador-Usuario.



Figura 4.3 Relación Diseñador-Usuario (fuente: Escobar, 2010)

De los especialistas, se consiguen las experiencias previas, reduciendo la posibilidad de incurrir en errores previamente detectados. Por otra parte, los docentes son quienes mejor interpretan las necesidades del proceso de aprendizaje y quienes contribuirán con la manera de la integración del compañero de aprendizaje, en el aula.

El integrar a los usuarios descritos permitirá plantear los requisitos de manera coherente y oportuna los objetivos que se desean alcanzar en el desarrollo e implementación del compañero de aprendizaje. Al respecto Kuniavsky (2003) sugiere algunas características que deben seguir los requisitos de los usuarios:

- Razonables: deben ser tareas reales, es decir que en verdad se lleven a cabo en el desarrollo del proceso de aprendizaje.
- Orientarse al objetivo final: deben contextualizarse, enmarcados en la motivación mayor, en este caso el aprendizaje.
- Específicos: debe perseguir un objetivo concreto.
- Factibles: no considerar metas irrealizables, en cuanto a recursos de cualquier índole.
- Duración razonable: dividirla en tareas en el caso de que un requisito pueda incurrir en un tiempo extenso de desarrollo o implementación.

Para las sesiones presenciales, el equipo de diseño debe asistir a la institución educativa a fin de observar el entorno en el que se ejecuta el proceso de aprendizaje, así como las actividades y necesidades de docentes y estudiantes.

Para las sesiones virtuales, se recomienda que el equipo de diseño se una por el mismo medio o plataforma que utilizan para las sesiones diarias, a fin de ver la interactividad y particularidades que se pueden dar en el proceso.

Al igual se deben tener reuniones en línea, con cada uno de los grupos para obtener mayores detalles del proceso, actividades y necesidades de los docentes y estudiantes.

4.1.2 Co-diseño

En estas primeras fases del desarrollo del compañero de aprendizaje, al no contar con una forma definida del mismo se debe obtener información directa de cómo desean que sea los estudiantes.

Como se observa en la Figura 4.4, se puede obtener información de los requisitos de los usuarios mediante la aplicación de prototipos (modelos desechables elaborados específicamente para la evaluación y decisiones de diseño).

En este sentido, Zille et al., (2017) indica que co-diseño es un método que permite el desarrollo de productos por medio de la experimentación, determinando las necesidades reales de los usuarios, y con el fin de mejorar el producto tanto en la funcionalidad y usabilidad de un compañero de aprendizaje.

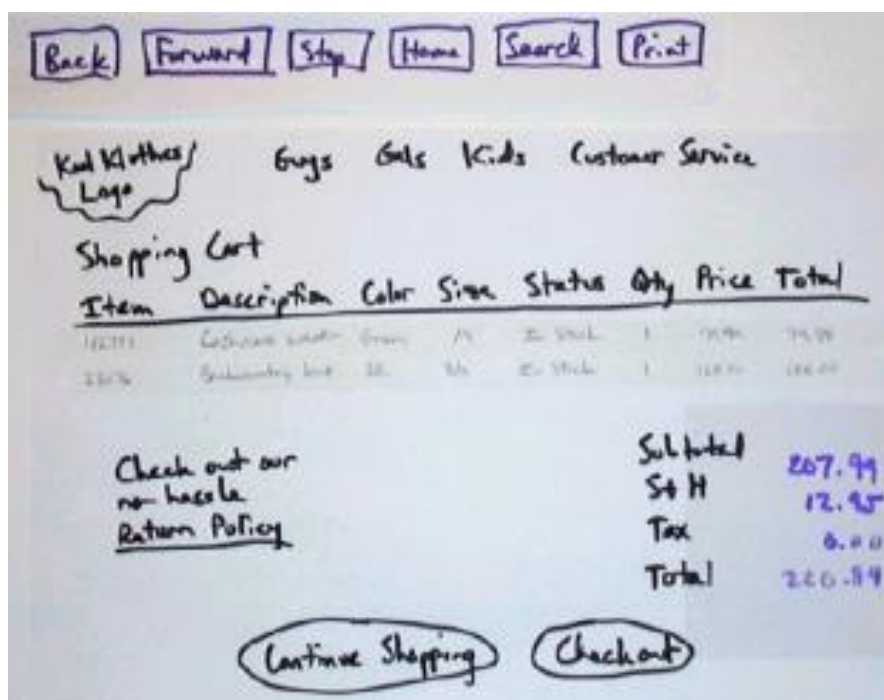


Figura 4.4 Prototipo en papel (fuente: Fitzgerald, 2013)

Como los estudiantes serán quienes utilizarán el entorno del compañero de aprendizaje, es importante incluir sus ideas y expectativas (Morales et al., 2017). En este contexto, para Manzini (2015) “todo diseño es (o debería ser) una actividad de investigación en sí misma que promueva experimentos sociotécnicos”.

Las encuestas son instrumentos útiles, que se deben estructurar en base a aspectos que se consideren relevantes. Es importante realizarlas tanto a docentes, y a estudiantes. En el co-diseño es importante no omitir el criterio de estudiantes, en virtud que serán ellos quienes lo usen (Bacigalupo et al., 2010).

En consideración, que la presente propuesta metodológica, se aplicaría en un entorno virtual respecto a un entorno presencial, se deberá obtener información, tanto de docentes, como de estudiantes, sobre la disponibilidad tecnológica de cada uno de los actores del proceso de enseñanza.

En este sentido, entre las preguntas que se pueden incluir en las encuestas para los docentes, se pueden considerar las siguientes:

- ¿Género?
- ¿Edad?
- ¿Dispone de un ordenador en casa?
- ¿Dispone de Internet en casa?
- ¿Con qué frecuencia del uso del internet para crear contenidos de aprendizaje?
- ¿Qué contenidos considera, deben incluirse en el módulo de aprendizaje?
- ¿Considera que un compañero de aprendizaje puede contribuir en el proceso de aprendizaje?
- ¿Considera importante que los estudiantes puedan usar dispositivos de entrada salida para comunicarse con el compañero de aprendizaje?
- ¿Puede describir la forma que tendría el compañero de aprendizaje?
- ¿Considera que un compañero de aprendizaje puede motivar en el proceso de aprendizaje a los estudiantes?
- ¿Cómo debería comportarse el compañero de aprendizaje, si el estudiante no entiende algo?
- ¿Cuántas veces te gustaría usar el compañero de aprendizaje?
- ¿Cuánto tiempo a la semana considera que se puede usar al compañero de aprendizaje?

- ¿Considera útil que el compañero de aprendizaje presente contenido de ayuda?
- ¿Sabes programar, o te gustaría hacerlo?
- ¿Crees que la programación es difícil de comprender?
- ¿Además de los contenidos de programación, qué te gustaría conversar con el compañero de aprendizaje?
- ¿Te gustaría, al finalizar un ejercicio, que el compañero de aprendizaje te presente el resultado de tu código?
- ¿Te gustaría que el compañero de aprendizaje te dé pistas de la resolución del ejercicio, si te encuentras en dificultades, al programar?

Entre las preguntas que se pueden incluir en las encuestas para los estudiantes, se pueden considerar las siguientes:

- ¿Qué es programar?
- ¿Te gustaría aprender a programar con un compañero de aprendizaje?
- ¿Cómo te gustaría aprender a programar?
- ¿Cómo te gustaría que el compañero se comunique contigo?
- Si no entiendes algo, ¿cómo te gustaría que fueran los mensajes de ayuda el compañero de aprendizaje?
- ¿Cuánto tiempo te gustaría practicar con el compañero de aprendizaje?
- ¿Dónde te gustaría trabajar con el compañero de aprendizaje?

4.2 Fase 2: Análisis de diálogos

Al momento de diseñar el diálogo entre el estudiante y el compañero de aprendizaje, es necesario comprender cómo les gustaría a los estudiantes comunicarse, para esto se recomienda el uso de técnicas que permitan autonomía a los estudiantes.

Tanto en sesiones presenciales y virtuales, los diálogos propuestos por los estudiantes para el análisis pueden ser realizados en papel, recogiendo al final de la actividad en sesiones presenciales y permitiendo se entreguen como imagen en sesiones virtuales.

4.2.1 Storytelling

Durante las primeras sesiones, para el análisis del diálogo es importante, en la recolección inicial de la información, permitir a los estudiantes expresarse, de manera libre, sobre la manera que desea conversar e interactuar con el compañero de aprendizaje.

Una técnica de prototipado, que puede ser de utilidad para este propósito, es Storytelling, que se basa en exponer los conocimientos a través de historias o hechos que se pueden producir en el entorno del alumno o que este mismo puede imaginar.

Carrier (1973) indica que los niños consideran al papel como una pantalla, ya que su imaginación es tan poderosa, que incluso sin hablar realmente con el ordenador su mente puede crear todo el diálogo, como si fuera real.

Por otra parte, en Liang et al., (2017) se indica que, usando Storytelling, los niños pueden crear todo el diálogo basándose en una narrativa de historias. La Figura 4.5 permite visualizar ejemplos de historias de conversación usando la estrategia de Storytelling.

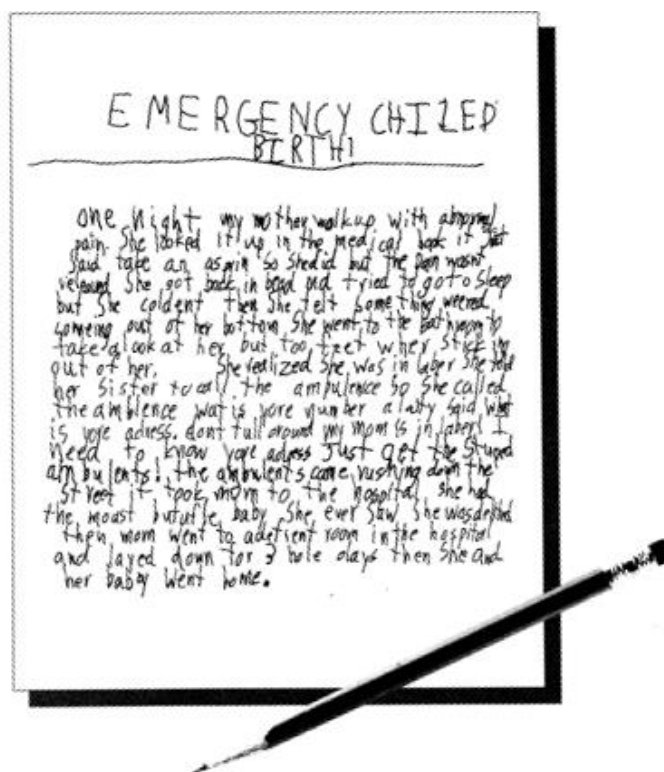


Figura 4.5 Ejemplo de Storytelling (fuente: Liang et al., 2017)

De acuerdo con las categorías principales del modelo IBIS (tema, posición y argumento), el esquema se centra en el nivel de razonamiento explícito mostrado durante la actividad al incluir dos categorías clave adicionales, a saber, posiciones explícitas y argumentos explícitos.

De esta manera, por ejemplo, la contribución de un estudiante podría identificarse como un argumento o un argumento explícito basado en si simplemente apoyó / objetó una posición previamente articulada o también mostró alguna forma de razonamiento explícito sobre conceptos de dominio (Tegos, 2015).

La Tabla 4.1 describe un esquema utilizado para el proceso de análisis del discurso.

Categoría	Descripción
Fuera de la tarea	Expresiones que son irrelevantes para la tarea o que juegan un papel puramente social (por ejemplo, "Hola")
Repetición	Reiteraciones de declaraciones anteriores
Asunto	¿Qué debe hacerse o resolverse para continuar con la tarea general?, por ejemplo, ¿Cree que este principio es inapropiado aquí?
Posición	Opiniones que suelen estar relacionadas con la resolución de un problema planteado (por ejemplo, "El principio de redundancia es importante para esta tarea")
Posición explícita	Posiciones que muestran explícitamente el razonamiento sobre los conceptos del dominio (por ejemplo, "Basado en el principio de redundancia, evitar la narración y el texto" idéntico "es una buena manera de permitir que los estudiantes aprendan ...")
Argumento	Opiniones que apoyan o se oponen a una posición (p. Ej., "Sí, es cierto", "No estoy de acuerdo contigo")
Argumento explícito	Argumentos que (al igual que las posiciones explícitas) muestran un razonamiento explícito sobre los conceptos del dominio (por ejemplo, "No estoy de acuerdo porque los principios de diseño son más importantes para los principiantes que para los estudiantes avanzados")
Administración	Expresiones orientadas a la gestión que se utilizan para la coordinación de tareas (p. Ej., "Date prisa, por favor ... envía nuestra respuesta")
Entendimiento común	Declaraciones breves (generalmente de una o dos palabras) utilizadas por los estudiantes para establecer un entendimiento común sobre el tema (por ejemplo, "Lo entiendo", "Lo entiendo", "OK")

Tabla 4.1 Esquema de análisis del diálogo (fuente: Tegos, 2015)

En el momento de analizar el diálogo, es necesario comprender cómo les gustaría a los alumnos hablar con el agente. En este sentido, se plantea revisar (Ocaña et al., 2019):

- ¿Qué temas les gustaría tratar a los estudiantes al hablar con su ordenador?
- ¿Los estudiantes usarían preguntas para hacer actividades en el ordenador?
- ¿Los estudiantes usarían exclamaciones para preguntar al ordenador?
- ¿Los estudiantes usarían oraciones cortas / medias / largas?
- ¿Los estudiantes conocen términos relacionados con la programación?
- ¿Los estudiantes consideran el ordenador como su amigo?
- ¿Existen diferencias significativas entre los diálogos de los estudiantes?

A fin de brindar una respuesta a estas preguntas se sugiere considerar los siguientes parámetros:

Criterio	Posibles valores
Curso	Nivel educativo
Tipo	Cuento de fantasía, diálogo, cuento personal o programa.
Tema	Definir la temática del diálogo.
#Preguntas	Número de preguntas utilizadas por el niño.
#Exclamaciones	Número de exclamaciones utilizadas por el niño.
Longitud	Definir tipos de longitud, acorde al número de palabras utilizadas por los estudiantes.
Conocimiento	En base a los diálogos desarrollados, por los estudiantes definir el nivel de conocimiento en el objetivo de aprendizaje.
Amistad	Si/no el estudiante considera que el compañero de aprendizaje es su amigo en el proceso de aprendizaje.

Tabla 4.2 Parámetros para analizar los diálogos agente-alumno

4.3 Fase 3: Gestor de diálogo

Una vez realizado el contexto inicial del proyecto y desarrollado el análisis del diálogo, el siguiente paso es determinar la manera en que se va a realizar la interacción de la comunicación estudiante – compañero de aprendizaje.

En este sentido, se recomienda el uso de tecnologías basadas en la Web, permitiendo que el gestor de diálogo conserve sus funcionalidades tanto para sesiones presenciales como para sesiones en línea.

4.3.1 API Text-to-Speech

Si se incluye la opción del ingreso información por la voz del estudiante, se puede considerar el uso de una API de Speech Recognition como la disponible en Google Cloud que permite convertir voz en texto (ver Figura 4.6).

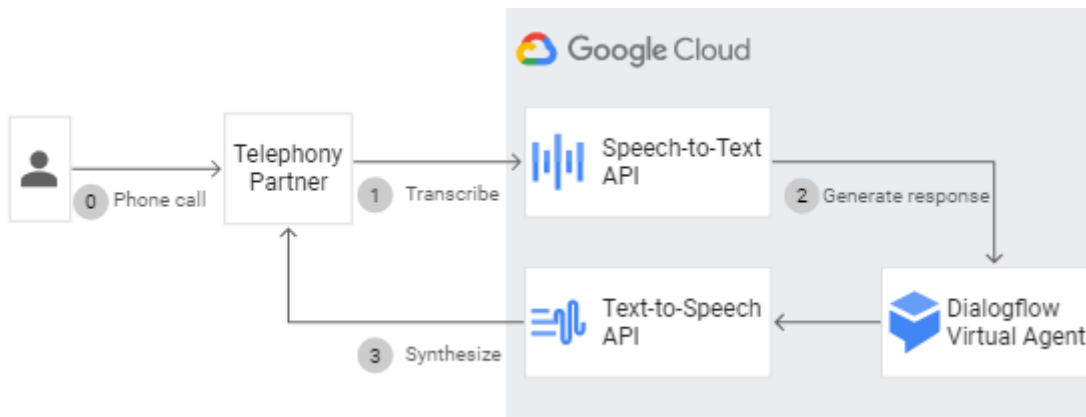


Figura 4.6 Funcionamiento de la API Text-to-Speech (fuente: Google Cloud, 2020)

4.3.2 AIML

En este contexto, una de las opciones que se puede considerar es el lenguaje de programación basado en XML, AIML (Artificial Intelligence Mark-up Language), fue diseñado específicamente para ayudar en la creación de la primera entidad chatbot informática de lenguaje artificial ALICE. En la Figura 4.7 se puede observar el funcionamiento de AIML.

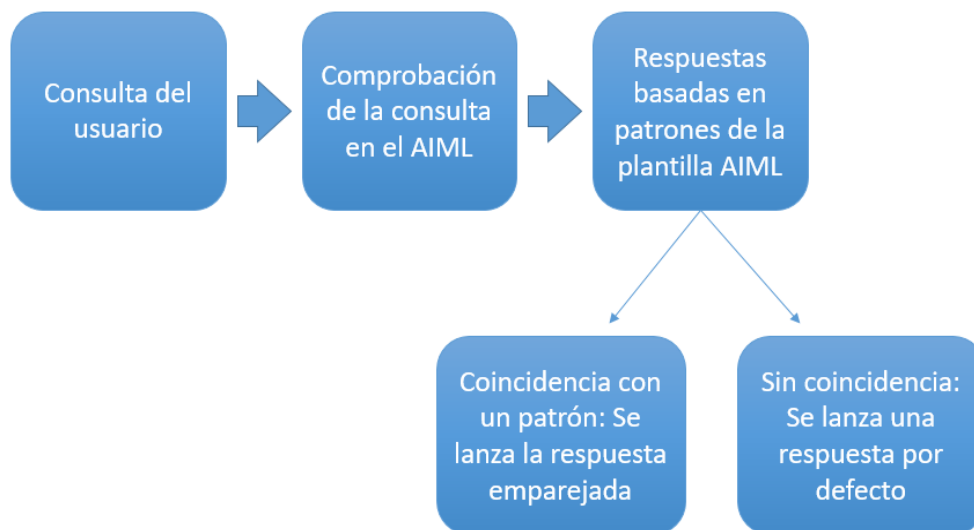


Figura 4.7 Funcionamiento de un gestor de diálogo con AIML (fuente: Crespo y Cabrera, 2020)

Al iniciar este proceso de interacción es importante definir los dispositivos que van a utilizarse para el ingreso de información por parte del usuario. Si este va a ser solo el teclado, o teclado y micrófono.

El gestor de diálogo, en su estructura debe definir, las intenciones, las entidades y las respuestas del compañero de aprendizaje. Además, de AIML se puede considerar por

ejemplo gestores de diálogo de pago como son: Lex de Amazon (Kepuska y Bohouta, 2018) o Watson de IBM (Yeves-Martínez y Pérez-Marín, 2019).

Como se ha indicado anteriormente. AIML se basa en XML y sus tres etiquetas principales son: las categorías, los patrones y las plantillas. En la Figura 4.8 se describe el formato AIML.

<category> las categorías son la unidad básica de conocimiento de AIML, compuesto de dos elementos: el patrón y la plantilla.

<pattern> los patrones representan lo que el usuario dice, mediante el ingreso de información, ya sea por teclado, o transformada en texto con el uso de la API Text-to-Speech.

<template> son las respuestas específicas que encuentra el gestor de diálogo para los patrones ingresados por el usuario.

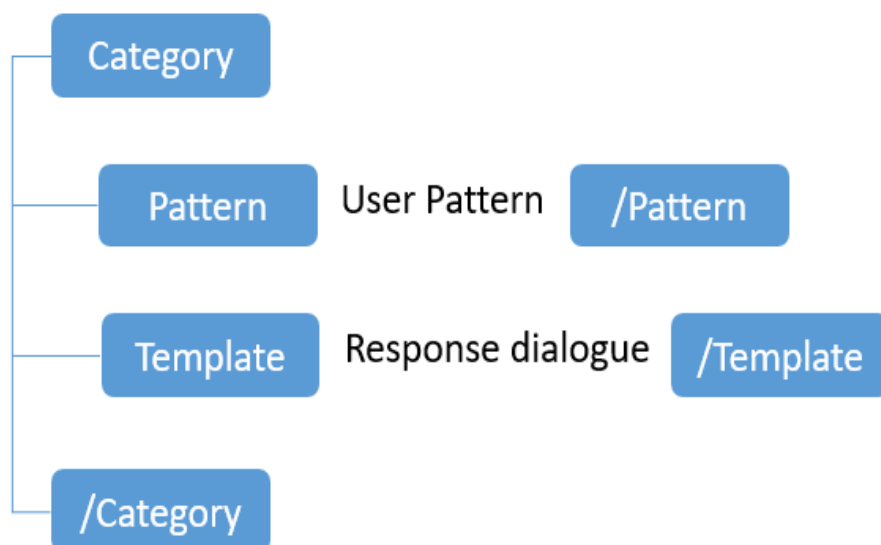


Figura 4.8 Formato AIML (fuente: Crespo y Cabrera, 2020)

En este sentido, se puede considerar un esquema de funcionalidad para el gestor de diálogo. La interacción inicia el ingreso de la información por parte del usuario, esta puede ser mediante texto o voz. En el caso de ser la segunda se deberá utilizar la API de Google text-to-speech.

El esquema de funcionalidad del gestor de diálogo se inicia con la entrada por parte del usuario, por texto o voz (de ser el caso se utilizará la API text-to-speech para transformar la voz en texto) y de esta manera enviar el procesador de lenguaje natural, para emitir desde el gestor de diálogo una respuesta hacia el usuario, acorde a las coincidencias encontradas en su base de conocimiento (ver Figura 4.9).

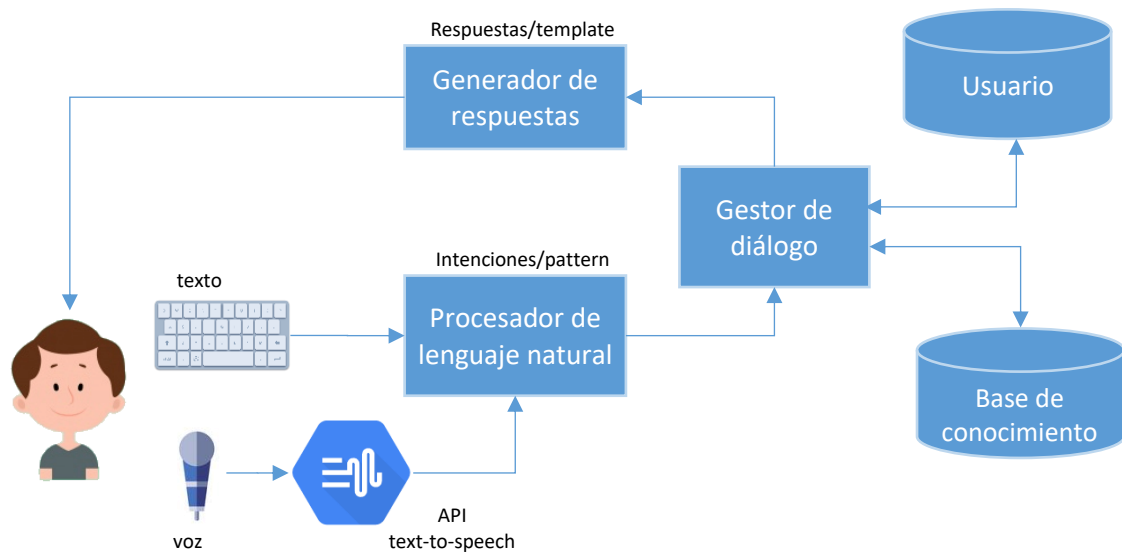


Figura 4.9 Esquema de funcionalidad del Gestor de diálogo (fuente: propia)

4.4 Fase 4: Editor de código

En esta fase se determina qué editor de programación se va a implementar o desarrollar, para llevar a cabo el proceso de aprendizaje, brindando a los estudiantes oportunidades para crear pensando (Papert, 1980; Ocaña et al., 2019).

En la metodología, no se considera conveniente trabajar con editores de código con software de escritorio, sino en línea, permitiendo obtener las mismas funcionalidades de ejecución y depuración, tanto para sesiones presenciales, como para sesiones virtuales.

4.4.1 Lenguaje de programación

En esta fase se debe elegir un lenguaje de programación que se utilizará con el compañero de aprendizaje. En esta propuesta se considera el uso de pseudocódigo.

El pseudocódigo integra una serie de palabras con un formalismo muy sencillo, una mezcla de frases del lenguaje común, instrucciones de programación y palabras clave que definen estructuras básicas (Mathieu, 2014).

En este sentido, se puede definir una estructura del pseudocódigo para conceptos tradicionales de un curso de introducción a la programación: entrada/salida, condicionales y bucles (Morales et al., 2020).

4.4.1.1 Entrada-salida: permiten el ingreso de información por parte del usuario (entrada) y presentación de la información (salida). Se presenta en una estructura secuencial como se detalla en la Figura 4.10.

```
Empezar
Escritura ("Ingresa tu edad:")
Lectura Edad
Terminar
```

Figura 4.10 Ejemplo de entrada-salida (fuente: Mathieu, 2014)

4.4.1.2 Condicionales: estructura de programación de condición lógica que se ejecuta si se cumple la misma. Las condicionales siguen una estructura selectiva, como se puede observar en la Figura 4.11.

```
Empezar
Escritura ("Ingresa un valor:")
Lectura Valor
Si Valor>0 entonces
  Escritura ("Su número es positivo")
Fin Si
Terminar
```

Figura 4.11 Ejemplo de condicionales (fuente: Mathieu, 2014)

4.4.1.3 Bucles: estructura de programación repetitiva que permite ejecutar un bloque de código por varias veces. Como se puede observar en la Figura 4.12, los bucles siguen una estructura repetitiva.

```
Empezar
Escritura ("Ingresa un valor:")
Lectura Valor
Para i=0 hasta Valor
  Escritura i
Fin Para
Terminar
```

Figura 4.12 Ejemplo de bucles (fuente: Mathieu, 2014)

Según el enfoque de crear pensando, la escritura es un medio poderoso para que los estudiantes piensen y razonen; y, por lo tanto, programen. En la propuesta, la atención se centra en el entorno de programación no visual basado en bloques, sino en código de texto. Algunas características que se deben considerar, en la elección de un editor, serían:

- Sea una aplicación en línea.
- Permita ejecutar el código.
- Soporte varios lenguajes de programación.
- Diseño sencillo y limpio.
- Brinde facilidad de uso para el estudiante.

4.4.2 Ejemplos compiladores / intérpretes

Como se revisó anteriormente, los compañeros de aprendizaje que enseñan programación, cuentan con editores que permiten la ejecución del código ingresado por el estudiante, pero generalmente no describen qué compilador/intérprete usan.

Existen diferentes opciones de compiladores o intérpretes en línea, que se pueden utilizar para cumplir con la ejecución de código, por ejemplo:

4.4.2.1 Ideone

Ideone es una herramienta en línea que le permite compilar código fuente y ejecutarlo, permitiendo seleccionar de entre más de 60 lenguajes de programación. Su interfaz se puede observar en la Figura 4.13.



Figura 4.13 Interfaz de Ideone (fuente: HTTP1)

4.4.2.2 CodePad

CodePad es un intérprete/compilador en línea, con un diseño sencillo y limpio. Cuenta con 13 lenguajes de programación. Su interfaz se puede observar en la Figura 4.14.

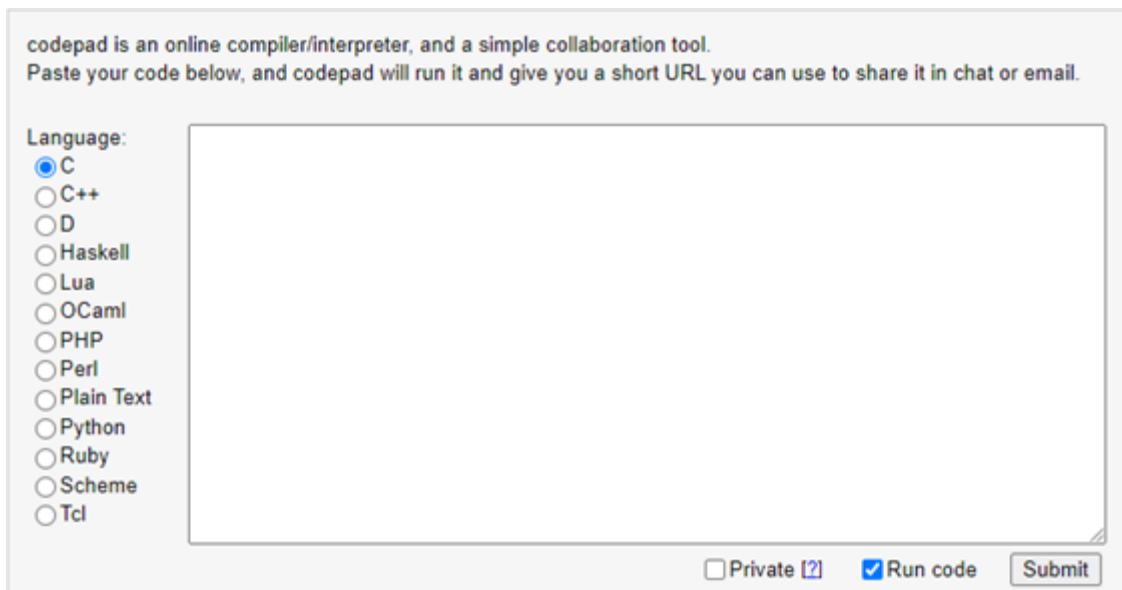


Figura 4.14 Interfaz de CodePad (fuente: HTTP2)

4.4.2.3 Python

Interprete para el lenguaje de programación Python, que permite ejecutar su código Python en el shell interactivo en línea. Su interfaz se puede observar en la Figura 4.15.

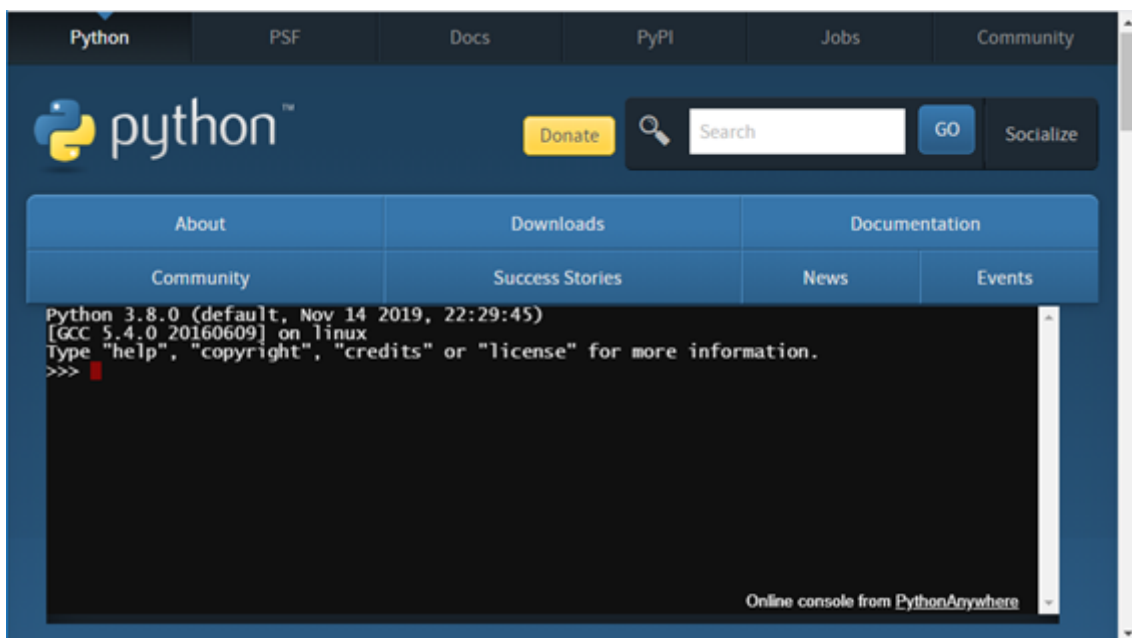


Figura 4.15 Interfaz del intérprete Python (fuente: HTTP3)

4.4.2.4 WritePhpOnline

Interprete en línea orientado para los desarrolladores Web que trabajan con los lenguajes PHP y JavaScript. Su interfaz se puede observar en la Figura 4.16.

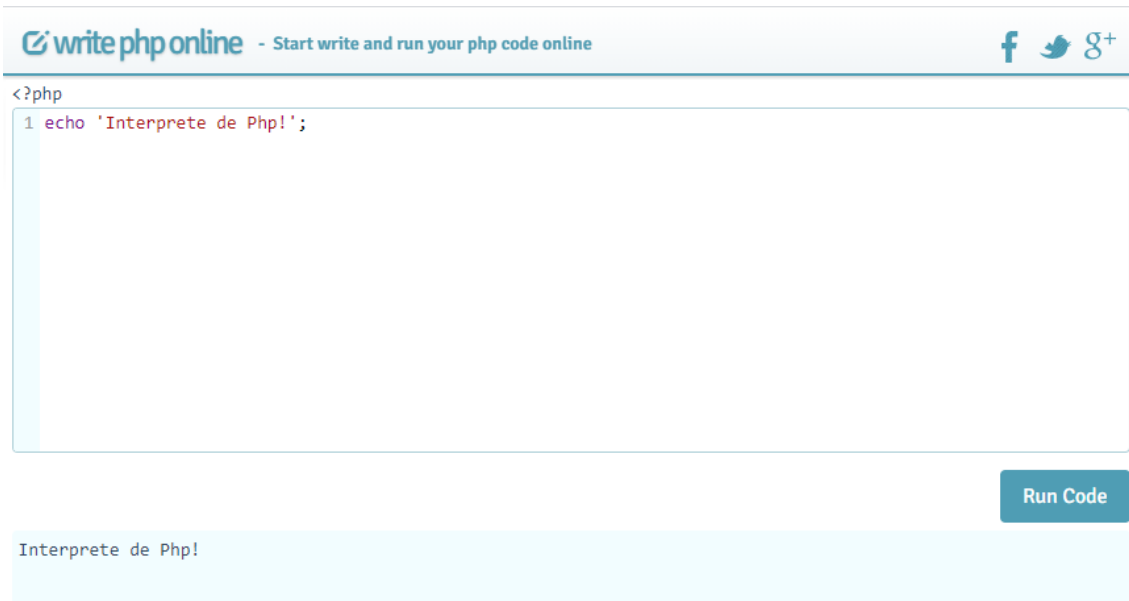


Figura 4.16 Interfaz de WritePhpOnline (fuente: HTTP4)

4.4.2.5 Browxy

Una herramienta en línea que le permite compilar código fuente y ejecutarlo, cuenta con varios lenguajes de programación. Su interfaz se puede observar en la Figura 4.17.

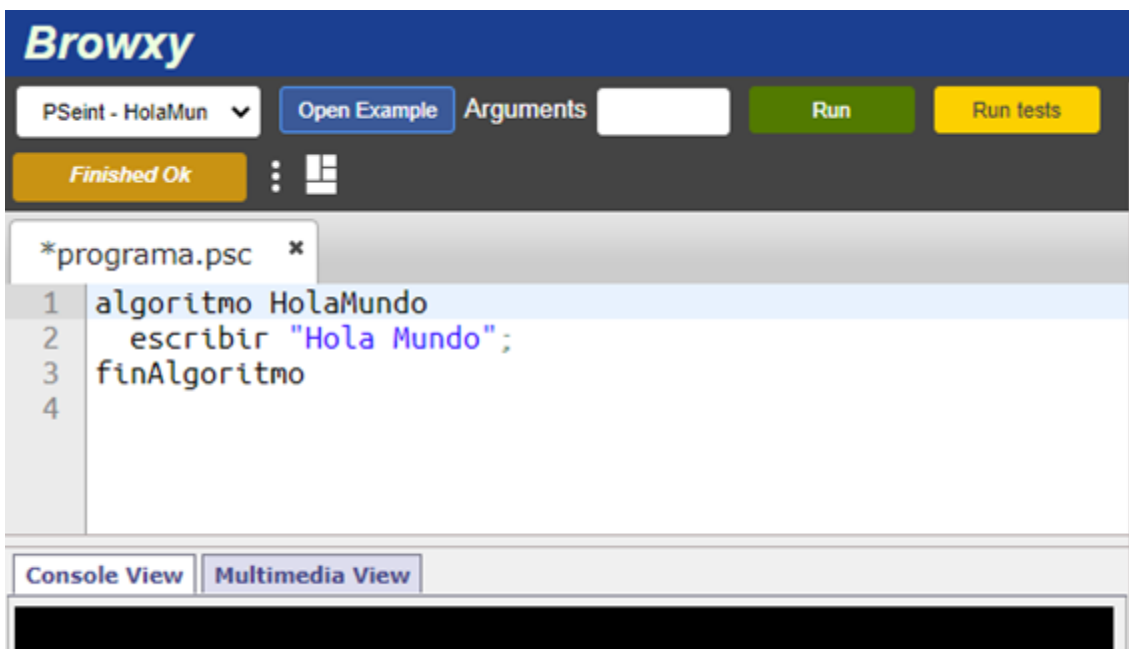


Figura 4.17 Interfaz de Browxy (fuente: HTTP5)

Además, algunos editores permiten enviar como parámetro el código realizado en su interfaz desde un proyecto independiente, devolviendo su resultado. Varios de los compiladores/intérpretes presentan un listado de lenguajes de programación como, por ejemplo: C, Java, Php, Phyton, etc.

4.4.3 Proceso de ejecución

Como se observa en la Figura 4.18, el estudiante teclea el código fuente, pasando de manera inicial a la exploración, donde se reconocen y omiten caracteres adicionales a la programación, para dar paso al proceso de análisis léxico, sintáctico y semántico.

Reconociendo las instrucciones, símbolos y valor, según el análisis, y soportado en la tabla de símbolos, se procede a la ejecución del código, resultado que es presentado al usuario en pantalla.

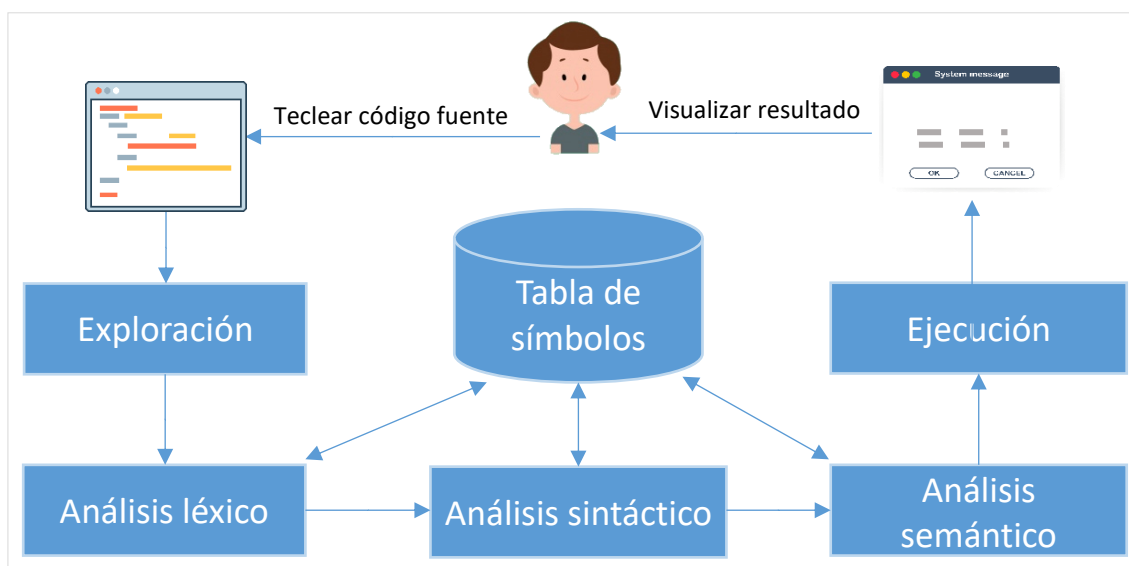


Figura 4.18 Proceso de ejecución del código (fuente: propia)

Finalmente, se deberá brindar la posibilidad de que el entorno de programación del compañero de aprendizaje cuente con un depurador de código. Éste debe permitir a los estudiantes encontrar los lugares de su código fuente donde han cometido algún error, con la finalidad de poder corregirlo y volver a ejecutarlo.

4.4.4 Proceso de depuración

Es importante que el compañero de aprendizaje de programación brinde este acompañamiento de una manera efectiva, para poder ayudar a los estudiantes en la resolución de los ejercicios de programación (Morales et al., 2020).

En este contexto, Ocaña et al., (2020) mencionan que mientras los estudiantes envía a ejecutar su código, el compañero de aprendizaje depurará posibles errores y le brindará ayuda en caso cuando sea necesario.

Como se puede visualizar en la Figura 4.19, se inserta el proceso de depuración previo a la ejecución del código fuente. Este proceso de depuración presenta al usuario en forma de mensajes de texto, esto puede ser de manera inmediata (predictiva), al momento que el usuario va tecleando su código, como al momento que envía a ejecutar el código.

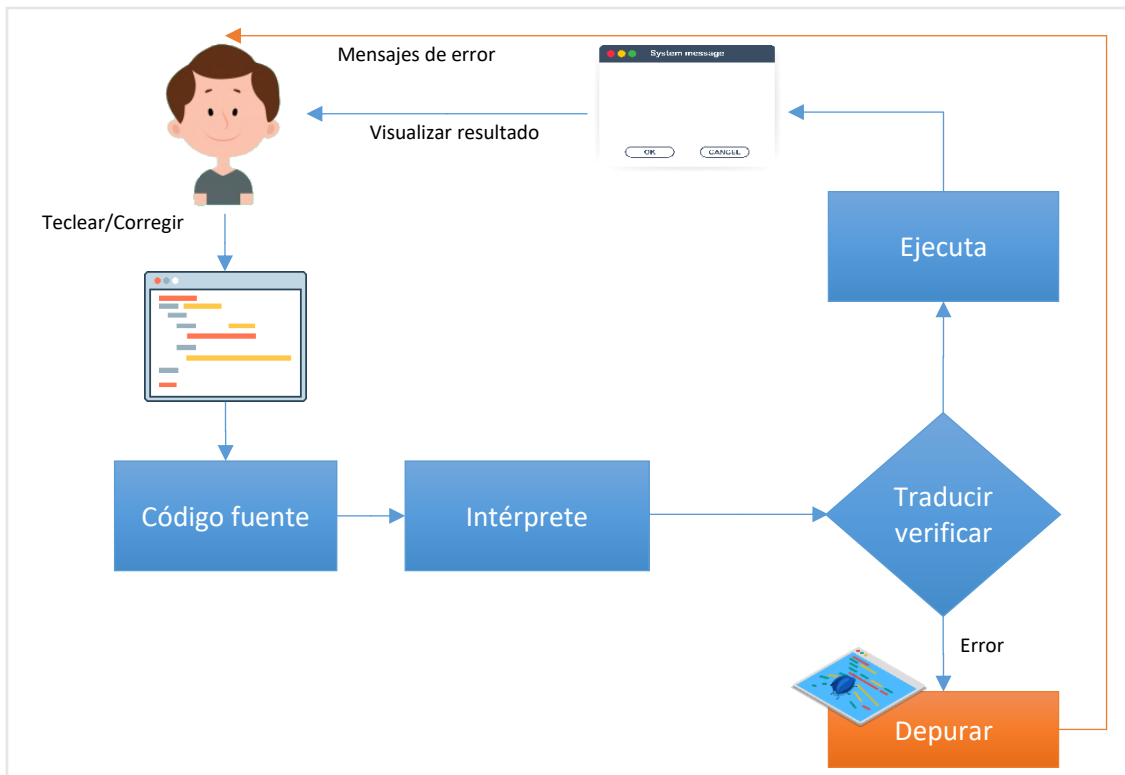


Figura 4.19 Proceso de depuración (fuente: propia)

De esta manera la depuración permite al usuario ir corrigiendo los errores presentados en las líneas detectadas. Posteriormente se realizan estas correcciones y el usuario puede nuevamente volver a ejecutarlo hasta que su código quede libre de errores.

En este sentido, se debe considerar el establecer log's de los ejercicios desarrollados en el editor de programación, a fin de poder capturar errores y poder controlarlos de manera personalizada.

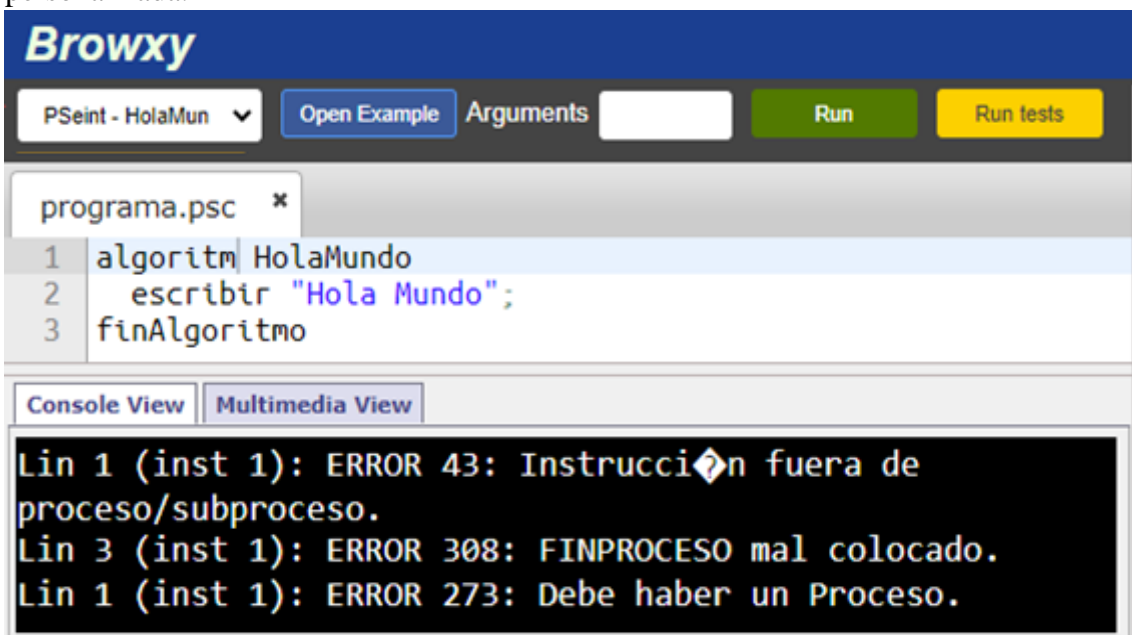


Figura 4.20 Mensajes de depuración de Browxy (fuente: HTTP5)

Varios de los ejemplos de compiladores/intérpretes, mencionados anteriormente, cuentan con un depurador. Como es el caso de: Ideone, CodePad, Browxy. En la Figura 4.20, se visualiza el proceso de depuración en la interfaz de Browxy.

4.5 Fase 5: Validación de la interfaz y de la funcionalidad

4.5.1 Validación de la interfaz

Esta parte de la metodología se encuentra encauzada a la validación de la interfaz, tomando decisiones de diseño del entorno del compañero de aprendizaje, para lo cual se considera la realización de prototipos que se dieron en el co-diseño.

De manera específica, se deberán realizar reuniones en las que se muestren los avances del diseño para poder socializarlo con los usuarios del proyecto (especialistas, docentes, y estudiantes).

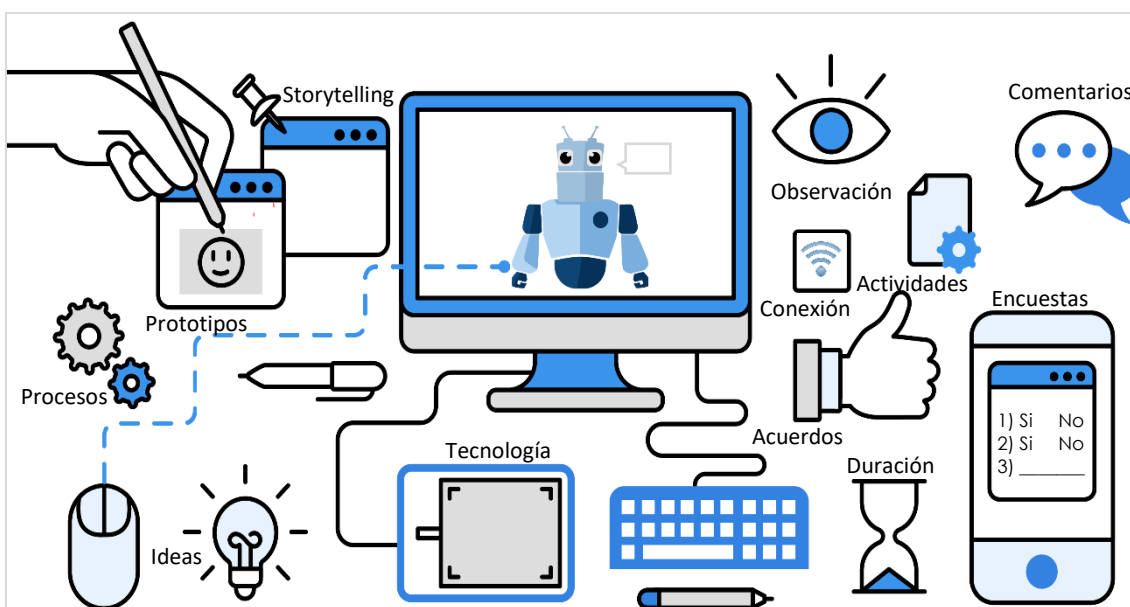


Figura 4.21 Adaptación en la interfaz de requisitos de usuarios y co-diseño (fuente: propia)

Como se puede observar en la Figura 4.21, cada requisito que se presenta mediante prototipos, encuestas o comentarios los docentes y estudiantes, deben irse adaptando en la construcción del diseño del compañero de aprendizaje.

De igual manera contribuye en este proceso de adaptación, la observación que hace el diseñador, detectando posibles omisiones de los docentes y estudiantes, así como también las experiencias previas en la construcción de otros compañeros de aprendizaje.

Al ser MEDIE_GEDILEC un ciclo iterativo e incremental, al igual que las otras fases, la interfaz seguirá evolucionando hasta que se consiga la aceptación por parte de los usuarios.

4.5.2 Validación de la funcionalidad

En este punto se validará la funcionalidad del compañero de aprendizaje, el equipo de desarrollo deberá probar el comportamiento que tendrán los módulos del entorno de aprendizaje y la interacción con el compañero de aprendizaje. En particular, se comprueba el correcto desempeño de la programación en la gestión de usuarios, contenidos de aprendizaje, gestor de diálogo, editor de código, ejecución y depuración.

Aquí se aborda la codificación de la funcionalidad del compañero de aprendizaje, así como la validación de ésta. Los algoritmos de estos módulos se pueden expresar mediante el uso de diagramas de flujo.

Es importante que las pruebas de funcionalidad se desarrollen en entornos de simulación similares al medio donde se realizarán en la práctica real del compañero de aprendizaje.

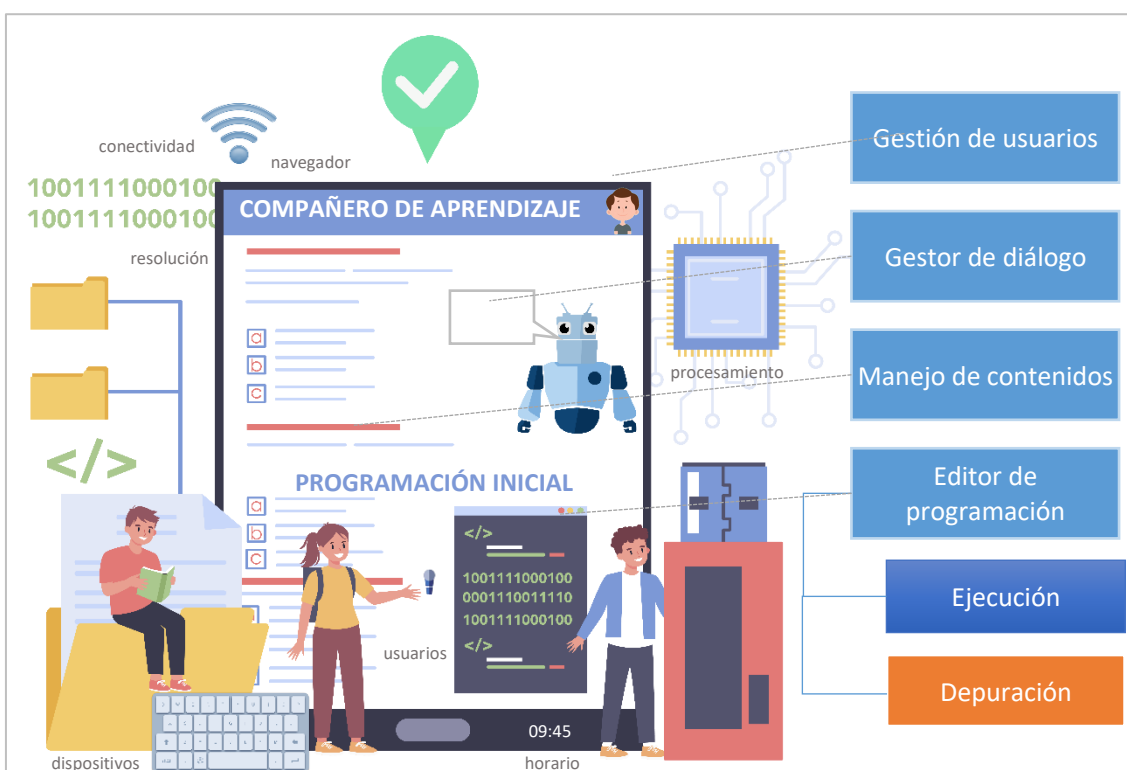


Figura 4.22 Entorno de simulación, para la validación de funcionalidad (fuente: propia)

Como se observa en la Figura 4.22, se debe considerar características tecnológicas de los equipos informáticos, dispositivos, conexión de Internet, así como también es importante integrar a este proceso de validación a los usuarios.

A fin de evitar sesgos en los resultados del proceso de validación, de manera adicional a los tecnológicos, se deben realizar en condiciones similares en torno a la edad de los estudiantes de prueba, conocimientos previos, tiempos de interacción, etc.

De manera adicional, para garantizar el funcionamiento del compañero de aprendizaje, tanto en sesiones presenciales, como sesiones virtuales, se recomienda durante la

construcción del compañero de aprendizaje, el uso de herramientas en línea que permiten realizar pruebas de funcionamiento.

Los parámetros que generalmente se miden dentro del análisis de funcionamiento, son:

- Tiempo de interacción.
- Índice de velocidad de carga.
- Bloqueos.
- Tamaño de archivos multimedia.
- Uso de cookies.
- Https.
- Visualización de dispositivos y resoluciones.

Entre las herramientas en línea que pueden utilizarse se encuentran:

- PageSpeed.
- Pingdom.
- Pagespeed.
- GTMetrix.
- Screenfly.

4.6 Fase 6: Sesiones prácticas

En la fase de sesiones prácticas, se integra en el aula al compañero de aprendizaje, presentándolo a los estudiantes, por parte de los diseñadores, y en presencia y dirección de los docentes.

Desde la primera sesión es importante la observación directa de la interacción estudiante - compañero de aprendizaje por parte de los diseñadores, desarrolladores y docentes, identificando reacciones e impresiones de los estudiantes.

Se verifican aspectos de facilidad de uso, aceptación, motivación, entre otros, que contribuyen con el cumplimiento de los objetivos propuestos.

El diseño Web del compañero de aprendizaje, facilita su integración en el aula, debiendo por parte de los docentes establecer contenidos de programación inicial en Educación Primaria, número de sesiones y distribución del tiempo.

En este sentido, se deben ir registrando las sesiones prácticas a detalle, la Figura 4.23 muestra las modalidades presencial y en línea de las sesiones prácticas.



Figura 4.23 Parámetros de sesiones prácticas (fuente: propia)

4.6.1 Sesiones presenciales

1. Los estudiantes, asisten al laboratorio para la clase de programación, el tiempo establecido para la misma.
2. Los estudiantes hacen uso del ordenador asignado en el laboratorio de computación.
3. Los estudiantes introducen su usuario y contraseña al sistema interactivo educativo.
4. Durante unos minutos, los estudiantes interactúan con el compañero de aprendizaje en las actividades propuestas.
5. El docente de ser necesario, dará un breve repaso del tema anterior, construyendo conocimiento con los estudiantes.
6. El docente debe presentar el nuevo tema, soportado en presentaciones o tutoriales.
7. La información de los tutoriales debe contener conocimiento teórico, práctico y preguntas interactivas.
8. Bajo la supervisión del docente, los estudiantes realizan ejercicios de programación propuestos sobre el tema tratado en el editor de código.
9. Las evaluaciones de aprendizaje deben contar con un tiempo máximo establecido.
10. Los docentes pueden contar con la posibilidad de revisar la resolución de ejercicios en el sistema interactivo educativo.
11. Para las encuestas de satisfacción se recomienda el uso de formularios en línea.

4.6.2 Sesiones virtuales

1. Los estudiantes usan su ordenador personal desde sus hogares.
2. Los estudiantes se conectan por un tiempo establecido a la videoconferencia para clase (se recomienda encender la Webcam).
3. Los estudiantes acceden en el horario establecido, con su usuario y contraseña al sistema interactivo educativo.
4. Durante unos minutos, los estudiantes interactúan con el compañero de aprendizaje en las actividades propuestas.
5. El docente, de ser necesario, dará un breve repaso del tema anterior construyendo conocimiento con los estudiantes.
6. El docente debe presentar el nuevo tema, soportado en presentaciones o tutoriales.
7. La información de los tutoriales debe contener: conocimiento teórico, práctico y de preguntas interactivas.
8. Los estudiantes realizan ejercicios de programación propuestos sobre el tema tratado en el editor de código.
9. Las evaluaciones de aprendizaje deben contar con un tiempo máximo establecido.
10. Los docentes pueden contar con la posibilidad de revisar la resolución de ejercicios en el sistema interactivo educativo.
11. Para las encuestas de satisfacción se recomienda el uso de formularios en línea.

4.6.3 Interfaces para la integración

Para el proceso de integración en el aula, se considera en la propuesta del compañero de aprendizaje, las siguientes interfaces:

4.6.3.1 Inicio de sesión

La Figura 4.24 muestra el inicio de sesión para usuarios registrados. Este debe estar asociado al horario de clase del curso de programación inicial.

La imagen muestra una interfaz de inicio de sesión con un fondo gris claro. En la parte superior hay un ícono de usuario en un círculo azul. Debajo, el título "Ingreso a Usuarios" está centrado. Hay dos campos de entrada: el primero está etiquetado "Usuario" y tiene un ícono de persona a la derecha; el segundo contiene siete puntos y tiene un ícono de llave a la derecha. Debajo de los campos hay un botón azul con el texto "ACCEDER" en blanco.

Figura 4.24 Ingreso de usuarios (fuente: propia)

4.6.3.2 Opciones generales

En la Figura 4.25 se muestra la interfaz de opciones generales, con una distribución propuesta para realizar el diálogo con el compañero de aprendizaje y el estudiante.

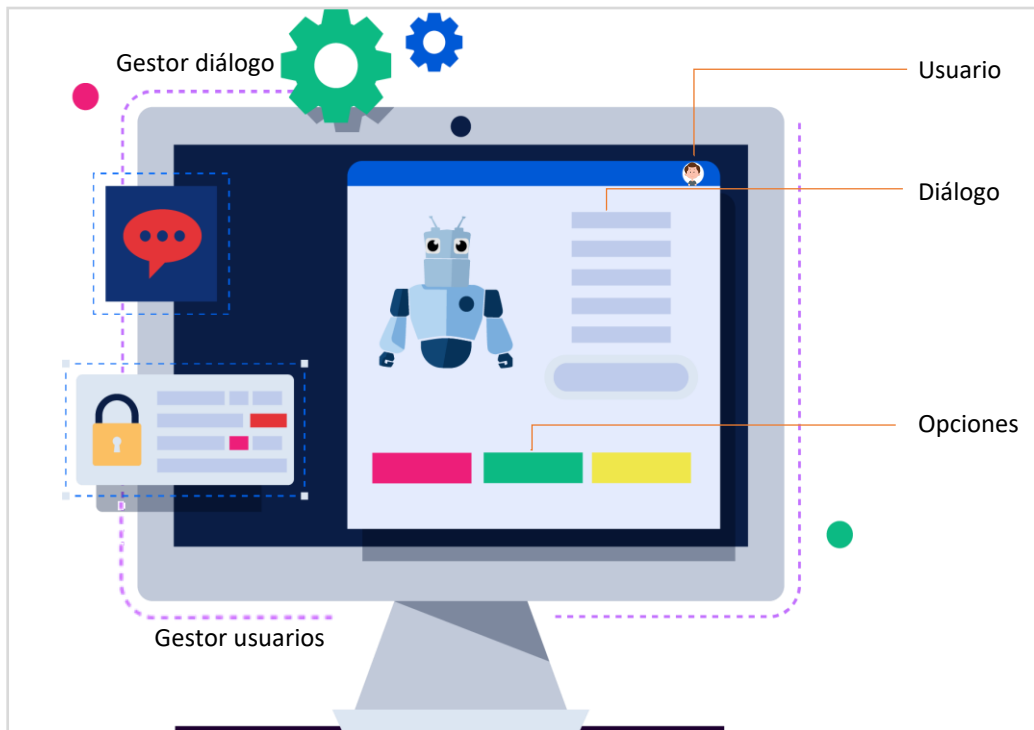


Figura 4.25 Interfaz general del compañero de aprendizaje (fuente: propia)

4.6.3.3 Tutoriales

Los tutoriales deben invocarse por parte del estudiante a través del gestor de diálogo, debiendo ser un compendio de información estructurada (contenido de aprendizaje, videos, ejercicios, etc.), como se observa en la Figura 4.26.



Figura 4.26 Tutoriales de aprendizaje (fuente: propia)

Para el diseño de los contenidos de aprendizaje, se propone considerar aspectos de la gamificación, ya que permite experimentar el aprendizaje con motivación a los estudiantes captando su atención (Morales et al., 2021).



Figura 4.27 Pasos principales de la Gamificación (fuente: propia)

Como muestra la Figura 4.27, la Gamificación promueve la participación del estudiante en la actividad de aprendizaje, activando la creatividad y la búsqueda de soluciones creativas en la resolución de ejercicios.

En este sentido, en la Gamificación es importante motivar a los estudiantes, reconociendo sus logros, generando en ellos la intención de repetir el aprendizaje, hasta lograr mejorar su resultado.

4.6.3.4 Editor de programación

En la Figura 4.28, se observa la interfaz del editor de programación, esta permitirá al estudiante practicar los contenidos de aprendizaje en el curso de programación inicial, en Educación Primaria.

En esta interfaz, se juntan los módulos principales del compañero de aprendizaje, como son: Gestor de usuarios, Gestor de diálogo, Editor de programación (ejecución y depuración).

De manera adicional, se debe brindar acceso a los docentes para visualizar la resolución de ejercicios de los estudiantes, a fin de que puedan valorar el avance del aprendizaje individual de los estudiantes, en cada contenido de programación.

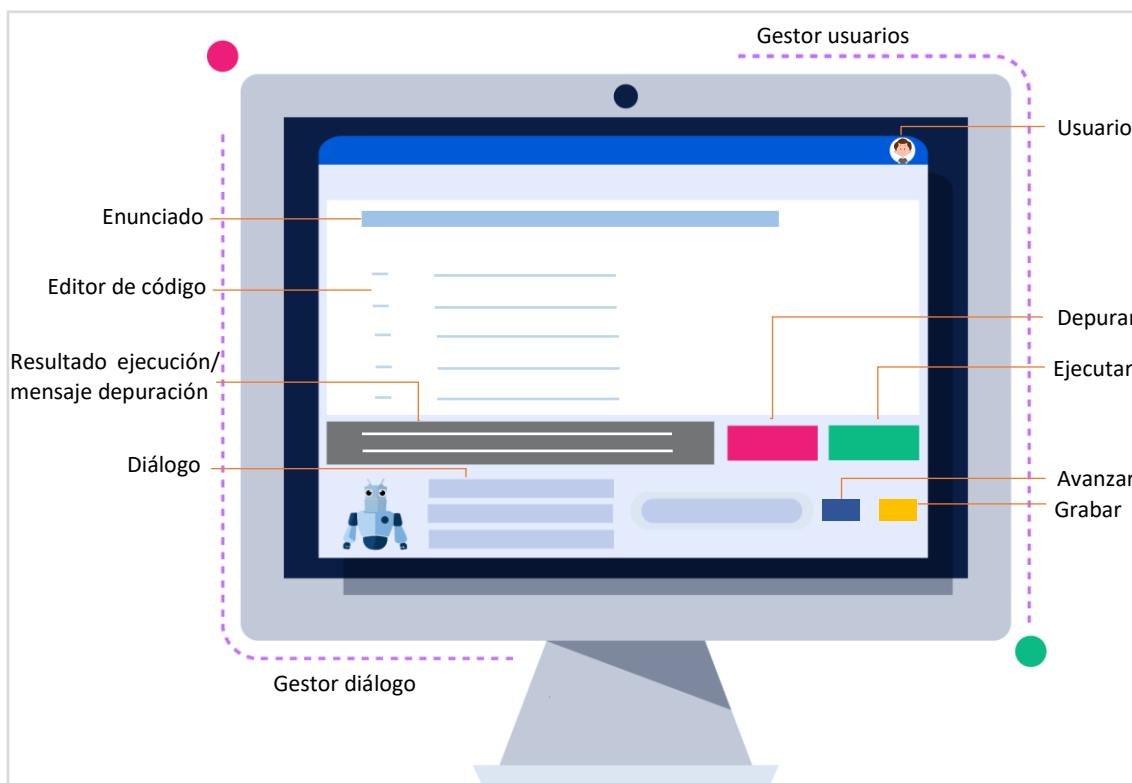


Figura 4.28 Cuestionarios (fuente: propia)

Finalmente, es importante para corregir y/o mejorar las funcionalidades y la integración del compañero de aprendizaje en el aula, y consultar a los estudiantes sobre sus niveles de satisfacción.

4.6.3.5 Encuestas de satisfacción

Es recomendable combinar técnicas de observación directa, con el uso de encuestas, sean físicas o en línea. Para ellos se deberá consultar tanto a docentes como a estudiantes.

En la Figura 4.29, se observa la interfaz del formulario de encuesta en línea, para medir estos niveles de satisfacción, para es recomendable utilizar distintos tipos de preguntas, tales como selección múltiple o única, verdadero o falso, escala y abierta.

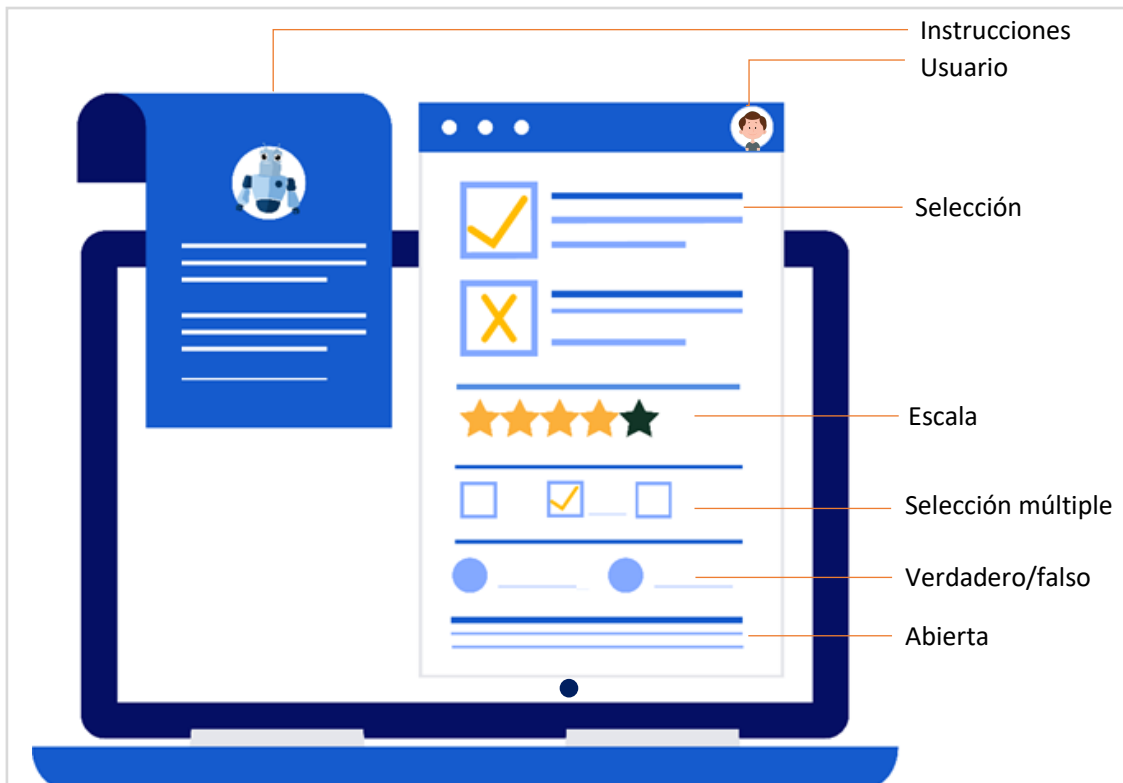


Figura 4.29 Encuestas de satisfacción (fuente: propia)

Capítulo 5. Validación de la metodología

La metodología MEDIE_GEDILEC, descrita en el Capítulo 4, se valida mediante el desarrollo del compañero de aprendizaje llamado Alcodey. El nombre de Alcodey viene de (ALgoritmo - CODIGO) y su integración en el aula. Su objetivo es la enseñanza de programación a niños de Educación Primaria.

En la Figura 5.1 se observa las fases de la metodología MEDIE_GEDILEC en la aplicación del compañero de aprendizaje Alcodey para la enseñanza de programación a niños en dos escuelas: una en España y otra en Ecuador.

El proceso iterativo de la metodología MEDIE_GEDILEC ha permitido que el experimento del compañero de aprendizaje Alcodey, cuente con una mejora continua que ha permitido obtener una adecuada integración en el aula.

La Figura 5.1 muestra un breve recorrido de los prototipos, recursos y de la interfaz, que plasma gráficamente los cambios que se han realizado en Alcodey durante el proceso de diseño actual.



Figura 5.1 Aplicación de MEDIE_GEDILEC en compañero de aprendizaje Alcodey (fuente: propia)

En la Figura 5.2, se visualiza el sistema interactivo educativo con el compañero de aprendizaje, Alcody. De manera general el entorno consta de: Gestión de usuarios (validan los usuarios y las condiciones de acceso); contenidos de aprendizaje (definiciones, videos y ejercicios); gestor de diálogo (análisis de diálogo y respuesta); y editor de código (ejecución y depuración).

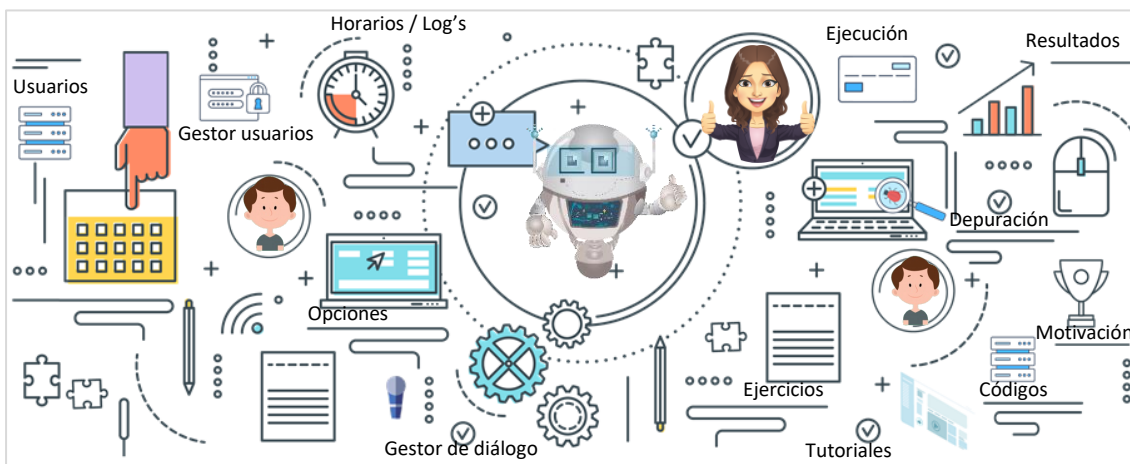


Figura 5.2 Sistema interactivo educativo del compañero de aprendizaje Alcody (fuente: propia)

5.1 Fase 1: Comunicación y co-diseño: docentes y estudiantes.

5.1.1 Comunicación con docentes.

Como primera etapa en el diseño de la interfaz para el análisis de requisitos, se entrevistó a un docente de una escuela de Educación Primaria en Madrid - España, y dos docentes de una escuela de Educación Primaria – Latacunga –Ecuador, quienes plantearon rasgos iniciales del sistema interactivo educativo.

Estos manifestaron que el compañero de aprendizaje debería presentar una estructura sencilla constituida por formas básicas y que tuviera colores pasteles. Su visión dio un punto de partida para la presente investigación, como se observa en la Figura 5.3.



Figura 5.3 Diseño de Alcody en base a las primeras entrevistas con docentes (fuente: Morales et al., 2019)

En las primeras reuniones con los docentes se levantaron requisitos, esto se realizó por entrevistas y observación directa.

En las Tablas 5.1 y 5.2, se describe la información inicial recolectada, en las escuelas de España y Ecuador, respectivamente.

Consulta	Valor	Observación
País	España	
Nivel educativo	Educación Primaria	
Año escolar	Cuarto y quinto año	Año escolar 2018-2019
Rango de edad	8 a 10 años	
Género	Mixto (niñas y niños)	
Conocimiento previo	Scratch	
Contenidos de aprendizaje	Entrada y salida Condicionales / Bucles	
Duración de la sesión	60 minutos	
Sesiones por semana	1	
Cantidad de estudiantes por aula	25 25	13 niñas / 12 niños 13 niñas / 12 niños
Cantidad de ordenadores por aula	23	1 ordenador (docente) 21 ordenadores (niños)
Tipo de ordenadores	Core i3	No disponen de cámaras
Conexión de Internet	Si	200Mbps
Sistema operativo	Linux	
Navegadores de internet	Chrome	

Tabla 5.1 Características y requisitos en escuela de Educación Primaria – España
(fuente: propia)

Consulta	Valor	Observación
País	Ecuador	
Nivel educativo	Educación Primaria	
Año escolar	Octavo año	Año escolar 2019-2020
Rango de edad	8 a 10 años	
Género	Mixto (niñas y niños)	
Conocimiento previo	Scratch	
Contenidos de aprendizaje	Entrada y salida Condicionales / Bucles	
Duración de la sesión	40 minutos	10 minutos para reflexión
Sesiones por semana	2	
Cantidad de estudiantes por aula	34 (Octavo C) 35 (Octavo D)	14 niñas / 20 niños 14 niñas / 21 niños
Cantidad de ordenadores por aula	41	1 ordenador (docente) 40 ordenadores (niños)
Tipo de ordenadores	Core i3, 4Gb	No disponen de cámaras
Conexión de Internet	Si	60Mbps
Sistema operativo	Windows 10	
Navegadores de internet	Chrome, Firefox	

Tabla 5.2 Características y requisitos en escuela de Educación Primaria – Ecuador
(fuente: propia)

5.1.2 Co-diseño con niños

Se trabajó con 66 estudiantes en la escuela de España y 69 estudiantes en la escuela de Ecuador, con edades comprendidas de 8 a 10 años, por medio de observación, entrevistas, encuestas y prototipos.

Para definir los requisitos, los niños se convirtieron en catalizadores y generadores de ideas; y definitivamente cambiaron la dinámica de grupo y de los diseños. Una de las herramientas que se utilizó para esto fueron las encuestas.

Las Figuras 5.4 y 5.5, muestran las encuestas en línea y en papel respectivamente, para el co-diseño con niños. Se les realizaron 10 preguntas a fin de obtener información de cómo perciben al personaje, conocimiento general de programación, el tiempo de conexión, modo interacción con el compañero de aprendizaje, etc.

¿Qué es programar?

Hacer algo en el ordenador

¿Te gustaría aprender a programar?

Si

No

¿Te apetece que te enseñe a programar?

Si

No

SIGUIENTE

Figura 5.4 Encuesta en línea para el co-diseño con niños (fuente: propia)

Preguntar (<http://iprog.site/>)

- ¿Qué es programar? es hacer programas con el ordenador como el social
- ¿Te gustaría aprender a programar? me gustara
- ¿Te apetece que te enseñe a programar? Si me gustaria
- ¿Prefieres que te enseñe Scratch o te gustaría probar a aprender conmigo? aprender con tigo
- ¿Te gusta aprender a programar como si fuera cocinar? Si
- ¿Te gusta aprender a programar como si fuera escribir un cuento? Si
- ¿Prefieres aprender a programar como si fuera cocinar o como si fuera escribir un cuento? Si
- ¿Cómo te gustaría que hablase contigo? ¿qué tipo de ejercicios te gustan? Alegre mate
- Si no entiendes algo, ¿cómo te gustaría que te lo mostrase? Explicandda con otra ordenador
- ¿Cuánto tiempo te gustaría practicar conmigo a la semana? ¿en casa, en clase o en ambos sitios? on clase 2 veces y

Figura 5.5 Encuesta en papel para el co-diseño con niños (fuente: propia)

En la Tabla 5.3 se presentan los rasgos que aproximan la propuesta de diseño del compañero de aprendizaje, con una cabeza, tronco y extremidades como base para el diseño; además se toma en cuenta que el carácter del personaje debe ser calmado, y que debe comunicarse con los niños en un tono amigable.

Personaje	Género	Color	Características
ordenador (25)	*no define (34)	*no define (54)	deporte (7)
persona (19)	masculino (20)	varios colores (5)	*no define (6)
robot (7)	femenino (7)	Rosado (2)	Hablar (5)
animal (6)			Características humanas (brazos, boca, piernas) (4)
transporte (2)			característica de animales (2)
			bailar (2)

Tabla 5.3 Rasgos de características del compañero de aprendizaje (Morales et al., 2019)

Con la información recopilada y en relación al manejo de los estudiantes con medios y dispositivos, se define una estructura simplificada y tecnológica para el compañero de aprendizaje, al cual se lo denomina Alcody.

En este sentido, se inicia el diseño de un compañero de aprendizaje que enseñe a programar, considerando el análisis de la percepción de los estudiantes, mediante el co-diseño del personaje, adaptada a las preferencias y sugerencias de los niños (ver Figura 5.6).

Alcody tiene una forma compacta donde los elementos más representativos resaltan como los ojos y la pantalla del cuerpo, mientras que los elementos complementarios que no interactúan en la interfaz se los ha simplificado para facilitar la interacción del personaje.

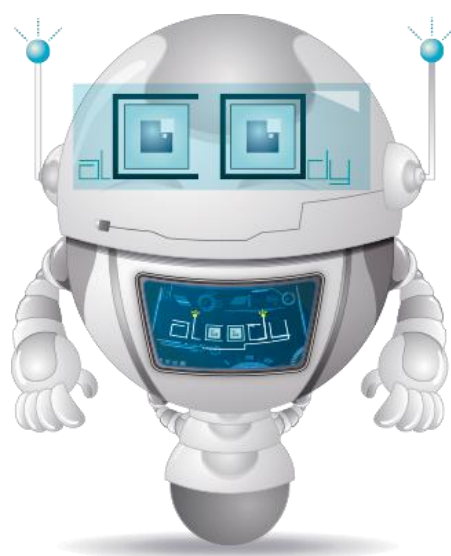


Figura 5.6 Compañero de aprendizaje, Alcody (Morales et al., 2019)

La forma general es ovoide, con detalles y complementos redondos; se utilizan figuras básicas y simplificadas que definen la forma del personaje; los trazos son finos y no tienen gran impacto en la estructura global. Los pies se simplifican porque no tiene interacción dentro de las actividades de la interfaz.

El entorno va asociado a la personalidad de Alcodey, por tanto, se hace un cambio que contrastan sobre el fondo e ilustraciones coloridas. La interfaz denota tecnología donde se manejan líneas y formas muy sutiles que no interfieren con las actividades lúdicas del personaje y el usuario.

Los fondos son neutros y se prioriza los espacios de información con textos simples contrastantes e ilustraciones coloridas. A cada tipo de información se le asigna un color para ser identificado con mayor facilidad y rapidez (ver Figura 5.7).



Figura 5.7 Interfaz general de Alcodey (fuente: propia)

5.2 Análisis de diálogos

Para poder analizar el diálogo Alcodey - estudiante, fue necesario comprender cómo les gustaría a los estudiantes hablar con el compañero de aprendizaje, planteándose las siguientes preguntas (ver Tabla 5.4):

No.	Pregunta
RQ1	¿Qué tema les gustaría a los estudiantes hablar con su ordenador?
RQ2	¿Los estudiantes usarían preguntas para hacer actividades en el ordenador? ¿Cómo? ¿Cuántas?
RQ3	¿Los estudiantes usarían exclamaciones para preguntar al ordenador? ¿Cómo? ¿Cuántas?
RQ4	¿Los estudiantes usarían oraciones cortas / medias / largas?
RQ5	¿Los estudiantes saben muchas palabras relacionadas con la programación?
RQ6	¿Los estudiantes consideran el ordenador como su amigo?
RQ7	¿Hay diferencias significativas entre los cursos?

Tabla 5.4 Parámetros del análisis de diálogos (Ocaña et al., 2019)

5.2.1 Storytelling

Usando la técnica de prototipado, Storytelling, se da libertad a los estudiantes para que en una hoja de papel creen una historia que describa su diálogo con el compañero de aprendizaje, receptando 66 historias que contribuyen con el proceso de análisis de diálogo (Morales et al., 2017).

En este sentido, se pidió a los estudiantes que escribieran un diálogo breve (máximo de 1 página) con Alcodey en el formato que eligieran. De manera intencional se dio libertad a los estudiantes para que pudieran ser creativos.

Debido a la política de privacidad de la escuela sobre los datos personales de los niños, no fue posible recopilar ninguno, sino solo sus diálogos anónimos.

5.2.1.1 Storytelling aplicado a estudiantes de cuarto año

El estudio se dividió en dos grupos: del primero se obtuvieron 17 historias de los niños de cuarto grado. La primera observación interesante es que cuando se les dio libertad para escribir, los niños escribieron cuentos de fantasía y cuentos personales. Solo uno de ellos escribió un diálogo real (ver Figura 5.8).

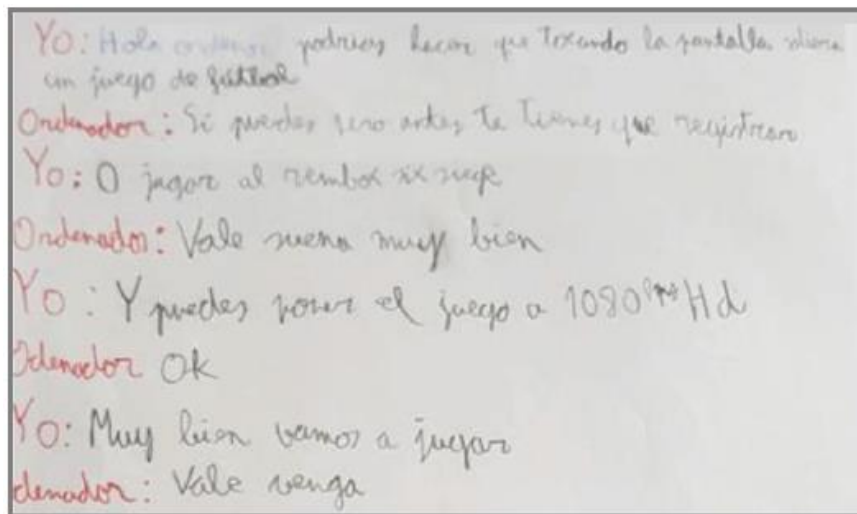


Figura 5.8 Stortelling, diálogo con Alcody, estudiante cuarto año (Ocaña et al., 2019)

Detallando la Figura 5.8 se puede visualizar el siguiente dilogo:

(I) le pregunta al Ordenador.

(C) si puede ejecutar un juego de fútbol en la pantalla.

(C): Sí, pero debes registrarte.

(I): Yo también quiero jugar.

(C): OK, suena genial.

(I): ¿Puedes jugar el juego a 1080 HD?

(C): OK.

(I): Vamos a jugar.

(C): OK.

Además, pese a que a los estudiantes se les dijo que el objetivo es aprender a programar, muchos optaron por escribir sobre programas ya existentes:

Programas deportivos (5 estudiantes), videojuegos (5 estudiantes), realidad virtual (2 estudiantes), controles (1 estudiante), coches (1 estudiante), películas (1 estudiante), animales (1 estudiante) y robots (1 estudiante).

Solo dos estudiantes hicieron preguntas al ordenador, como “¿Puedo jugar fútbol contigo?”; y un estudiante usó la exclamación “¡Has ganado!”.

Con respecto a la longitud de las oraciones, no se encontraron diferencias claras, ya que 6 estudiantes usaron oraciones cortas, mientras que 7 estudiantes usaron oraciones largas, y 4 estudiantes oraciones de longitud media.

Por lo tanto, se puede decir que escriben desde oraciones cortas hasta oraciones largas, y el agente debe estar preparado para manejar los errores ortográficos y gramaticales. Incluso para el lector humano fue difícil entender algunos errores gramaticales y algunas de las palabras debido a la mala ortografía de los niños.

Con respecto al vocabulario, los niños utilizaron muy pocas palabras relacionadas con la informática. Normalmente solo se utilizaron palabras como robot, videojuego, programa, ordenador o pantalla.

En general, cuando utilizaban algún término informático, era bastante sofisticado, como realidad virtual, gafas 3D, 1080 HD. Los niños también sabían nombres específicos de videojuegos como Clash Royale o Minecraft, incluso a pesar de su corta edad.

Durante la construcción de los diálogos de manera general, los estudiantes mostraron expresiones amistosas al momento de desarrollar las historias. En particular, de lo analizado, cuatro estudiantes hablaron con el agente como si fuera su amigo.

5.2.1.2 Storytelling, aplicado a estudiantes de quinto año

En la aplicación del estudio con el segundo grupo de estudiantes de quinto año, 49 entregaron su trabajo. Cinco estudiantes escribieron diálogos similares al que se muestra en la Figura 5.9.

Dos estudiantes eligieron contar una historia personal, dos estudiantes escribieron algo sin ninguna estructura y sin relación con los ordenadores, y el resto escribió historias relacionadas con Ciencias de la Computación como creadores o usuarios de programas.

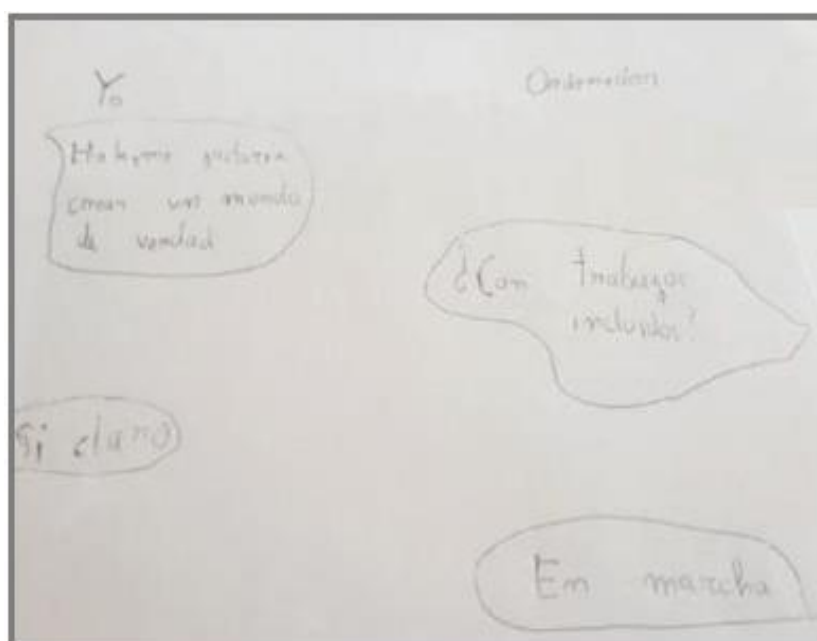


Figura 5.9 Storytelling, diálogo con Al Cody, estudiante quinto año (Ocaña et al., 2019)

La lista de temas ordenados por mayor uso por parte de los estudiantes fue:

Deportes (8 estudiantes), videojuegos (7 estudiantes), amistad (6 estudiantes), animales (6 estudiantes), fantasía (5 estudiantes), ordenadores (5 estudiantes), deseos (3 estudiantes), música (3 estudiantes), para crear un mundo (2 estudiantes), automóviles (1 estudiante), cocinero (1 estudiante) y no relacionado (2 estudiantes).

12 estudiantes hicieron preguntas al ordenador como “¿Podemos programar un juego?” o “¿Qué es un programa?”; y 6 estudiantes usaron exclamaciones como “¡El ordenador habla!” o “¡Me encanta mi ordenador!”.

Con respecto a la longitud de las oraciones (sin considerar las dos historias no relacionadas): 25 estudiantes usaron oraciones largas, 18 oraciones de longitud media, y 4 estudiantes usaron oraciones cortas.

Finalmente, con respecto al conocimiento de términos relacionados con el ordenador, 3 estudiantes tienen un conocimiento medio, 35 de bajo conocimiento y el resto no usó ninguna palabra relacionada con la programación. Las palabras más utilizadas fueron: ordenador, Internet, virus, digital, cables, pantalla y programa.

Como en el caso de los estudiantes de cuarto año, los estudiantes de quinto año también usaron palabras sofisticadas como realidad virtual o Scratch, y nombres de videojuegos como Clash Royale. 17 estudiantes hablaron con Alcody como su amigo.

5.2.2 Revisión de diálogos.

Para responder a las preguntas planteadas para el análisis de diálogo, se crearon ocho parámetros, con valores discretos para calcular en cada diálogo. Los casos fueron contados manualmente para evitar errores (ver Tabla 5.5).

Criterios	Posibles Valores
Curso	Cuarto y quinto año (Educación primaria, 8-10 años).
Tipo	Cuento de fantasía, diálogo, cuento personal o programa.
Tema	Cocinero, fantasía, videojuegos, deportes, música, danza.
#Preguntas	Número de preguntas utilizadas por el niño.
#Exclamaciones	Número de exclamaciones utilizadas por el niño.
Longitud	1 (oraciones cortas con menos de 10 palabras). 2 (oraciones medianas con 10-20 palabras). 3 (oraciones largas con más de 20 palabras).
Conocimiento	1 (bajo con menos de 5 palabras relacionadas con ordenadores). 2 (medio con 5-10 palabras relacionadas con los ordenadores). 3 (alto con más de 20 palabras relacionadas con los ordenadores).
Amistad	1 (el niño considera a Alcody como un amigo). 0 (el niño no habla de amistad en absoluto).

Tabla 5.5 Parámetros de análisis de los diálogos Alcody – estudiante (fuente: propia)

De acuerdo con los datos recopilados, se proporcionan las directrices para los parámetros a las preguntas planteadas en el análisis del diálogo entre Alcody – estudiante (ver Tabla 5.6).

No.	Pregunta	Respuesta
RQ1	¿Qué tema les gustaría a los estudiantes hablar con su ordenador?	El tema favorito de los estudiantes es el deporte, seguido de cerca por los videojuegos. Podría ser interesante cambiar la interfaz de Alcody para que no esté tan centrado en la cocina sino en los juegos de estrategia, ya que parece un tema que es más interesante para los estudiantes.
RQ2	¿Los estudiantes usarían preguntas para hacer actividades en el ordenador? ¿Cómo? ¿Cuántas?	Los estudiantes no suelen usar preguntas, y algunos realizan preguntas de qué/cómo.
RQ3	¿Los estudiantes usarían exclamaciones para preguntar al ordenador? ¿Cómo? ¿Cuántas?	Los estudiantes no suelen usar exclamaciones, solo para expresar emociones como “¡Me encanta mi ordenador!”, “¡Muy bien!”
RQ4	¿Los estudiantes usarían oraciones cortas / medias / largas?	Los estudiantes pueden usar oraciones que van de muy cortas a muy largas con muchos errores gramaticales y ortográficos que el agente debe poder manejar.
RQ5	¿Los estudiantes saben muchas palabras relacionadas con la programación?	No utilizan más de cinco palabras diferentes relacionadas con la programación. Aunque conocen algunas palabras sofisticadas y nombres de videojuegos.
RQ6	¿Los estudiantes consideran el ordenador como su amigo?	21 de los 66 estudiantes que proporcionaron una historia relacionada con ordenadores lo consideraron como un amigo.
RQ7	¿Hay diferencias significativas entre los cursos?	No hay muchas diferencias entre los cursos. Aunque, se puede apreciar una variabilidad más léxica relacionada con la programación en el caso de estudiantes mayores, y también una mayor variedad de temas y estructuras en sus historias.

Tabla 5.6 Directrices para los parámetros del análisis de diálogos (fuente: propia)

5.3 Gestor de diálogo

Una vez realizado el análisis del diálogo, el siguiente paso es determinar la manera en que se va a realizar la interacción de la comunicación estudiante – compañero de aprendizaje.

Para la interfaz de comunicación entre el estudiante y Alcody, se adapta la experiencia de los niños relacionada con la forma de comunicarse por medio de aplicaciones digitales, de los textos escritos analizados, donde se observa la estructura de un chat; la misma que ha servido de base para el diseño de la interfaz (ver Figura 5.10).

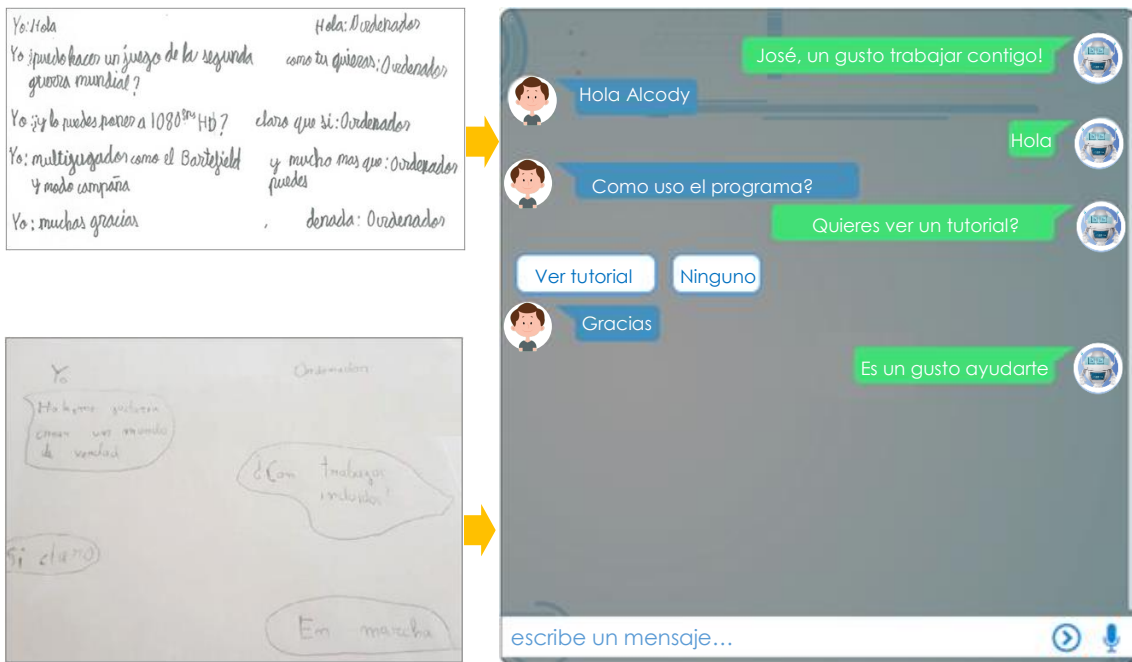


Figura 5.10 Adaptación de prototipos a la interfaz del gestor de diálogo (fuente: propia)

5.3.1 API Text-to-Speech.

La Figura 5.11 describe el diseño de la interfaz del gestor de diálogo en base a prototipos, incluyendo el manejo del micrófono para la invocación a la función del API que permite convertir voz en texto, a fin de enviar al gestor de diálogo la entrada del usuario en forma de texto.

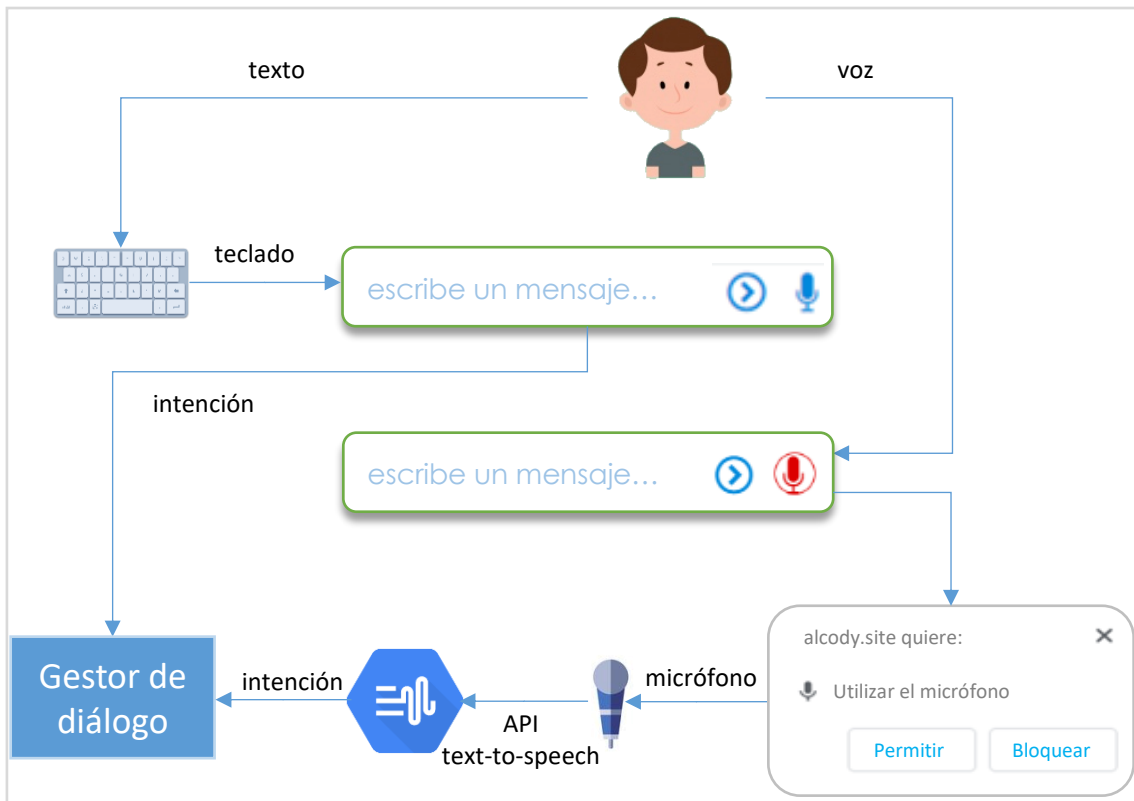


Figura 5.11 Diseño de la interfaz del gestor de diálogo en base a prototipos (fuente: propia)

5.3.2 Desarrollo del gestor de diálogo

Una vez revisado el funcionamiento de AIML y Watson, se diseñó un gestor de diálogo para el compañero de aprendizaje, para lo cual se definieron tablas en la base de datos: intenciones, entidades y respuestas.

Intenciones: Es el objetivo que tiene el estudiante cuando envía un mensaje al compañero de aprendizaje. En la Tabla 5.7 se pueden observar ejemplos de intenciones.

id_intencion	id_respuesta	Intención
1	1	Saludar
2	2	Despedirse
3	3	Ayuda con la pregunta
4	4	Ver tutorial entrada-salida

Tabla 5.7 Ejemplo de intenciones (fuente: propia)

Entidades: Representa la frase que introduce el usuario haciendo mención a una intención. En este sentido se deben colocar las entidades que se considere necesarias para una intención determinada. En la Tabla 5.8 se pueden observar ejemplos de entidades.

id_intencion	id_entidad	Entidad
1	1	Buenos días
1	2	Buenas tardes
1	3	Hola
1	4	Cómo te va

Tabla 5.8 Ejemplo de entidades, para la intención –Saludar- (fuente: propia)

Respuesta: Es el mensaje de salida que se proporciona al usuario, esto en base a la detección de intenciones, con las entidades ingresadas a la BD. Los mensajes de respuesta de Alcodey, no son solo de texto, también incluyen imágenes y/o direccionamientos URL. La Tabla 5.9 muestra ejemplos de respuestas.

id_respuesta	id_intencion	Respuesta	tipo	imagen	enlace
1	1	Hola, USUARIO	mensaje		
2	2	Adiós, USUARIO	mensaje		
3	3	Solución tipo:	tutorial	PregS.png	../guia.html
4	4	Entrada-salida	tutorial		../entrada.html

Tabla 5.9 Ejemplo de respuestas de Alcodey

Al no encontrar una coincidencia en las intenciones de la base de datos, una entrada de usuario produce una respuesta por defecto, al momento se muestra el mensaje: “**retorna a la actividad**”.

En la Figura 5.12 bajo las características antes establecidas para gestionar los mensajes de entrada de los usuarios, se puede observar el funcionamiento del gestor de diálogo del compañero de aprendizaje, Alcodey.

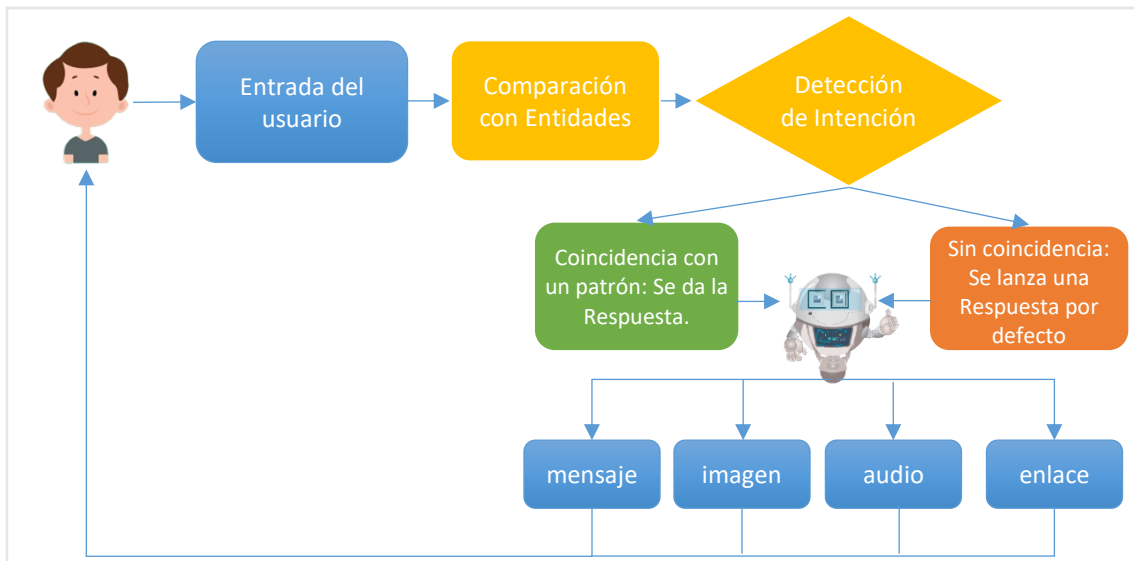


Figura 5.12 Funcionamiento del gestor de diálogo de Alcody (fuente: propia)

En la Figura 5.13, observamos el esquema de funcionalidad del gestor de diálogo de Alcody, que inicia con la entrada por parte del estudiante, por texto o voz, de ser el caso se utilizará la API text-to-speech, necesita un sitio seguro para transformar la voz en texto.

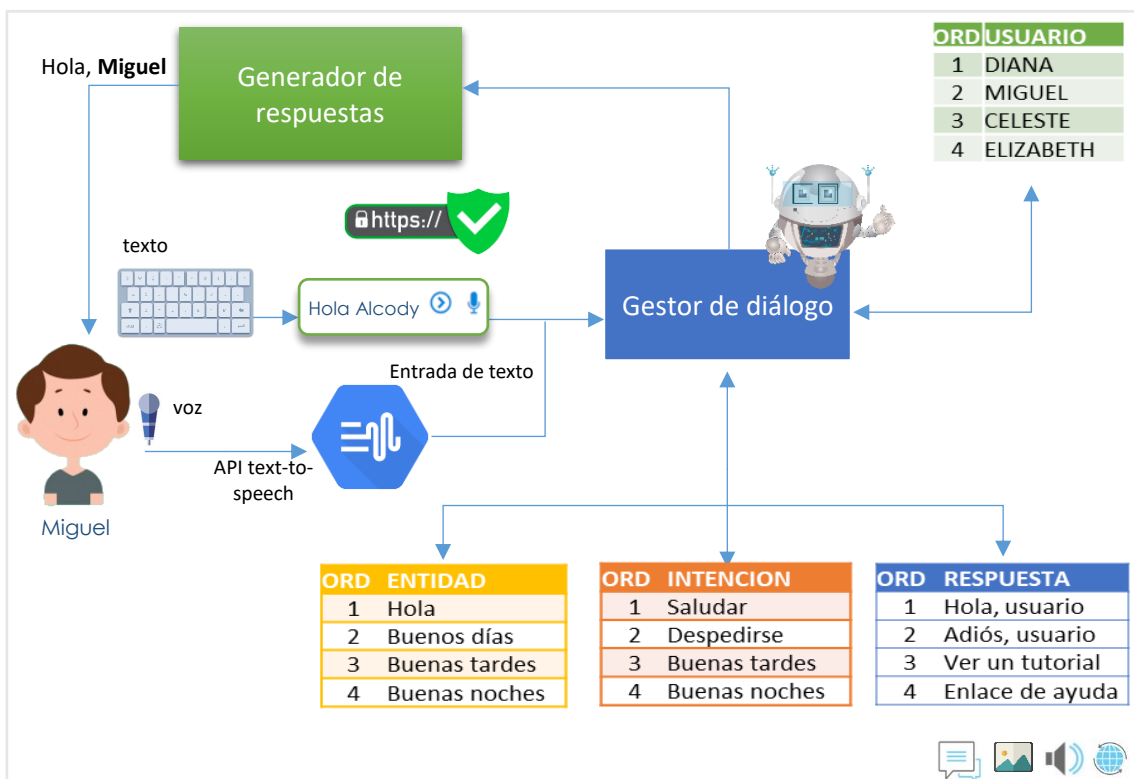


Figura 5.13 Esquema de funcionalidad del Gestor de diálogo (AlcodyGDiag) (fuente: propia)

De esta manera se envía una entrada de texto al gestor de diálogo que la busca entre las entidades, identificando la intención del mensaje, para proporcionar una respuesta por mensaje, imagen, audio o enlace al estudiante.

5.4 Editor de código

Acorde a las indicaciones del co-diseño, se implementa el editor de programación para Alcodey, con una interfaz sencilla permitiendo a los estudiantes el crear pensando, sin tener elementos distractores (ver Figura 5.14).

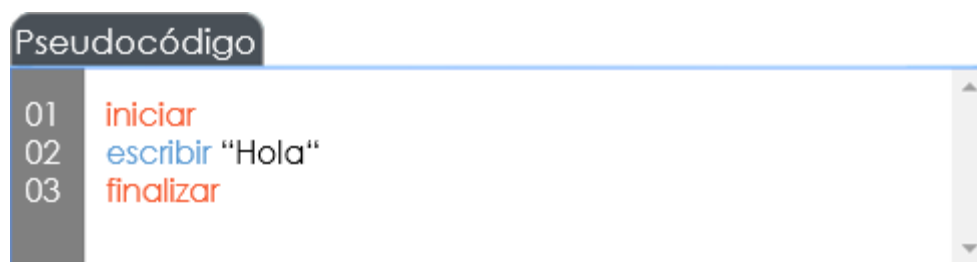


Figura 5.14 Editor de código de Alcodey (fuente: propia)

Este editor de código es parte del entorno del compañero de aprendizaje: Alcodey, con esto no se requiere abrir una interfaz diferente, al tiempo que permitirá tener las mismas funcionalidades, tanto en sesiones presenciales, como para sesiones virtuales.

5.4.1 Lenguaje de programación

En base a las entrevistas con los docentes de las escuelas que participan en el experimento, se selecciona como lenguaje de programación pseudocódigo. Esto es porque integra una serie de palabras con un formalismo muy sencillo, una mezcla de frases del lenguaje común (Mathieu, 2014).

En este sentido, según las entrevistas realizadas con los docentes de programación de las escuelas tanto de España como de Ecuador, a fin de abarcar los contenidos de aprendizaje, se establecen las siguientes instrucciones (ver Tabla 5.10).

Ord	Instrucción	Contenido
1	Iniciar	General
2	Finalizar	General
3	Leer	Entrada – salida
4	Si	Entrada – salida
5	entonces	Estructura condicionales
6	sino	Estructura condicionales
7	para	Estructura bucles
8	hasta	Estructura bucles
9	pasos	Estructura bucles
10	siguiente	Estructura bucles

Tabla 5.10 Instrucciones de pseudocódigo (fuente: propia)

En la Tabla 5.11, se detallan los símbolos para realizar las operaciones y comparaciones del intérprete.

Ord	Operador	Significado
1	=	Igual
2	>	Mayor
3	<	Menor
4	>=	Mayor igual
5	<=	Menor igual
6	<>	Diferente
7	+	Sumar
8	-	Restar
9	*	Multiplicar
10	/	Dividir
11	^	Exponente

Tabla 5.11 Tabla de símbolos (fuente: propia)

Una vez seleccionado el pseudocódigo como lenguaje y establecer las instrucciones, en las sesiones de trabajo con los docentes, se determina el orden del contenido de aprendizajes en referencia a la experiencia con los niños.

5.4.1.1 Entrada-salida para abarcar en las sesiones de clase este contenido de programación se propusieron resolver los siguientes ejercicios:

- Ejercicio 01: Escribe un programa que pida al usuario el nombre de sus dos mejores amigos y luego los imprima en pantalla.
- Ejercicio 02: Escribe un programa para sumar dos números.
- Ejercicio 03: Escribe un programa para restar dos números.
- Ejercicio 04: Escribe un programa para pedir la medida del lado de un cuadrado y encontrar su perímetro.

- Ejercicio 05: Escribe un programa para pedir la medida la base y la altura de un triángulo y encontrar su área (ver Figura 5.15).
- Ejercicio 06: Escribe un programa para convertir una cantidad ingresada en Km a metros.

```

Pseudocódigo
01  iniciar
02  escribir "Ingresa la base del triángulo"
03  leer base
04  escribir "ingresa la altura del triángulo"
05  leer altura
06  area=base*altura / 2
07  escribir "El área del triángulo es: ", area
08  finalizar

```

Figura 5.15 Ejemplo de resolución del Ejercicio 05 (fuente: propia)

5.4.1.2 Condicionales para abarcar en las sesiones de clase este contenido de programación se propusieron resolver los siguientes ejercicios:

- Ejercicio 07: Escribe un programa para determinar si un número ingresado es positivo o negativo.
- Ejercicio 08: Escribe un programa para hallar el número menor de 2 números.
- Ejercicio 09: Escribe un programa para hallar el número mayor de 2 números.
- Ejercicio 10: Escribe un programa que indique si un número ingresado es o no par (ver Figura 5.16).
- Ejercicio 11: Escribe un programa que pida la edad de una persona e indique si es mayor o menor de edad.

```

Pseudocódigo
01  iniciar
02  escribir "Ingresa un número"
03  leer numero
04  par = numero / 2
05  si par=0 entonces escribir "Par" sino escribir "Impar"
06  finalizar

```

Figura 5.16 Ejemplo de resolución del Ejercicio 10 (fuente: propia)

5.4.1.3 Bucles para abarcar en las sesiones de clase este contenido de programación se propusieron resolver los siguientes ejercicios:

- Ejercicio 12: Escribe un programa con la estructura bucles, que pida al usuario ingresar 5 números y que luego imprima como resultado la suma de los números.
- Ejercicio 13: Escribe un programa con la estructura bucles, para calcular el promedio de N notas.
- Ejercicio 14: Escribe un programa que escriba la tabla de multiplicar de un número ingresado.
- Ejercicio 15: Escribe un programa que calcule el factorial de un número ingresado (ver Figura 5.17).

```
Pseudocódigo
01  iniciar
02  escribir "Ingresa un número"
03  leer numero
04  //para que no sea el resultado cero
05  resultado=1
06  para i=1 hasta numero
07  resultado=resultado * i
08  siguiente i
09  escribir "El factorial es: ", resultado
10  finalizar
```

Figura 5.17 Ejemplo de resolución del Ejercicio 15 (fuente: propia)

5.4.2 Intérprete del compañero de aprendizaje

Bajo el criterio de mantener la navegación del estudiante sin salir de la interfaz del sistema interactivo educativo de Alcody, se integró un intérprete.

Como se observa en la Figura 5.18 el proceso de ejecución del código fuente (AlcodyGEjec), se basa en las fases de análisis léxico, sintáctico y semántico.

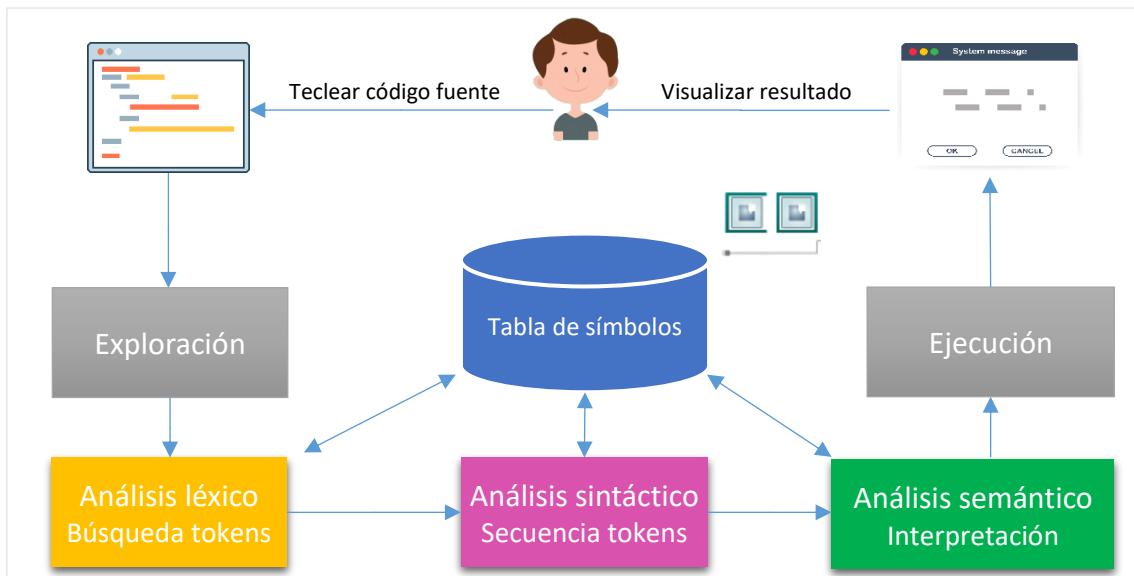


Figura 5.18 Esquema de ejecución del código fuente (AlcodyGEjec) (fuente: propia)

5.4.3 Proceso de ejecución

5.4.3.1 Análisis léxico

Como paso inicial se realiza el proceso de exploración, para pasar al análisis léxico, también conocido como Lexer que es el proceso de convertir una secuencia de caracteres, en una secuencia de tokens. Para esto toma el código fuente y lo compara con la tabla de símbolos (ver Tabla 5.11) y la gramática regular de instrucciones (ver Tabla 5.12).

En la gramática regular se identifican los elementos más pequeños del código, que permiten que un significado dentro de la programación en el lenguaje usado que es el pseudocódigo, para lo cual la gramática usada utiliza el siguiente listado instrucciones:

Ord	Instrucción	Tipo	Tipo
1	iniciar	General	Sentencia
2	finalizar	General	Sentencia
3	leer	Entrada – salida	Sentencia
4	si	Entrada – salida	Sentencia
5	entonces	Estructura condicionales	Sentencia
6	sino	Estructura condicionales	Sentencia
7	para	Estructura bucles	Sentencia
8	hasta	Estructura bucles	Sentencia
9	pasos	Estructura bucles	Sentencia
10	siguiente	Estructura bucles	Sentencia

Tabla 5.12 Listado de instrucciones de la gramática (fuente: propia)

A manera de un vector, el código es recorrido carácter por carácter para detectar posibles instrucciones que pueden ser de tipo: variables, sentencias o funciones, el resultado final de este análisis es una secuencia de tokens.

Esta secuencia de tokens corresponde al mismo orden en que los tokens fueron detectados en el recorrido del código fuente. La Figura 5.19 describe el esquema de funcionamiento del analizador léxico, que recibe el código fuente y da como resultado la secuencia de tokens.

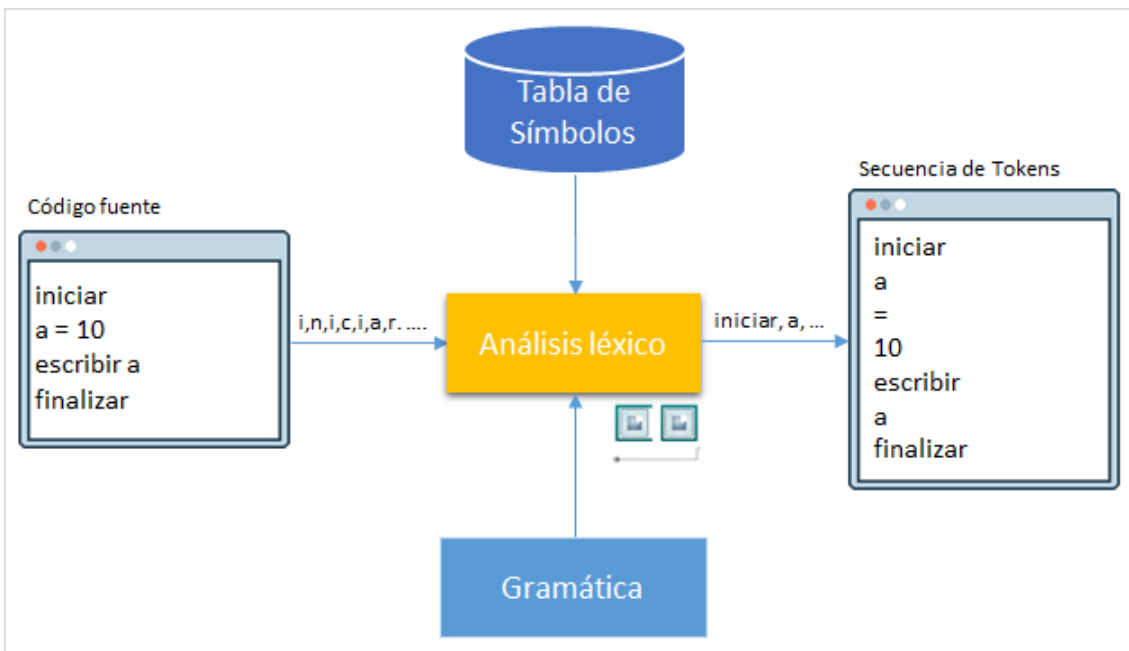


Figura 5.19 Esquema de funcionamiento del Análisis léxico en Alcodey (fuente: propia)

5.4.3.2 Análisis sintáctico

El siguiente paso es el análisis sintáctico que se realiza con ayuda de un componente generalmente llamado “parser” que toma como entrada el flujo de tokens generado por el análisis léxico con la gramática formal y la tabla de símbolos.

La gramática formal son especificaciones sintácticas precisas del pseudocódigo, el parser (análisis sintáctico) especifica el orden en el que debería venir la secuencia de tokens (una a continuación de la otra), para que el programa se considere sintácticamente correcto.

En la Figura 5.20, se observa en el ejemplo un árbol de sintaxis concreta, siendo este resultado del análisis sintáctico, el cual organiza de manera jerárquica la información proveniente del flujo de tokens y la gramática formal reflejando su estructura sintáctica.

Para ejemplificar, se va a utilizar el siguiente algoritmo que ingresa como código fuente al analizador léxico, y este pasa como secuencia de tokens del analizador semántico.

- 1 iniciar
- 2 a = 10
- 3 escribir a
- 4 finalizar

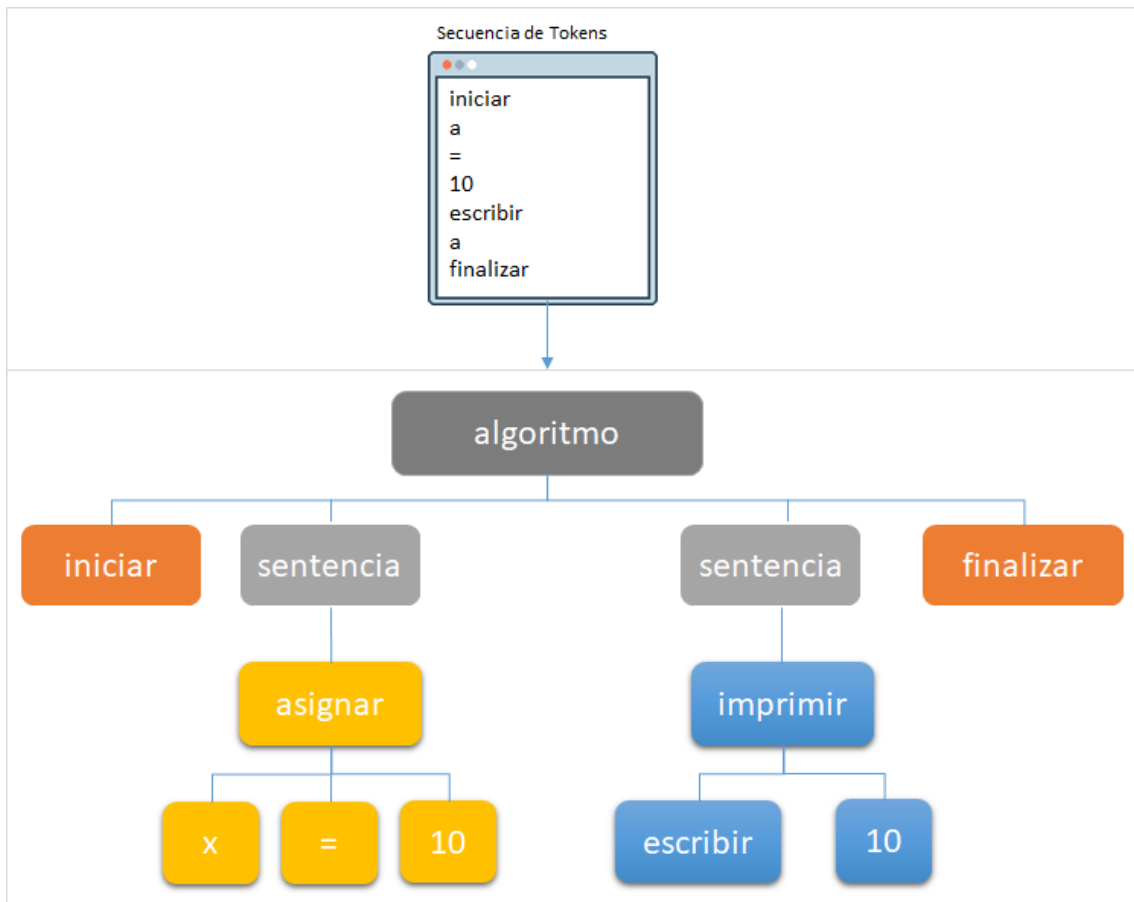


Figura 5.20 Ejemplo de árbol de sintaxis (fuente: propia)

La Figura 5.21 describe el esquema de funcionamiento del analizador sintáctico, recibe toda secuencia de tokens encontrados y da como resultado un árbol de sintaxis concreta.

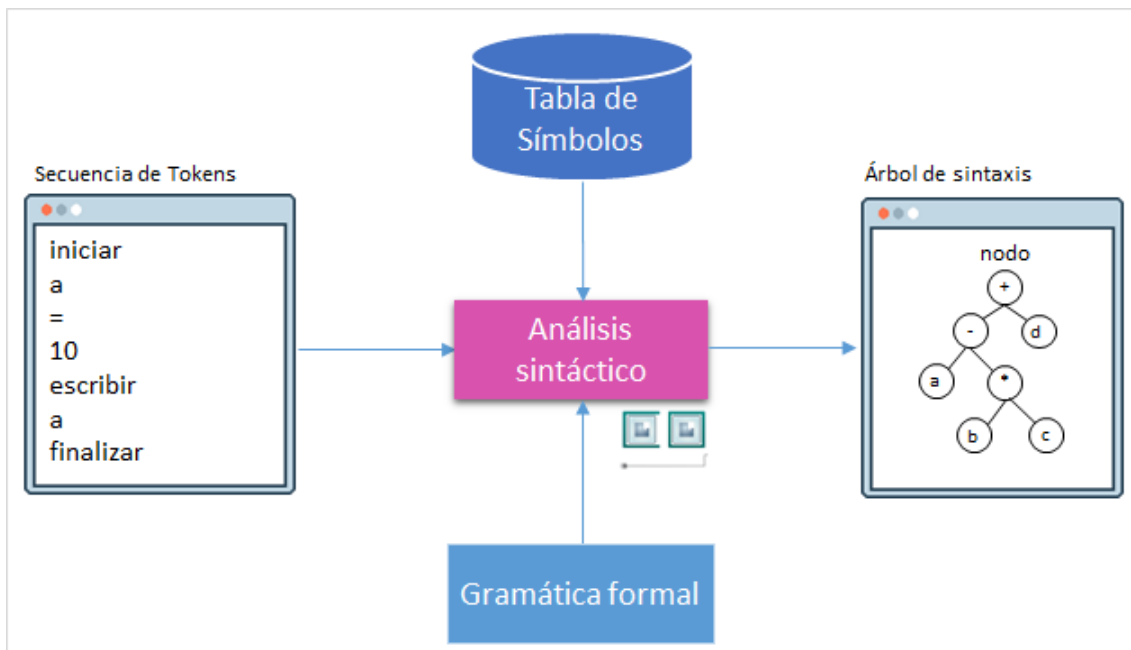


Figura 5.21 Esquema de funcionamiento del Análisis sintáctico en Alcodey (fuente: propia)

5.4.3.3 Análisis semántico

La última fase es el análisis semántico, que determina el significado real del programa siendo la principal función del intérprete la de traducir la información del pseudocódigo en las correspondientes sentencias que van a ser ejecutadas (ver Tabla 5.13).

Ord	Identificador inicial de Tokens	Asignación a sentencias ejecutables
1	this.functions["ESCRIBIR"]	this.print_statement;
2	this.functions["INICIAR"]	this.end_statement;
3	this.functions["LEER"]	this.input_statement;
4	this.functions["SI"]	this.if_statement;
5	this.functions["FIN"]	this.end_statement;
6	this.functions["PARA"]	this.for_statement;
7	this.functions["SIGUIENTE"]	this.next_statement;
8	this.functions["ALEATORIO"]	this.randomize_statement;

Tabla 5.13 Listado de sentencias ejecutables del intérprete (fuente: propia)

El análisis semántico toma como entrada el árbol de sintaxis concreta, generado del análisis sintáctico y otra parte la gramática de atributos para dar información adicional sobre datos que se asignan a cada nodo.

La Figura 5.22 describe el esquema de funcionamiento del analizador semántico, que recibe el árbol de sintaxis concreta y da como resultado un código que permite al intérprete dar paso a la ejecución.

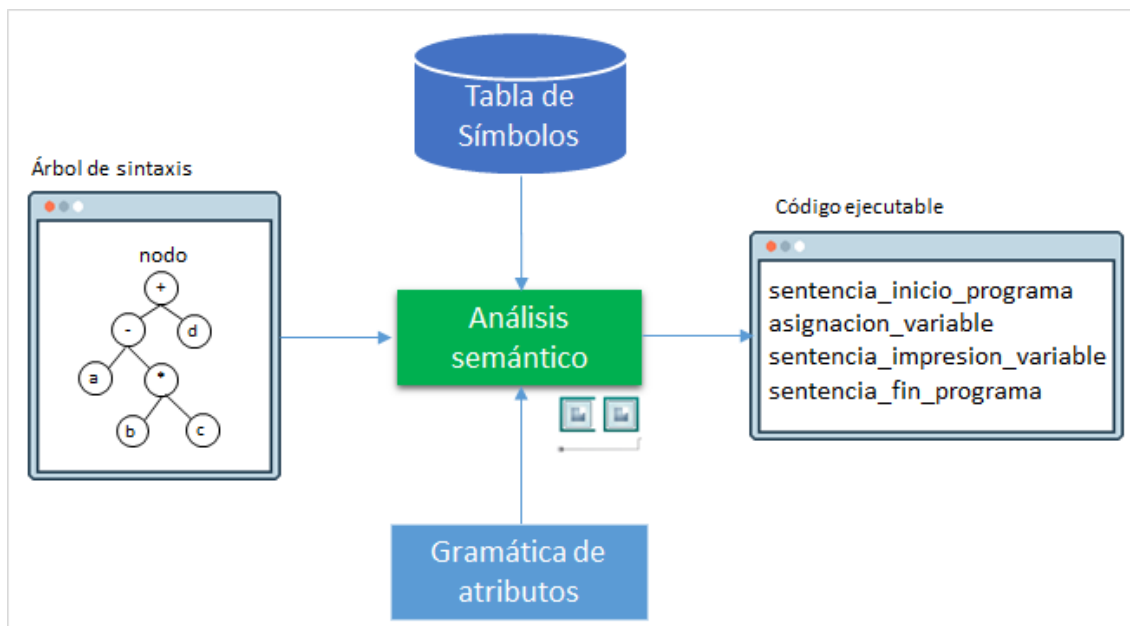


Figura 5.22 Esquema de funcionamiento del Análisis semántico en Alcody (fuente: propia)

5.4.4 Proceso de depuración

Lee et al., (2011) plantean la progresión del aprendizaje (Usar-Modificar-Crear) en la enseñanza de programación en niveles pre-universitarios, de igual forma en Alcody los niños son capaces de escribir y modificar su código.

Se consideró relevante brindar la posibilidad de tener un depurador de código, permitiendo a los estudiantes, detectar e identificar las líneas de su código fuente que contienen algún error, para que los estudiantes puedan corregirlo.

El desarrollo del proceso de ejecución y depuración se realizan en paralelo, es por ello que en cada una de las etapas del proceso de análisis (léxico, sintáctico y semántico) proporcionan información para poder detectar errores en el código.

Para ayudar a los estudiantes a identificar las instrucciones, se asoció las instrucciones a un color. Así al terminar de escribir la palabra que identifica a la instrucción esta toma el color texto correspondiente (ver Tabla 5.14).

Ord	Instrucción	Color	
1	Iniciar	#D6234A	
2	Finalizar	#D6234A	
3	Leer	#008000	
4	Escribir	#0000FF	
5	Si	#FFA500	
6	Entonces	#FFA500	
7	Sino	#FFA500	
8	Para	#87CEEB	
9	Hasta	#87CEEB	
10	Siguiente	#87CEEB	

Tabla 5.14 Listado colores para instrucciones de pseudocódigo (fuente: propia)

Se crearon logs que permiten identificar las palabras donde los estudiantes incurren en errores comunes al escribir las instrucciones o en su defecto si usan otras palabras con la intención de invocar a una instrucción determinada.

Esto permitió que a modo del proceso de exploración que se realiza de carácter en carácter, de manera adicional se de una ayuda predictiva a los estudiantes identificando la intención y generando un mensaje de ayuda (ver Tabla 5.15).

De la misma manera se identificó que en estudiantes de esta edad, es frecuente el encontrar errores ortográficos o de omisión de caracteres, por lo que se debe gestionar esto en la consulta a la base de datos una vez identificada la aparición de un token.

Ord	Instrucción	Usos frecuentes	Mensaje
1	Iniciar	inicio, comienzo, empezar	Línea XX, debes teclear iniciar
2	Finalizar	terminar, fin, culminar	Línea XX, debes teclear finalizar
3	Leer	ler, pedir, tomar	Línea XX, debes teclear leer
4	escribir	presentar, imprimir	Línea XX, debes teclear escribir

Tabla 5.15 Ejemplos de escritura errónea de instrucciones (fuente: propia)

De igual manera la detección de errores de la gramática formal, se detectó en la omisión de partes de las estructuras de código, como por ejemplo no colocar **entonces** en la estructura condicional, o **hasta** en el bucle para, ejemplos se describen en la Tabla 5.16.

Ord	Instrucción	Omisión	Ejemplo: Mensaje
1	Si	entonces	Línea XX, se espera sentencia ENTONCES
2	Si	condición	Línea XX, se espera CONDICIÓN
2	Si	sino	Línea XX, se espera sentencia SINO
3	Para	hasta	Línea XX, se espera sentencia HASTA
4	Para	Siguiente	Línea XX, se espera sentencia SIGUIENTE

Tabla 5.16 Ejemplos de omisiones a la gramática formal (fuente: propia)

La Figura 5.23 describe el esquema de funcionamiento del depurador (AlcodyGProg), con el que cuenta el sistema interactivo educativo de Alcody, que muestra mensajes de ayuda al estudiante tanto al momento de la escritura del código como previo a la ejecución del código.

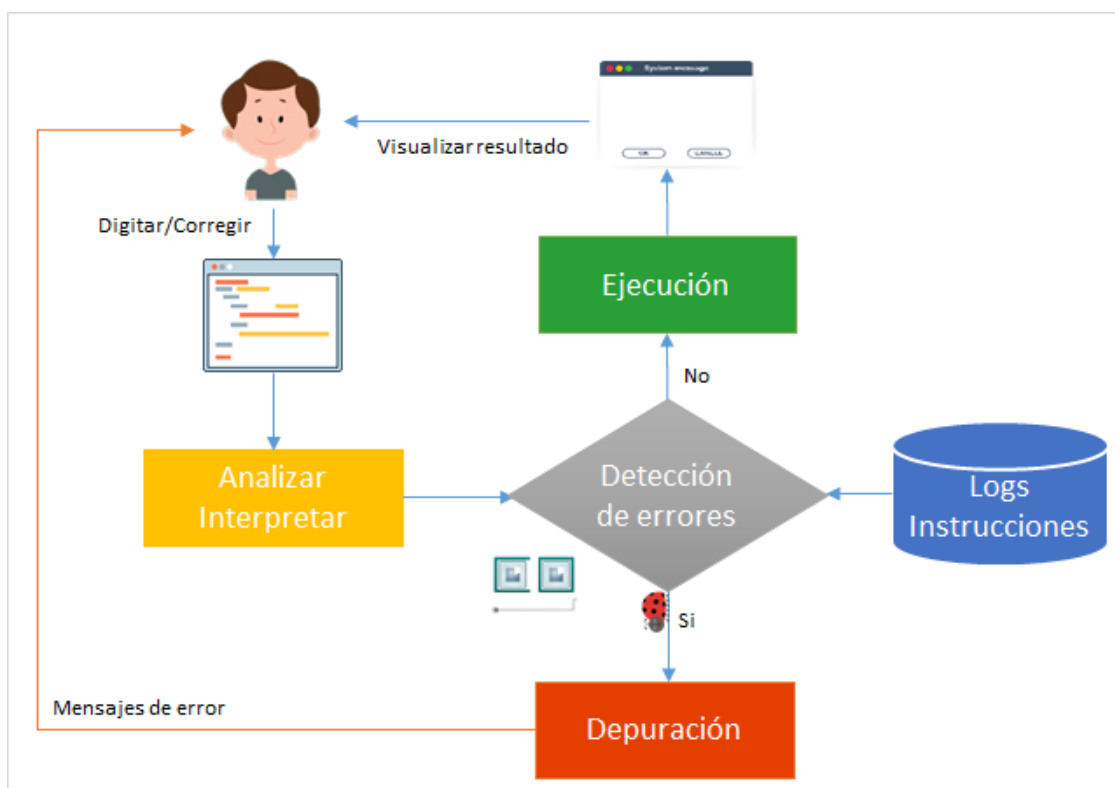


Figura 5.23 Esquema de ejecución y depuración (AlcodyGProg) (fuente: propia)

5.5 Validación de la interfaz y de la funcionalidad

5.5.1 Experimento 1

Artículo: Pueden los niños aprender a programar usando un entorno de programación basado en texto con un compañero de aprendizaje (Morales et al., 2020)

En este estudio se investiga si los niños, sin conocimiento previo de programación, podrán aprender a programar usando un entorno de programación basado en texto con un agente compañero como Alcody.

La hipótesis es que los niños mejorarán sus notas de forma significativa usando un entorno de aprendizaje de la programación basado en texto con el agente compañero.

El presente experimento se ha trabajado con 21 estudiantes de entre 10 y 12 años participaron en el experimento (10 niños y 11 niñas). Todos estaban matriculados en 6º de Educación Primaria.

No habían tenido clases de programación con anterioridad, pero sí iban de forma regular a la clase de informática (con conexión a Internet) para usar los ordenadores en otras asignaturas como inglés. En el aula de informática había un ordenador por niño.

El experimento se llevó a cabo en un colegio público de Madrid en España. La razón por la que se escogió este colegio es porque al no ser obligatoria la enseñanza de la programación en Educación Primaria en la Comunidad de Madrid en España se dependía de la voluntad del Director y de los docentes para usar tiempo de otras asignaturas.

En el caso de este colegio, la Directora aceptó el uso de Alcody puesto que creía que podía ser beneficioso para los estudiantes y habló con el docente de Plástica que aceptó ceder parte de su horario.

Cada niño fue asignado con un usuario y contraseña para poder acceder a Alcody. Con sus credenciales individuales podían acceder a Alcody en su ordenador. Como currículum, Alcody tiene tres tutoriales que abarcan los conceptos tradicionales de un curso de introducción a la programación: entrada/salida, condicionales y bucles.

El primer tutorial comienza con la explicación del pseudocódigo y la navegación en Alcody, el concepto de programa, programación, variable y entrada/salida y en los siguientes tutoriales se revisan el resto de conceptos y cómo programarlos en Alcody.

Los tutoriales están integrados dentro de Alcody con eso los niños podían acceder en cualquier momento en el que les surge una duda.

Se siguió un procedimiento pre-post test con un solo grupo experimental. Para poder identificar si los estudiantes tenían algún tipo de conocimiento previo de programación,

el primer día de clase, a principios de febrero, se les pidió a todos que completasen de forma individual y con su código de alumno un pre-test con las siguientes cinco preguntas de programación:

- P1 (*entrada/salida*). Escribe un programa que muestre “Hola” en la pantalla.
- P2 (*entrada/salida + variables*). Escribe un programa que muestre el resultado de sumar 2 más 2.
- P3. (*entrada/salida + variables + condicionales*) Escribe un programa que compare dos números dados e indique cuál es el mayor.
- P4. (*entrada/salida + variables + condicionales*) Escribe un programa que pregunte una palabra en español y muestre su traducción en inglés. En la pantalla deben aparecer ambas palabras.
- P5. (*entrada/salida + variables + condicionales + bucles*) Escribe un programa que cuente del 1 al 5. En el pre-test se indicó a los estudiantes que podían usar cualquier pseudocódigo o lenguaje de programación para resolver estos ejercicios de programación. El tiempo para realizar el test fue de una hora.

La Tabla 5.17 recoge la rúbrica de evaluación, cada pregunta fue evaluada de 0 (nota mínima) a 1 (nota máxima). La nota mínima para el test fue 0 y la máxima 5, siendo necesario como mínimo 2,5 puntos para poder considerar que se había aprobado el test.

Después del pre-test, los estudiantes pudieron usar Alcodey durante cuatro meses desde cualquier ordenador conectado a Internet con sus credenciales. Además, una vez al mes se hace un repaso presencial en su clase con un docente experto en programación y en Alcodey.

En el aula también estaba el docente de Plástica de los estudiantes para vigilar su comportamiento respecto al uso de los ordenadores y en general, en el aula. El mismo pre-test se usó como post-test como última actividad que realizaron los estudiantes, en las mismas circunstancias que el pre-test y durante el mismo periodo de tiempo.

Puntos	Evaluación
0	Respuesta en blanco o equivocada
0,25	Se usa una instrucción correcta
0,5	Todas las instrucciones usadas son correctas, pero sin parámetros
0,75	Todas las instrucciones usadas son correctas, pero algunas no son necesarias
1	Código perfecto

Tabla 5.17 Rúbrica para el test de programación (Morales et al., 2020)

	\bar{x}	SD
Pre	1,69	1,07
Post	2,42	1,45

Tabla 5.18 Parámetros estadísticos del experimento (Morales et al., 2020)

5.5.2.1 Resultados

La Tabla 5.18 recoge los parámetros estadísticos con la media y la desviación típica del pre-test y el post-test realizado a los 21 estudiantes que usaron Alcody durante los 4 meses que duró el experimento.

Como se puede observar hay una mejora en las notas de los estudiantes después de usar Alcody, como se desprende del uso de una t de Student para muestras apareadas con un valor p-valor significativo ($t=2,509$, $p < 0,021$).

Inicialmente, debido a su falta de conocimientos de programación la media es realmente baja en la escala de 0 (mínima puntuación) a 5 (máxima puntuación), no llegan a aprobar, y después de usar Alcody la media sube de forma significativa hasta 2,42 casi rozando el aprobado que se estableció en 2,5.

Desde el punto de vista cualitativo, cuando se preguntó a los estudiantes respecto a su grado de satisfacción con el uso de Alcody en entrevistas realizadas a los estudiantes según iban usando el sistema, las opiniones recibidas fueron positivas con frases como:

- “Me encanta Alcody. No sabía qué era eso de programar y sonaba muy difícil, y gracias a Alcody me parece interesante y quiero aprender más”
- “Recordar las instrucciones y encontrar el orden correcto para que funcione el programa sería muy complicado sin la ayuda de Alcody. Gracias a Alcody me doy cuenta de dónde me he equivocado y me alegra mucho ver cuando funciona el programa.”

Finalmente, cuando se preguntó al docente de Plástica su opinión respecto a lo que estaba viendo en clase con los estudiantes usando Alcody, su comentario fue que le llamaba la atención el grado de interés y la involucración tan alta que tenían los estudiantes con el sistema, intentando siempre conseguir que los programas funcionaran y respondiendo las preguntas de Alcody.

5.5.2 Experimento 2

Artículo: Can learning companion be used to continue teaching programming to children even during the Covid-19 Pandemic? (Ocaña et al., 2020)

El objetivo principal del estudio fue averiguar si los estudiantes con conocimientos previos de Scratch seguirían aprendiendo programación incluso durante la pandemia de COVID-19 con la ayuda del compañero de aprendizaje de Alcodey.

Las puntuaciones de aprendizaje de los estudiantes sirvieron como medida cuantitativa para comprobar si su aprendizaje con Alcodey había sido efectivo antes y durante el COVID-19. La pregunta de investigación 1 (RQ1) fue la siguiente:

RQ1. ¿Los estudiantes (de entre 10 y 12 años) aumentaron sus puntuaciones de aprendizaje en un pre-test y post-test con Alcodey antes y durante la pandemia de COVID-19?

Este enfoque cuantitativo se combinó con un enfoque cualitativo teniendo en cuenta la motivación, satisfacción y opinión general de los estudiantes al utilizar Alcodey antes y durante el COVID-19. La pregunta de investigación descriptiva 2 (RQ2) fue la siguiente:

RQ2 - ¿Qué tan satisfechos estaban los estudiantes al interactuar con Alcodey antes y durante la pandemia de COVID-19? ¿Qué opinión tenían de Alcodey? ¿Estaban motivados para usar Alcodey incluso durante el confinamiento?

La experiencia se realizó en el año académico 2019/2020 con 137 alumnos (de entre 10 y 12 años) de una escuela de Educación Primaria en Ecuador. Casi la mitad de los estudiantes eran niñas.

La escuela fue elegida por dos razones principales: (i) el director de la escuela estaba dispuesto a autorizar el experimento siempre que todos los niños usaran el compañero de aprendizaje para aprender programación, y (ii) aunque no es obligatorio en Ecuador, los niños de esta escuela aprenden programación usando Scratch a partir de cuarto año de Educación Primaria (8-9 años).

Durante el período anterior al COVID-19, los alumnos se encontraban en clase (cuatro grupos de 32, 36, 34 y 35 alumnos) con su profesor y dos investigadores que se encargaban de realizar el experimento. Los investigadores fueron los observadores de la experiencia de aprendizaje para el enfoque cualitativo.

El docente solo debía brindar apoyo si los estudiantes tenían algún problema. Cada niño tenía un ordenador para usar en la clase.

Durante el encierro, los estudiantes estaban en su casa. Se conectaron en línea, en su grupo, y con su docente desde su ordenador personal. Todos los niños tenían un ordenador, una habilidad digital similar para usarla y una conexión a Internet en casa.

La escuela no otorgó permiso para grabar las sesiones. El consentimiento fue para permitir que los niños usaran Alcody y los observadores tomaran notas. La confidencialidad estaba asegurada, ya que los niños no dieron su nombre a Alcody en ningún momento. Sin embargo, se les asignó un código.

Se les dijo que era muy importante que recordaran su código ya que Alcody lo pedía como su nombre de usuario para todas las sesiones. Para garantizar respuestas honestas a los cuestionarios, no se solicitó ese código. A los niños se les aseguró que el maestro no penalizaría a nadie por sus respuestas.

Todos los materiales educativos se proporcionaron en el entorno educativo de Alcody como se describe en la Subsección 5.6.3.5. Los niños no recibieron material adicional en clase antes de COVID-19, ni durante el encierro.

El plan de estudios de Alcody cubre conocimientos sobre secuenciación, memoria, variables, entradas, salidas y conceptos de condicionales. El objetivo es enseñar a los estudiantes su significado en la programación y desarrollar sus habilidades para escribir programas usando esos conceptos y sus instrucciones en pseudocódigo.

Alcody enseña conceptos de programación de forma interactiva. Alcody pedirá en primer lugar a los estudiantes que escriban un programa para decir "Hola" en la pantalla. Al no centrarse en la teoría sobre lo que es un programa, los conceptos se cubren de acuerdo con la idea central del construccionismo de "aprender haciendo".

Cuando el estudiante escribe un programa que muestra "Hola" en la pantalla, demuestra que comprende no solo el concepto del programa, sino también que es capaz de hacer un programa de entrada / salida. Los niños deben, por lo tanto, comprender qué es un programa para crear programas sintácticamente correctos.

Las actividades se han realizado una hora a la semana con Alcody. Antes del confinamiento, se les pidió que trabajaran con Alcody durante esa hora en su grupo. Durante el encierro, la clase fue a la misma hora y por la misma duración que antes y con el mismo maestro conectado en línea mediante un sistema de videoconferencia.

Los niños tenían la posibilidad de leer un tutorial si no entendían la explicación proporcionada por Alcody, como si estuvieran en el aula.

Durante los dos primeros meses, los estudiantes utilizaron las actividades o recetas durante media hora y leyeron las tutorías de Alcody durante el resto de la clase. Durante los meses restantes, los estudiantes programaron con Alcody sobre los conceptos en estudio durante cincuenta minutos.

Como recompensa, y para ofrecer un refuerzo, durante los últimos diez minutos, los estudiantes tuvieron la opción de completar algunas actividades o seguir algunas recetas. En la Tabla 5.19 se muestra el tiempo con las actividades docentes solicitadas a los niños.

Periodo	Conceptos cubiertos con Alcodey
Septiembre-octubre 2019	Pre-test, secuencias con actividades. Tutoriales sobre el pseudocódigo de Alcodey
Noviembre-diciembre 2019	Secuencias, memoria, variables con recetas (actividades para reforzar)
Enero-febrero 2020	Secuencias, memoria, variables y entradas / salidas con chat (actividades y recetas para reforzar)
Marzo-mayo 2020	Secuencias, memoria, variables, entradas / salidas y condiciones con chat (actividades y recetas para reforzar)

Tabla 5.19 Tiempo de enseñanza con Alcodey (Ocaña et al., 2020)

Las variables dependientes del experimento de un solo grupo pre-post test están todas relacionadas con el aprendizaje de conceptos de programación, medidos por las puntuaciones obtenidas por los estudiantes en tests en diferentes meses a lo largo del curso académico.

Estas puntuaciones (NotaPre, NotaPost1, NotaPost2, Notapre_cond, Notapost_cond) se recopilaron con las pruebas descritas.

Solo hubo una variable independiente en el experimento, definida de manera general como exposición al uso de Alcodey. Esta variable tiene diferentes niveles en función del uso del compañero de aprendizaje hasta un mes determinado: es decir, septiembre, febrero y junio. La Tabla 5.20 contiene la lista de variables dependientes e independientes.

Aspecto	Tipo	Variable	Nombre
Aprendizaje de conceptos de programación	DV	Puntuaciones de la prueba realizada en septiembre	NotaPre
	DV	Puntuaciones de la prueba realizada en febrero	NotaPost1
	DV	Puntuaciones de la prueba realizada en junio	NotaPost2
	DV	Puntuaciones de preguntas condicionales en la prueba realizada en septiembre	Notapre_cond
	DV	Puntuaciones de preguntas condicionales en la prueba realizada en junio	Notapost_cond
Compañero de aprendizaje	IV	Uso de Alcodey	α

Tabla 5.20 Resumen de variables, tipo (dv- variable dependiente, iv - variable independiente), nombres y descripción (Ocaña et al., 2020)

A continuación, primero, se presenta una de las principales hipótesis de investigación para luego describir cada una de las hipótesis estadísticas de trabajo utilizadas para responderlas.

Ha: Los estudiantes (de entre 10 y 12 años) con conocimientos de Scratch podrán mejorar significativamente su aprendizaje utilizando un compañero de aprendizaje para continuar aprendiendo a programar incluso durante la pandemia de COVID-19.

Esta mejora en su aprendizaje se mide en términos de las puntuaciones de las pruebas descritas. Entonces, en este caso, dos aspectos diferentes están involucrados en esta hipótesis de investigación:

- En primer lugar, como hipótesis general, el interés se centra en las posibles diferencias que puedan existir entre las puntuaciones de las tres evaluaciones realizadas en septiembre, febrero y junio. Las hipótesis nulas y alternativas son las siguientes:

$$H_{0af}: \mu_i = \mu_j = \mu \quad \forall i \neq j \quad H_{1af}: \exists \mu_i \neq \mu_j$$

Es decir, la hipótesis nula H_{0af} es que no hay diferencia entre la media de las puntuaciones de la prueba, siendo μ_i la media de la puntuación en la evaluación en el momento $i = \{\text{septiembre, febrero, junio}\}$. La hipótesis alternativa H_{1af} es que existen algunas diferencias entre las medias en los resultados de la prueba. Esta hipótesis se resuelve mediante un análisis de Anova.

En segundo lugar, si se encuentra alguna diferencia entre la media de las puntuaciones, se lleva a cabo una segunda parte, y es saber exactamente entre qué pares de medias en las puntuaciones se encuentran esas diferencias:

$$H_{0as}: \mu_i = \mu_j ; H_{1as}: \mu_i \neq \mu_j \quad \forall i \neq j$$

Se trata, en total, de 3 contrastes de hipótesis diferentes a modo de estudio post-hoc (en estos estudios, de forma general, el importe es $\binom{n}{2}$, siendo $n =$ número de niveles del factor). La hipótesis nula H_{0as} es que no existe diferencia entre la media de las puntuaciones de la prueba realizada en el mes i y en el mes j .

La hipótesis alternativa H_{1as} es que existe. Estas hipótesis se resuelven con una prueba de Tukey HSD.

En el caso de los conceptos condicionales, se realizaron dos pruebas, en septiembre y junio. Entonces, las hipótesis nulas y alternativas son las siguientes:

$$H_{0ac}: \mu_s = \mu_j ; H_{1ac}: \mu_s \neq \mu_j \quad \forall s \neq j$$

Es decir, la hipótesis nula H_{0ac} es que no hay diferencia entre la media de las puntuaciones de la prueba en septiembre y junio (a saber, μ_s y μ_j). La hipótesis alternativa H_{1ac} es que existen algunas diferencias entre las medias de los resultados de estas pruebas. Esta

hipótesis se resuelve mediante un análisis de la prueba t de Student para muestras pareadas.

Es importante señalar que las diferencias en la hipótesis de contraste se encuentran en un nivel de confianza del 95%.

Se siguió un procedimiento de pre-test y post-test en un periodo de diez meses. Una condición de la escuela era que todos los estudiantes debían usar Alcodey, por lo que los estudiantes no se pudieron separar en grupo control / experimental.

El progreso del conocimiento de programación de los estudiantes se midió pidiendo a todos los estudiantes que completaran una pre-test a principios de septiembre de 2019 antes de usar Alcodey, un post-test posterior justo antes del cierre en marzo de 2020 y un segundo post-test en junio de 2020 durante el confinamiento.

Todas las pruebas utilizan las siguientes preguntas enfocadas a evaluar las habilidades de programación de los estudiantes utilizando los conceptos cubiertos en Alcodey:

Q1. Escriba un programa para mostrar "Hola" en la pantalla (secuencia, conceptos de salida).

Q2. Escribe un programa para sumar dos números (conceptos de secuencia, memoria, variable, entrada, salida).

Q3. Escriba un programa que calcule el perímetro de un cuadrado a partir del valor ingresado por el usuario de un lado (conceptos de secuencia, memoria, variable, entrada, salida). El pre-test (septiembre de 2019) y el post-test (junio de 2020) también incluyeron una pregunta centrada en evaluar el uso de condiciones (se excluyó del post-test de febrero de 2020 porque aún no se había enseñado el concepto de condicionales).

Q4. Escribe un programa que muestre el número más alto de dos números dados (secuencia, memoria, variable, entrada, salida, conceptos condicionales).

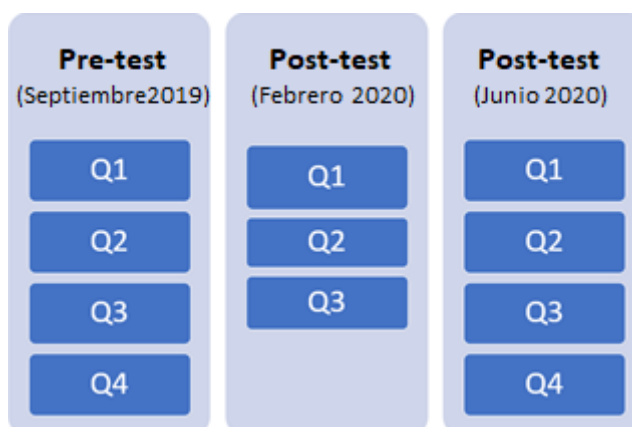


Figura 5.24 Preguntas analizadas en cada prueba (Ocaña et al., 2020)

La Figura 5.24 muestra la distribución de preguntas en cada test. Cada pregunta se evaluó de 0 (puntuación mínima) a 1 (puntuación máxima) siguiendo la rúbrica que se muestra en la Tabla 5.21.

La puntuación mínima para la prueba fue 0 (todas las respuestas incorrectas) y la máxima fue 10 (todas las respuestas fueron correctas).

La puntuación de la prueba fue el resultado de la media ponderada reflejada en la Ecuación 1, ya que todas las preguntas se ponderaron igual en la puntuación de la prueba. Una nota de aprobado fue 5.

$$Note = \frac{10}{n} \sum_{i=1}^n s_i$$

Donde n es el número de preguntas de la prueba y S_i es la puntuación de la pregunta i de la prueba.

Puntos	Evaluación
0	Sin respuesta o incorrecta
0,25	Las instrucciones de inicio y / o finalización se utilizan correctamente pero el resto es incorrecto
0,5	Las instrucciones de inicio y finalización se utilizan correctamente, así como la secuencia general de instrucciones, aunque el uso de variables o sintaxis es incorrecto.
0,75	El inicio, final y secuencia de instrucciones son correctos así como el uso de variables con posibles errores menores en la sintaxis
1	El inicio, el final y la secuencia de instrucciones son correctos, así como el uso de variables y sintaxis.

Tabla 5.21 Rúbrica para la prueba (Ocaña et al., 2020)

Después de la prueba de febrero, se pidió a todos los estudiantes que completaran un cuestionario de opción múltiple de diez ítems.

La Tabla 5.22 muestra el cuestionario creado para esta evaluación como un cuestionario corto validado no previo (máximo de 10 preguntas) que debía ser suficiente para cubrir información sobre opinión, satisfacción y motivación.

Durante el tiempo que el docente pudiera estar en clase con los alumnos, de septiembre de 2019 a marzo de 2020, se podría emplear la observación directa y registrar los programas codificados por los alumnos para conocer cómo estaban programando, reconocer posibles errores comunes y cómo los estudiantes podían superar esos errores.

De marzo de 2020 a mayo de 2020, no fue posible tener observación directa, dada la pandemia de COVID-19.

Como los estudiantes habían podido continuar usando Alcodey, se creó un cuestionario anónimo en línea más corto (cinco ítems de opción múltiple) para conocer su opinión, satisfacción y motivación como se muestra en la Tabla 5.23, los registros de su actividad y los programas creados fueron analizados.

La Tabla 5.24 resume las técnicas de recolección de datos utilizadas para esta investigación y la pregunta de investigación a ser respondida con los datos analizados.

Ítem	Pregunta	Factor evaluado
1	¿Qué tarea has elegido más a menudo con Alcodey?	Opinión
2	¿Cuál es tu tarea favorita con Alcodey?	Opinión
3	¿Cuál es su actividad favorita?	Opinión
4	¿Cuál es tu receta favorita?	Opinión
5	¿Te gusta la ayuda que te ofrece Alcodey?	Satisfacción
6	¿Estás satisfecho con obtener recompensas?	Satisfacción
7	¿Qué tan satisfecho está con su interacción con Alcodey?	Satisfacción
8	¿Te gusta aprender a programar con Alcodey?	Satisfacción
9	¿Crees que Alcodey te motiva a programar?	Motivación
10	¿Crees que podrás programar con la ayuda de Alcodey?	Opinión

Tabla 5.22 Cuestionario de pandemia pre-covid-19 (Ocaña et al., 2020)

Ítem	Pregunta	Factor evaluado
1	¿Crees que puedes concentrarte en casa usando Alcodey?	Opinión
2	¿Estás motivado para seguir usando Alcodey?	Motivation
3	¿Te gusta la ayuda que te ofrece Alcodey?	Satisfaction
4	¿Te gusta que Alcodey te ayude a afrontar la pandemia de COVID-19?	Satisfaction
5	¿Qué recomendaciones quieres que te dé Alcodey para ayudarte a seguir aprendiendo a programar con él?	Opinión

Tabla 5.23 Cuestionario durante la pandemia covid-19 (Ocaña et al., 2020)

Periodo	Técnica	RQ
Septiembre 2019 – Marzo 2020	Pre-test & post-test 1	1
	Cuestionario pre-covid	2
	Observación directa	2
	Programas de estudiantes	2
Marzo 2020 – Mayo 2020	Pre-covid cuestionario	1
	Observación directa	2
	Programas de estudiantes	2
		2

Tabla 5.24 Técnicas de recopilación de datos (Ocaña et al., 2020)

Para responder RQ1 y prueba Ha, se utilizó un ANOVA de una vía para analizar la diferencia entre las medias de la puntuación en las pruebas en los tres meses distintos en los que se realizó, esto es, septiembre, febrero y junio:

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

Donde y_{ij} son observaciones para la variable dependiente (siendo $y = \{\text{NotaPre}, \text{NotaPost1}, \text{NotaPost2}\}$), calculadas como las puntuaciones de la prueba; μ corresponde a la media del modelo; α_i es el efecto principal del nivel i (la variable independiente introducida, donde cada nivel depende del uso del compañero de aprendizaje hasta ese momento i) y ε_{ij} es el error aleatorio (donde $i = \text{septiembre, febrero, junio}$, y $j = \text{índice sobre alumno}$).

En el caso del concepto condicionales, las hipótesis se resolvieron con una prueba t para muestras pareadas. Para responder a la RQ2, se analizaron las respuestas a los cuestionarios junto con el análisis de los programas, la observación directa y los registros de registro.

Los cálculos estadísticos se realizaron con el programa IBM SPSS Statistics Versión 25. Para probar la confiabilidad entre evaluadores durante la codificación de las asignaciones usando la rúbrica, se usa el valor kappa de Cohen. Esta estadística es una de las más utilizadas para medir la concordancia entre evaluadores. Su valor puede oscilar entre -1 y 1. En los cuestionarios examinados, todos los elementos tienen valores kappa de Cohen superiores a 0,80.

La validez del contenido del cuestionario, en este estudio se determinó mediante el análisis cuantitativo de nueve juicios de expertos, dando un valor V de Aiken superior a 0,7 en todos los ítems, resultado considerado como aceptable.

La validez del constructo se llevó a cabo mediante análisis factorial exploratorio, utilizando el método de rotación varimax y criterio de extracción de autovalores mayor que procedimiento de pre-test y post-test 1. Además, se calculó el alfa de Cronbach, siendo su valor aceptable: 0,812.

Para fortalecer la validez y confiabilidad de la observación directa, dos observadores independientes tomaron notas durante las sesiones presenciales de enseñanza con Alcodey (alrededor de 30 niños por grupo, varias sesiones por semana en el horario escolar).

A los observadores se les había entregado un cuaderno para que anotaran qué tan comprometidos y motivados percibían a los niños. Se calculó kappa de Cohen, siendo su valor bueno: 0,625.

5.5.3.1 Ganancias de aprendizaje

Se estudió cualquier posible mejora en las puntuaciones de cada individuo entre el pre-test y el post-test antes de la pandemia de COVID-19 y el segundo post-test durante la pandemia de COVID-19.

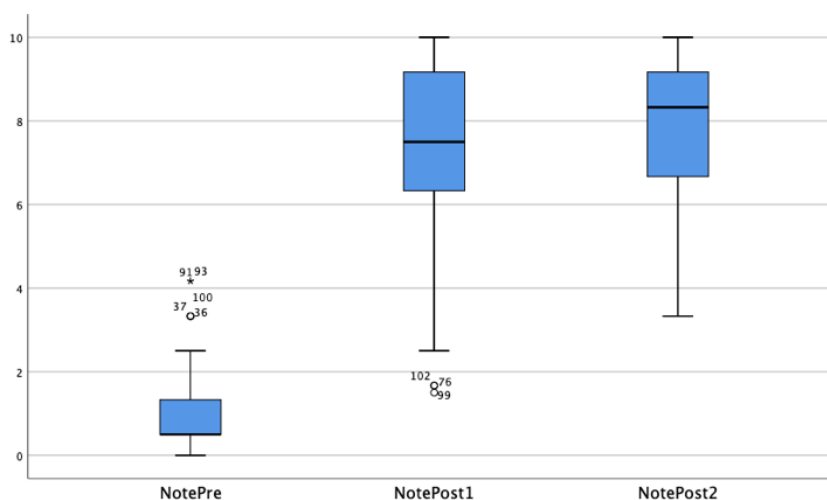


Figura 5.25 Pre-test y post-test diagramas de caja para todos los conceptos (Ocaña et al., 2020)

TEST	N	\bar{x}	M	SD
Pre-test	137	0,88	0,50	0,85
Primer post-test (antes de la pandemia)	137	7,56	7,50	2,05
Segundo post-test (durante de la pandemia)	137	8,01	8,33	1,47

Tabla 5.25 Análisis descriptivo simple (Ocaña et al., 2020)

Dado el número diferente de preguntas en las pruebas, se realizaron dos análisis. El primero compara las puntuaciones de entrada y salida con las respuestas de Q1, Q2, Q3.

El segundo compara las puntuaciones del concepto condicionales con las respuestas de la Q4.

La Figura 5.25 muestra diagramas de caja en los pre-test y post-test para las preguntas de entrada / salida (Q1, Q2, Q3).

Los círculos pequeños con un número representan datos atípicos. La Tabla 5.25 muestra valores medios y medianos (más representativos que la media en distribuciones asimétricas, como en este caso) y desviaciones estándar.

El valor de la M antes de la prueba fue de 0,50, con una dispersión de datos de 0,85. El primer post-test mostró una mayor dispersión de datos (2,05) pero, en general, muestra resultados mucho mejores, con la M en 7,50.

En la segunda prueba posterior, la M aumenta a 8,33 pero con una dispersión de datos de 1,47, las puntuaciones están más concentradas. Los resultados del ANOVA se muestran en la Tabla 5.26.

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	4368,43	2	2184.22	917,01	0,000
En los grupos	971,81	408	2,39		
Total	5340,24	410			

Tabla 5.26 Anova unidireccional (Ocaña et al., 2020)

Como se evidenció, hubo diferencias significativas ($p = 0,000$) en las puntuaciones entre los diferentes niveles para los conceptos entrada / salida. Se utilizó un análisis post-hoc para obtener más información sobre estas diferencias.

Las pruebas HSD de Tukey comparan cada puntuación de la prueba en un momento determinado (septiembre, febrero, junio) con cada otra puntuación dos veces. Estas muestran diferencias significativas ($p < 0,05$) entre todos los pares.

Es decir, las diferencias encontradas entre las puntuaciones de las pruebas en los diferentes meses son todas estadísticamente significativas.

Para el segundo análisis, para comparar las puntuaciones para el concepto de condicionales, se lleva a cabo una prueba t de Student para muestras pareadas entre las puntuaciones previas a la prueba (septiembre de 2019) y las segundas puntuaciones posteriores a la prueba (junio de 2020).

Para ver si hubo un resultado significativo, mejora también en estas puntuaciones, teniendo en cuenta que las condiciones son un concepto que solo se ha explicado en línea durante el bloqueo.

La Figura 5.26 muestra diagramas de caja de las puntuaciones.

La Tabla 5.27 muestra un análisis descriptivo de estos datos. La Tabla 5.28 demuestra el aumento significativo en las puntuaciones incluso con este concepto que sólo se enseñó durante el encierro, con $p < 0,001$.

Este aumento se cuantificó utilizando la d de Cohen. Hay un efecto enorme ($d = 4,01$) con un porcentaje de cambio del 86%.

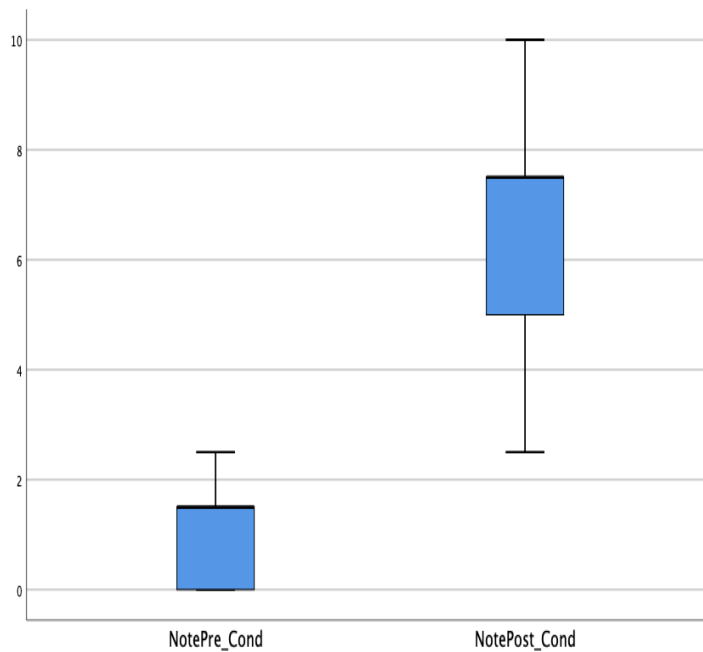


Figura 5.26 Diagramas de caja en la pre-test y post-test en el concepto de condicionales se enseña en las sesiones en línea durante la pandemia (Ocaña et al., 2020)

TEST	N	\bar{x}	Med	Sd
Pre	137	0,98	1,50	0,90
Pos	137	6,97	7,5	1,92

Tabla 5.27 Análisis descriptivo de la muestra para el concepto de condicionales, durante la pandemia (Ocaña et al., 2020)

t-test	Gf	p-valor
30,093	136	< 0,001

Tabla 5.28 Resultados de la prueba t para muestras pareadas entre pre-test y post-test en concepto condicional (Ocaña et al., 2020)

Se llevó a cabo una validez de constructo en preposteo utilizando análisis factorial exploratorio empleando rotación varimax y el criterio de extraer autovalores > 1.

También se calculó el alfa de Cronbach y su valor fue aceptable: 0,812. Como se indica en la Tabla 5.28, cinco ítems (1-4 y 10) del cuestionario se enfocaron en recopilar información sobre las opiniones de los estudiantes.

Tres ítems (5-8) se enfocaron en recopilar información sobre la satisfacción de los estudiantes. Un ítem (9) se centró en la información sobre la motivación de los estudiantes. La Tabla 5.29 resume los resultados.

En cuanto a la opinión, el 49,68% de los estudiantes respondió que eligió más las actividades de Alcody que las recetas (26,28%) o charlar con Alcody (24,09%). Cuando se preguntó a los estudiantes por su actividad favorita de Alcody, el 50,36% respondió el acertijo del lobo, la cabra y el repollo; 29,94% de ranas saltarinas; y 19,71% misioneros y caníbales.

Cuando se les pide a los estudiantes que elijan su receta favorita, el 43,07% dijo que la hamburguesa; 35,04% el desayuno; 11,68% la tortilla; y 10,22% la pasta.

La ayuda de Alcody resultó muy útil para el 71,53% de los estudiantes. El 21,9% respondió que la ayuda fue útil.

Solo dos estudinates mencionaron que la ayuda brindada por Alcody no les sirvió para nada porque no la habían necesitado. Asimismo, se calculó el alfa de Cronbach, siendo su valor aceptable: 0,887.

Con respecto al uso de Alcody para programar, el 64,23% de los estudiantes pensó que podría programar con la ayuda de Alcody, el 23,4% pensó que podría programar con la ayuda de Alcody en breve, el 7,3% dijo que no estaba seguro de si podrán programar incluso con la ayuda de Alcody, y el 5,1% pensó que podrán programar con la ayuda de Alcody en el futuro.

En cuanto a la satisfacción, los alumnos se mostraron muy satisfechos con su interacción con Alcody y con la recepción de premios.

Casi todos (98,5%) los estudiantes respondieron que están satisfechos con Alcody, y el 95,6% de los estudiantes disfrutó de tener recompensas, el 93,4% disfrutó aprendiendo a programar usando Alcody y el 94,4% consideró que Alcody los motivó a aprender a programar incluso durante la pandemia de COVID-19 cuando estaban en casa usando Alcody en línea.

Ítem	Pregunta	Respuesta
1	¿Qué tarea has elegido más a menudo para completar con Alcodey?	Actividades (49,6%), Recetas (26,3%), Cuestionarios (24,1%)
2	¿Cuál es tu tarea favorita con Alcodey?	Actividades (59,1%), Recetas (36,5%), Cuestionarios (4,4%)
3	¿Cuál es su actividad favorita?	Acertijo de lobo, cabra y repollo (50,4%), Misioneros y caníbales (19,7%), Ranas saltarinas (29,9%)
4	¿Cuál es tu receta favorita?	Desayuno (35%), Tortilla (11,7%), Hamburguesa (43,1%), Pasta (10,2%)
5	¿Te gusta la ayuda que te ofrece Alcodey?	Sí (93,4%)
6	¿Estás satisfecha con obtener recompensas?	Sí (95,6%)
7	¿Está satisfecho con su interacción con Alcodey?	Sí (98,5%)
8	¿Te gusta aprender a programar con Alcodey?	Sí (93,4%)
9	¿Crees que Alcodey te motiva a programar?	Sí (94,4%)
10	¿Crees que podrás programar con la ayuda de Alcodey?	Sí (92,7%)

Tabla 5.29 Análisis de los resultados del cuestionario pre-covid (Ocaña et al., 2020)

Ítem	Pregunta	Respuesta más seleccionada /%
1	¿Crees que puedes concentrarte en casa usando Alcodey?	Sí (92%)
2	¿Estás motivado para seguir usando Alcodey?	Sí (100%) - 7,84 valor medio (0 min-10 escala máxima)
3	¿Te gusta la ayuda que te ofrece Alcodey?	Sí (98,5%)
4	¿Te gusta que Alcodey te ayude a afrontar la pandemia de COVID-19?	Sí (91,2%)
5	¿Qué recomendaciones quieres que te dé Alcodey COVID para ayudarte a seguir aprendiendo a programar con él?	Cómo prevenir la infección por COVID (83,2%)

Tabla 5.30 Análisis de los resultados del cuestionario durante Covid-19 (Ocaña et al., 2020)

Factor	Antes	Durante
Motivación	94,4% (Q9)	100% (Q2)
Ayudar	93,4% (Q5)	98,5% (Q3)
Opinión	98,5% (Q7)	92% (Q1)
Satisfacción	95% (Q5-Q8)	95% (Q3-Q4)

Tabla 5.31 Comparación de resultados antes y durante Covid-19 (Ocaña et al., 2020)

En Ecuador, el confinamiento comenzó en marzo de 2020. Los estudiantes estuvieron en casa sin la posibilidad de salir desde entonces, pero pudieron trabajar con Alcody en línea durante ese tiempo.

Podían progresar con el concepto de condicionales y revisar los conceptos anteriores con el chat y continuar interactuando con las recetas y actividades.

A finales de mayo de 2020, se solicitó a los estudiantes que completaran el cuestionario que se muestra en la Tabla 5.23. Los resultados se muestran en la Tabla 5.29.

El 92% de los estudiantes respondieron que creen que podrían concentrarse en casa. Este podría ser el motivo que explique que están motivados para seguir aprendiendo con Alcody.

En una escala de 0 (motivación mínima) a 10 (motivación máxima), el valor medio fue de 7,84. Este nivel de motivación es incluso mayor que antes del confinamiento, como se puede ver en la Tabla 5.31.

En cuanto a la ayuda brindada por Alcody, al 98,5% de los estudiantes les gustó la ayuda durante la pandemia COVID-19: el 56,9% para estudiar entradas / salidas y el resto para estudiar condicionales. Las recomendaciones y ayudas de Alcody en relación a la pandemia COVID-19 fueron apreciadas por el 91,2% de los estudiantes.

Durante la observación directa con Alcody, los observadores coincidieron en que los estudiantes estaban comprometidos y motivados durante todas las sesiones con Alcody.

Todos los observadores vieron que los estudiantes se sentaron frente a sus ordenadores y estaban completando las actividades, siguiendo las recetas y escribiendo los programas.

A veces, cuando tenían dudas, los estudiantes se las hacían al docente pero la mayoría de las veces utilizaban la ayuda que les brindaba Alcody en las tutorías y trabajaban duro por su recompensa. Cuando recibieron una recompensa se sintieron realmente felices y emocionados.

Algunos programas correctos escritos por estudiantes fueron:

1 iniciar

2 escribir "hola"

3 finalizar

1 iniciar

2 escribir "Escribe el valor de un lado:"

3 leer A

4 $B = A + A + A + A$

5 escribir "El perímetro es", B

6 finalizar

Algunos programas incorrectos escritos por estudiantes fueron los siguientes. Se han seleccionado porque representan un error común cometido por al menos diez estudiantes.

1 comienzo

2 escribe "escribe la variable de saludo"

3 lee hola

En este caso, el error está relacionado con el uso de variables y palabras clave de pseudocódigo. Los estudiantes tuvieron, al principio, dificultades para comprender el concepto de variable.

Podría deberse a que era un uso diferente al de Scratch. Con Alcodey, los estudiantes tenían que escribir la instrucción en lugar de arrastrar y soltar los bloques, por lo que tenían que identificar dónde colocar la variable sin la ayuda de huecos en las instrucciones.

El segundo error común fue cambiar el nombre de una instrucción de pseudocódigo. Dado que en Scratch no tenían que escribir la instrucción, no necesitaban aprender la palabra correspondiente. Con Alcodey, muchos estudiantes preguntaron si "final" y "algoritmo final" eran lo mismo.

Después de seis meses de enseñanza, la mayoría finalmente pudo (según lo revelado por las puntuaciones posteriores a la prueba) comprender cómo usar correctamente las variables en los programas y las instrucciones del pseudocódigo.

Algunos comentarios incluyeron:

- "Alcody es muy útil para nosotros". "Es muy interesante". "Tiene buenas actividades".
- "Es interesante porque puedes hablar con Alcody y no es solo un programa aburrido".
- "Me gusta el botón de ejecución de Alcody y las posibilidades de depuración. Si Alcody no las tuviera, no sabría qué hacer".
- "Alcody nos ayuda a mejorar nuestro aprendizaje con recetas".
- "Alcody es una forma de mejorar mi programación".

Es importante resaltar que los estudiantes notaron y realmente apreciaron la posibilidad de ejecutar los programas con depuración y ayuda, y el manejo de las emociones para que no se sintieran solos al escribir programas, sino que reconocen el apoyo emocional de Alcody.

Los estudiantes disfrutaron de las actividades divertidas y siguieron las recetas y cada vez que completaban algunos programas, pedían tiempo con las actividades y recetas. A veces se sentían frustrados cuando su programa no funcionaba como esperaban, pero el apoyo emocional de Alcody y las posibilidades de depuración los guiaron para completar la actividad correctamente.

Parece que la recompensa otorgada y la posibilidad de actividades y recetas posteriores los motivó a seguir probando más programas y pensando en cómo redactarlos correctamente para lograr los objetivos propuestos por su "amigo", Alcody.

Los docentes acordaron continuar usando Alcody durante la pandemia de COVID-10, por esto se prepararon las lecciones del concepto de condicionales para impartirse en línea.

Los estudiantes estaban felices de seguir usando Alcody en casa. Los registros revelan que los estudiantes se conectaron en línea con su maestro y usaron Alcody al mismo tiempo que lo habrían hecho en clase. Al evaluar los programas redactados por los estudiantes, la puntuación media fue de 7,77 en una escala de 0 (incorrecto) a 10 (correcto). Algunos programas correctos escritos por los estudiantes fueron:

1 iniciar

2 escribir "Escribe tu edad"

3 leer edad

4 si edad >= 10 escribir "Más de 10 años" sino escribir "Menos de 10 años"

5 finalizar

Los errores comunes están relacionados con instrucciones confusas, como la siguiente muestra en la que el estudiante ha confundido leer con escribir, pero la idea general, el uso de variables y la estructura condicionales con entrada y salida son correctas.

1 iniciar

2 leer "Escribe tu edad", edad

3 si edad \geq 18 entonces escribir "Eres un adulto" sino escribir "No eres un adulto"

4 fin

Algunos comentarios proporcionados por los estudiantes incluyen:

- "Me gusta Alcody para dar consejos sobre prevención y cuidados".
- "Me gusta que Alcody me explique por qué debería estar en casa".
- "Me gusta jugar al aire libre pero teniendo Alcody puedo jugar con él y aprender a programar".
- "Me gusta estar en casa con mi familia y puedo seguir usando Alcody en línea".

En resumen, antes del COVID-19 lo usaban en clase con sus compañeros y docentes. Estaban enfocados en su tarea, comprometidos, motivados y felices de tener la posibilidad de seguir aprendiendo programación con un compañero de aprendizaje como Alcody.

Durante COVID-19, los estudiantes usaron Alcody en casa conectados al mismo tiempo que sus lecciones (una hora por semana) con su maestro.

Los estudiantes pudieron crear programas como antes, pero lograron un progreso significativo con nuevos conceptos como condicionales, mientras revisaban los conceptos anteriores para crear programas más complejos con la ayuda de Alcody.

Es importante resaltar que tanto antes como durante el confinamiento, el docente solo asumía el rol de asistente en caso de que los estudiantes no pudieran resolver algún problema con Alcody, pero la mayoría de las veces los estudiantes estaban interactuando directamente con Alcody en clase o en hogar usando Alcody como guía.

El compañero de aprendizaje, en este caso Alcody, está personalizado para los alumnos. Proporciona retroalimentación de acuerdo a las necesidades de cada niño y lo guía hacia la construcción de su solución sin decir la solución o dar una respuesta correcta.

Con Alcody, la posibilidad de interactuar en lenguaje natural (pseudocódigo) para escribir los programas, ejecutarlos y depurarlos; empodera a los niños con las herramientas para llegar a su solución, sin sentirse solos, incluso durante la pandemia de COVID-19.

Los resultados respondieron a la RQ1: ¿Los estudiantes (de entre 10 y 12 años) aumentaron sus puntuaciones de aprendizaje en una prueba pre-post usando Alcody antes

y durante la pandemia de COVID-19? Para el concepto de entrada / salida (Q1, Q2, Q3), los participantes mejoraron significativamente desde el pre-test hasta el primer post-test (antes de COVID-19) y el segundo post-test (durante la pandemia de COVID-19).

Además, las puntuaciones del segundo post-test mostraron una mejora significativa con respecto al primer post-test. Para el concepto de condicionales (Q4), los participantes también mejoraron significativamente desde el pre-test al segundo post-test (durante la pandemia de COVID-19) con un porcentaje de cambio del 86%.

La razón por la que las condiciones se aprendieron tan bien podría encontrarse en que el concepto de condicional está muy relacionado con el concepto de condiciones de la vida real y los niños durante el confinamiento tenían muchos recursos en casa para interactuar.

Por ejemplo, si la instrucción de la condición fuera "si bebo toda el agua del vaso, el vaso estará vacío". Los niños pueden beber toda el agua del vaso de agua para ver un vaso vacío.

Además, el hecho de que los niños estuvieran en casa con todos sus recursos físicos y objetos, y se concentraran solo en la interacción con el programa (sin distracciones de otros compañeros hablando o haciendo bromas) facilitó la enseñanza de las condiciones según lo registrado por los observadores, incluso durante la pandemia por COVID-19.

Los resultados obtenidos respondieron las preguntas planteadas:

RQ2: ¿Cuán satisfechos estaban los estudiantes al interactuar con Alcodey antes y durante la pandemia de COVID-19? ¿Qué opinión tenían de Alcodey? ¿Estaban motivados para usar Alcodey incluso durante el encierro?

En general, estaban satisfechos ya que el 95% de los estudiantes estaban motivados para usar Alcodey antes de COVID-19 (promedio calculado a partir de las preguntas 5 a 8, (ver Tabla 5.24) y el 95% permaneció motivado para continuar aprendiendo con Alcodey (promedio calculado a partir de las preguntas 3 a 4 (ver Tabla 5.25).

Los estudiantes usaron Alcodey antes y durante la pandemia de COVID-19 y su motivación y opinión se mantuvo igual (más del 90%) como se muestra en la Tabla 5.26.

La reformulación de las nueve pautas como recomendaciones e implicaciones en el caso de estudiantes que ya han utilizado Scratch y están interactuando online con un compañero de aprendizaje son las siguientes:

1. Interacción social: dijeron que la colaboración entre los miembros del equipo es una parte vital de las actividades de codificación. Donde esta interacción social no es posible debido al confinamiento, se propone el uso de un compañero de aprendizaje que pueda continuar enseñando programación a los niños de acuerdo con el construccionismo de Papert (1980).

2. Diseño apropiado según la edad: los diferentes grupos de edad (niños y adolescentes) necesitan diferentes enfoques y diseños para participar en una actividad de codificación. Alcodey permite personalizar tanto el entorno como las actividades. Con el diálogo, se pide a los niños que construyan su programa y se les orienta hacia su solución de acuerdo no solo con su edad sino también con sus respuestas particulares.
3. Duración de la actividad: Según el construccionismo, cuando los niños utilizan herramientas tecnológicas, la duración es clave para que se involucren personal, intelectual y emocionalmente. Alcodey se usó en línea durante el confinamiento de COVID-19 siguiendo el mismo horario de clases presenciales. La duración de la actividad debe ser suficiente para que lleguen a una solución, pero no demasiado para que se aburran. Los estudiantes deben poder jugar de acuerdo con las recompensas de gamificación para mantenerse motivados.
4. Relevancia de la actividad y contenido significativo: Ofrecer un tema de apoyo para el proceso de creación de artefactos, en el que los participantes puedan participar de manera significativa en entornos de la vida real, es un factor clave para apoyar los elementos psicológicos y socioculturales para un aprendizaje efectivo. Los problemas que se piden que se resuelvan mediante programación deben extraerse de situaciones de la vida real. Por ejemplo, el compañero de aprendizaje podría pedir a los estudiantes que escriban un programa para calcular cuánto gastarán en regalos para sus amigos en función del costo de cada regalo y la cantidad de amigos.
5. Artefactos físicos y digitales: parece que la inclusión de tareas físicas es interesante y permitió a los participantes mejorar sus habilidades. Al usar Alcodey, algunos niños usaron lápiz y papel para dibujar ideas que les ayudaran a programar y otros estudiantes buscaron objetos físicos en su hogar para comprender los ejemplos de la vida real propuestos. Esto se recomienda para ayudar a los estudiantes a comprender los conceptos de programación.
6. Actitudes y motivación de los niños: El proceso de aprendizaje debe estar respaldado por tareas que alienten a los niños a reflexionar, los motiven a colaborar y les den razones significativas para completar sus artefactos. Alcodey pide a los niños que escriban programas que sean significativos para ellos y los anima a reflexionar sobre su programación. No se debe proporcionar una solución correcta. Por otro lado, los niños tienen que pensar en escribir su programa y ejecutarlo para comprobar si funciona. Si el programa funciona, están motivados. Si se sienten tristes, pueden decirle esto al compañero de aprendizaje que le dará ayuda por medio de la depuración y tutoriales hasta que comprendan lo que están haciendo mal y lo cambien para llegar a resolver el ejercicio.

7. **Sobrecarga cognitiva:** las actividades de codificación para niños pueden tener una carga cognitiva alta, lo que afecta su desempeño y experiencia general con las tareas. Alcodey reduce la carga cognitiva proporcionando solo un programa a la vez. En primer lugar, los programas solo se centran en un concepto, como entrada / salida o condicionales. A medida que los niños logren resolver esos problemas, se ofrecerán programas más complejos que combinan varios conceptos.
8. **Tareas apropiadas:** Las tareas deben hacer que los niños se interesen y puedan aprender. Como se indica en el punto 4, los programas en Alcodey están relacionados con situaciones significativas para que los niños se motiven pero aquí Alcodey ofrece la gamificación como recompensa adicional para mantener la motivación. Las tareas no solo deben estar relacionadas con la programación como programas de escritura, sino también como juegos para pensar cómo programar. Por ello, el compañero de aprendizaje debe tener la posibilidad de seguir una receta como metáfora de un programa jugando con los ingredientes, una tarea de cocina relacionada con la vida real, significativa y adecuada para pensar como un programador. Además, el compañero de aprendizaje debe contar con juegos como acertijos para que los niños sigan pensando con lógica, tareas espontáneas que mantengan la motivación y les den recompensas que sean interesantes para ellos.
9. **Marco significativo para la participación de los instructores:** En la construcción de un artefacto, los niños no están solos: los practicantes (por ejemplo, maestros y asistentes) y cualquier otra persona que sea responsable de la tarea de aprendizaje también están involucrados. Como señaló Papert, "compartir el problema y la experiencia de resolverlo permite que un niño aprenda de un adulto no 'haciendo lo que dice el maestro' sino 'haciendo lo que hace el maestro'". Antes del encierro, los niños estaban en su clase hablando sobre cómo resolver los problemas presentados por Alcodey, siguiendo las tutorías y completando los juegos espontáneos. El docente estaba disponible, solo como apoyo a los niños, si tenían algún problema. De manera equivalente, durante un encierro o cuando no sea posible la interacción cara a cara, los niños pueden conectarse en línea con su compañero de aprendizaje en una videoconferencia con su maestro como apoyo para resolver cualquier problema y compartir sus comentarios que eviten cualquier sentimiento de aislamiento, y tener aún más apoyo con sus padres en casa.

5.5.3.2 Amenazas a la validez

El estudio presentó algunas amenazas a la validez:

Validez de constructo: Pueden presentarse el efecto novedad, ya que el uso de una herramienta informática en el aprendizaje siempre puede resultar atractivo y novedoso, algo que suele estar fuera de su rutina. Cuando se introduce una innovación, puede generar interés y entusiasmo que contribuyen al éxito, especialmente si se ha producido poca

innovación. Asimismo, la instrumentación se basa en cuestionarios para medir procesos cognitivo-emocionales. Se trata de fenómenos complejos y no fáciles de analizar. El uso de métodos físicos o multimodales podría proporcionar información adicional.

Validez externa: Dado que los grupos no son aleatorios, el estudio no tiene certeza de que la muestra sea representativa de la población general. En esta investigación, si bien los alumnos no se encuentran en un único escenario (los datos se recogieron cuando estaban en clase pero también en casa), no sería adecuado generalizar esto a todos los alumnos de su edad, teniendo en cuenta, además, la situación excepcional de pandemia global que surgió durante el desarrollo del experimento. Alcody debe probarse con niños sin conocimientos previos de programación y con un grupo de control. En un diseño de investigación de pre-test y post-test sin un grupo de control, el grado de control es bajo. Con un grupo de control, el grado de control sería mayor. Sin embargo, a veces con participantes humanos no es posible tener un grado de control tan alto.

Validez interna: aunque en este grupo único pre-pos prueba, parte de la validez interna se descarta automáticamente, por ejemplo, tasas diferenciales de mortalidad y sesgo de selección, se pueden presentar algunos efectos de maduración, cómo envejecer o tener más experiencia. Incluso en estudios breves, estos procesos son un problema. Otra amenaza son las habilidades digitales que involucran a los estudiantes, especialmente en el momento del encierro, donde el maestro está presente solo de forma remota y el equipo informático disponible en casa puede no ser el mismo para todos los estudiantes. Aunque todos los niños informaron que tenían un ordenador, algunos podían ser más rápidos que otros. Del mismo modo, la conexión a Internet podría ser más rápida en algunos hogares.

Validez de las conclusiones: Cook y Campbell (1979) lo llamaron validez de conclusión estadística (SCV). Destacaron que el SCV "se preocupa por las fuentes de error aleatorio y por el uso adecuado de estadísticas y pruebas estadísticas", por lo que no existe una validez perfecta. Aunque se proporciona un análisis descriptivo junto con todos los supuestos necesarios en los estudios inferenciales, los errores de tipo 1 (rechazo incorrecto de la hipótesis nula) y los errores de tipo 2 (no rechazar una hipótesis nula falsa) son parte de las pruebas estadísticas, un resultado perfecto no se puede garantizar, aunque se puede minimizar. Se han presentado pruebas de confiabilidad del diseño del cuestionario y del proceso de prueba pre-post. También se han probado las suposiciones sobre la normalidad (requerida para todas las pruebas paramétricas utilizadas) y la homocedasticidad (también requerida en el procedimiento ANOVA) en los datos.

5.6 Fase 6: Sesiones prácticas

Después de validar de manera inicial el sistema interactivo educativo, se procede a llevarlo al aula, acorde con la planificación de contenidos de programación que se acordó con los docentes en las sesiones iniciales.

Aprovechando que el sistema interactivo educativo trabaja en una plataforma web, las sesiones académicas pudieron darse de manera presencial o en línea.

5.6.1 Sesiones presenciales

Los estudiantes, durante su horario de clase acuden al laboratorio de computación por una hora, en dos ocasiones por semana. Cada estudiante utiliza de manera individual su ordenador asignado. Ingresando a Alcody con su usuario y contraseña (ver Figura 5.27).

El docente da un breve repaso del tema anterior, generalmente usando una lluvia de ideas y presenta el nuevo tema, soportado en presentaciones o tutoriales. Bajo su supervisión los estudiantes realizan ejercicios de programación propuestos (ver Figura 5.28).

Las evaluaciones de aprendizaje deben contar con un tiempo máximo establecido.

Los docentes pueden contar con la posibilidad de revisar la resolución de ejercicios en el sistema interactivo educativo.

Para las encuestas de satisfacción se recomienda el uso de formularios en línea.



Figura 5.27 Los estudiantes acceden al sistema interactivo educativo, Alcody (fuente: propia)



Figura 5.28 El docente supervisa el proceso de aprendizaje (fuente: propia)

4.6.2 Sesiones virtuales

El docente y los estudiantes se conectan por 40 minutos por Zoom a la videoconferencia, acceden en el horario establecido, con su usuario y contraseña al sistema interactivo educativo (ver Figura 5.29).

El docente repasa el tema anterior motivando la participación de los estudiantes, para continuar presentando el nuevo tema, soportado en presentaciones o tutoriales.

Los ejercicios propuestos en cada tema se realizan de manera conjunta con los estudiantes, una vez finalizada esta actividad se da un tiempo para que los estudiantes resuelvan ejercicios propuestos de manera individual (ver Figura 5.30).

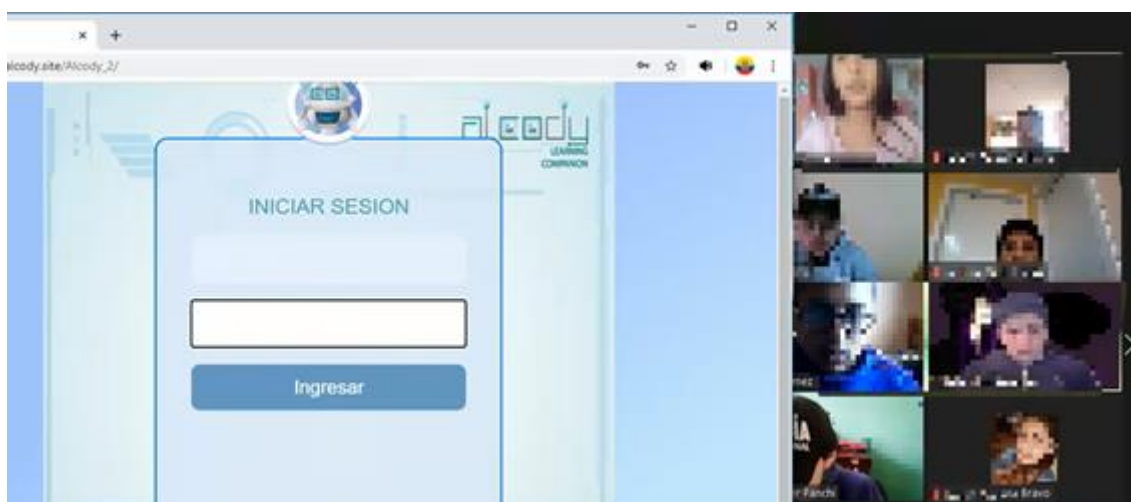


Figura 5.29 El docente presenta el tema de clase (fuente: propia)

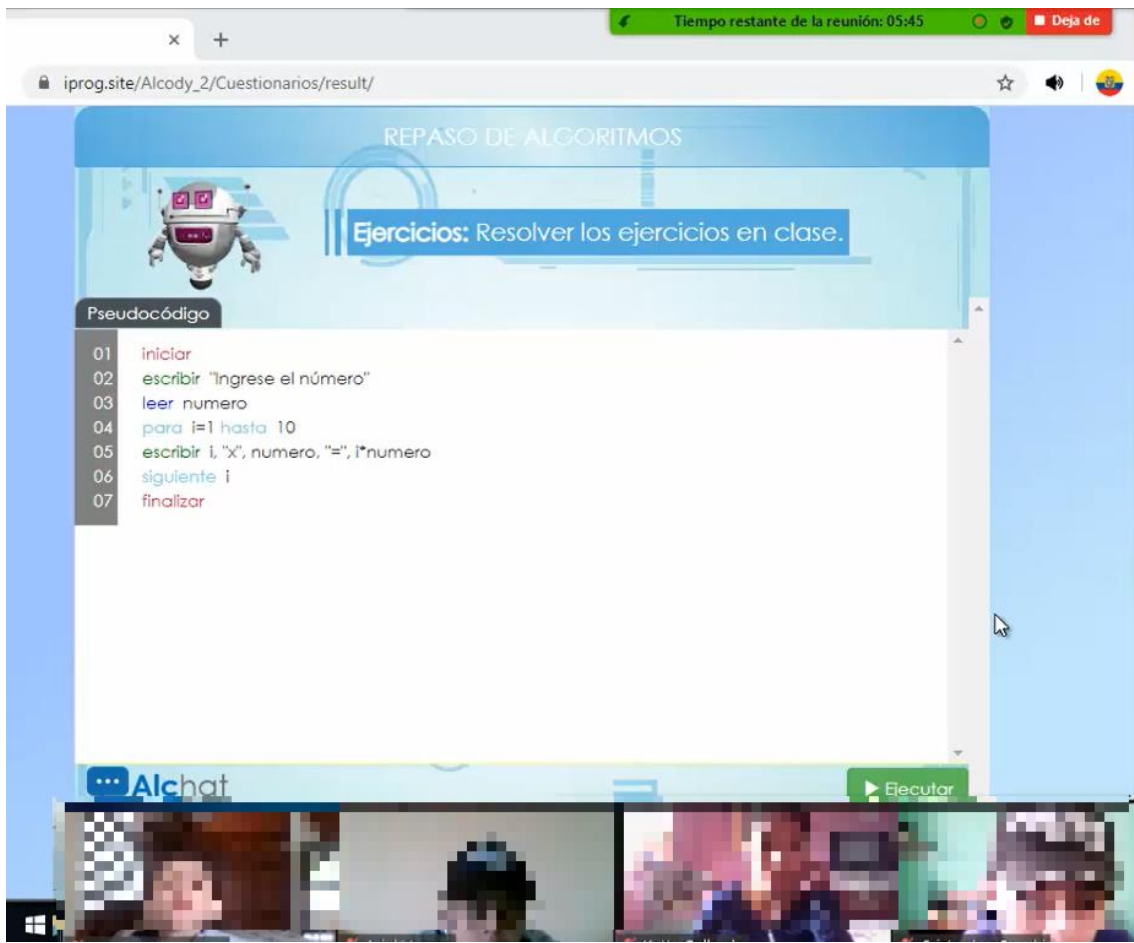


Figura 5.30 Resolución de ejercicios propuestos (fuente: propia)

5.6.3 Interfaces para la integración

Para el proceso de integración en el aula, se considera en la propuesta del compañero de aprendizaje, las siguientes interfaces:

5.6.3.1 Inicio de sesión

En la Figura 5.31 se observa la interfaz del inicio de sesión para usuarios registrados en el sistema interactivo educativo Alcody.

¹ A los estudiantes se les asigna un nombre de usuario compuesto de año lectivo, grupo o paralelo y número de lista. ² La contraseña inicial fue sencilla y fácil de recordar. ³ Durante el experimento, el ingreso se verificaba conforme con su horario de clase.



Figura 5.31 Ingreso de usuarios en Alcody (fuente: propia)

5.6.3.2 Perfil de usuario

En la Figura 5.32 se observa la interfaz para la edición del perfil usuarios registrados en el sistema interactivo educativo Alcody.

¹ Los usuarios pueden modificar su nombre de perfil (el nombre que usa Alcody para interactuar con ellos), pero no su nombre de usuario. ² Los usuarios tienen una serie de imágenes para su imagen de perfil, incluso si lo prefieren pueden cargar una desde su ordenador.

³ Entre otras opciones, cuenta en esta interfaz con la opción de cambiar su contraseña y luego de ellos enviar a actualizar su información.

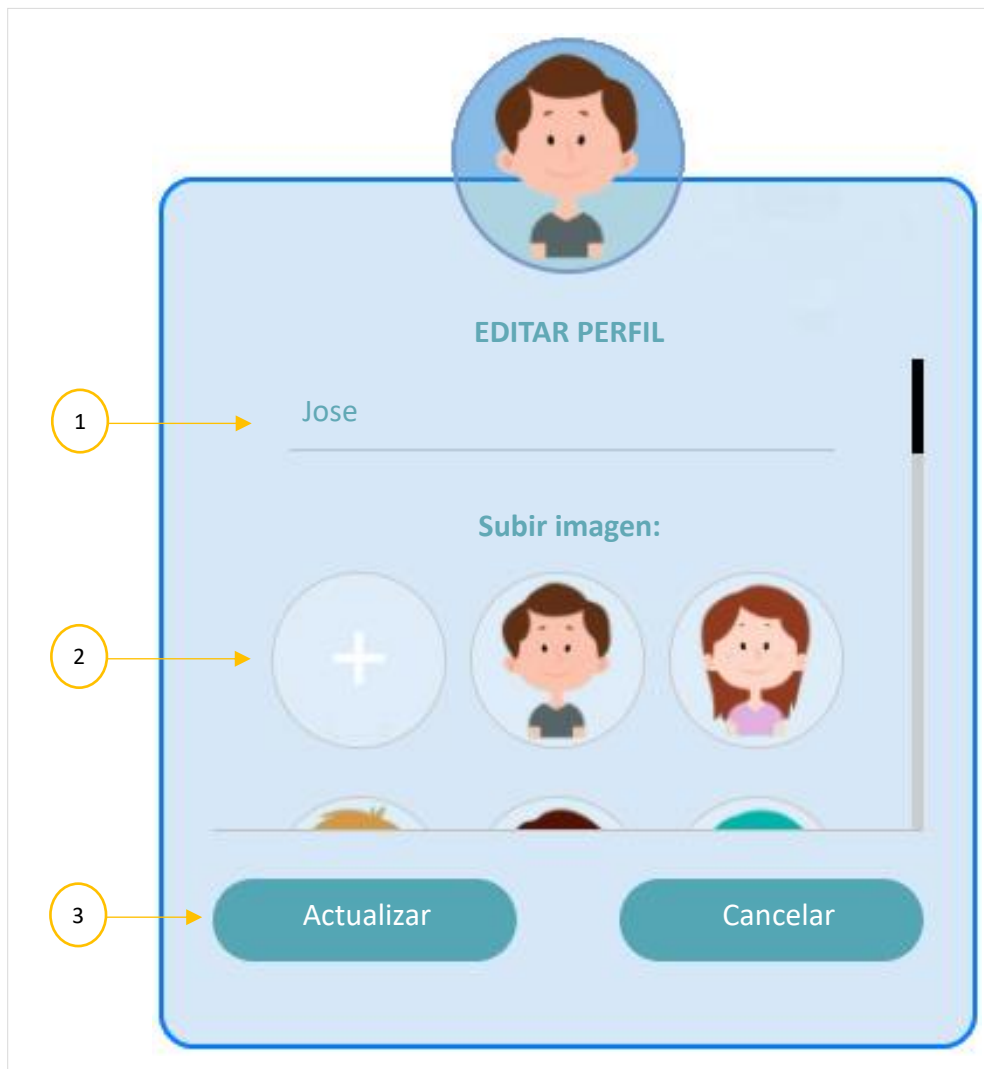


Figura 5.32 Editar perfil de usuario en Al Cody (fuente: propia)

5.6.3.3 Opciones generales

En la Figura 5.33 se muestra la interfaz de las opciones generales que tiene el entorno de aprendizaje Al Cody.

- 1 Cada estudiante tiene un trato personalizado, mediante su nombre e imagen de perfil.
- 2 El menú de opciones para la edición de perfil y cambio de contraseña.
- 3 El compañero de aprendizaje Al Cody.
- 4 El chat del gestor de diálogo, para la interacción Al Cody-estudiante.
- 5 Los botones de opciones para las actividades, recetas y cuestionarios.

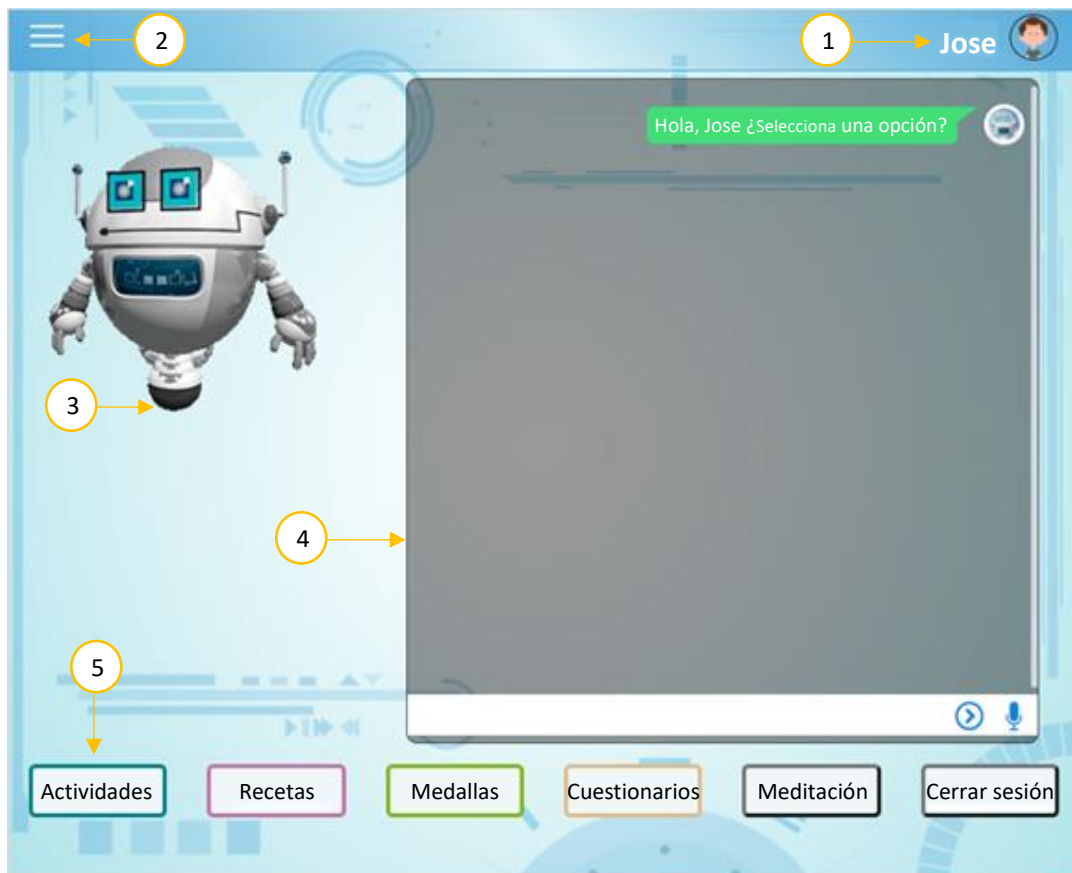


Figura 5.33 Interfaz de opciones generales del sistema interactivo educativo, Al Cody (fuente: propia)

5.6.3.4 Chat del gestor diálogo

En la Figura 5.34 se observa la interfaz del chat del gestor de diálogo del sistema interactivo educativo Al Cody.

1 Personalización de imagen de usuario. 2 Acorde con las características del co-diseño, se maneja una estructura tipo WhatsApp. 3 Al Cody se expresa de manera amigable 4 Fondo con colores, similares al co-diseño. 5 Cuadro de texto, para escritura del mensaje.

6 Scroll en barra vertical de para mensajes antiguos. 7 Uso de colores distintos para fondos de mensaje. 8 Botones de direccionamiento de contenido. 9 Activar, corrección ortográfica en mensajes. 10 Uso del micrófono para mensajes por voz.



Figura 5.34 Chat del gestor de diálogo del sistema interactivo educativo, Alcody (fuente: propia)

Durante la interacción con Alcody en el gestor de diálogo, el estudiante puede escribir o hablar sobre su intención de ver tutoriales sobre: contenido inicial de pseudocódigo, entrada-salida, condicionales o bucles (ver Figura 5.35).

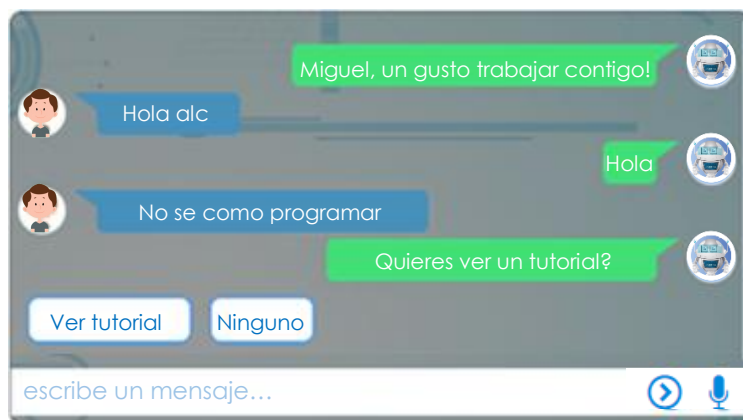


Figura 5.35 Invocación a tutoriales en el gestor de diálogo (fuente: propia)

5.6.3.5 Tutoriales

En la Figura 5.36 se observa un ejemplo de tutorial de aprendizaje de programación, estos se invocan por parte del usuario desde el chat del gestor de diálogo del sistema interactivo educativo Alcodey.

Van Merriënboer y Krammer (1987) plantearon un enfoque que permite enseñar programación basado en la lectura de códigos, considerando a éste como más efectivo. En este sentido, en Alcodey los estudiantes primero leían ejemplos de programas en los tutoriales y después ya pasaban a escribir sus propios programas.

1 Contenido introductorio del tema. 2 Ejercicios propuestos para el contenido de aprendizaje. 3 Comprobación del aprendizaje por medio de preguntas cortas. 4 Área de presentación del contenido.



Figura 5.36 Tutoriales de aprendizaje de programación (fuente: propia)

5.6.3.6 Recetas

En el sentido de gamificar para motivar a los estudiantes, reconociendo sus logros, generando en ellos la intención de repetir el aprendizaje, hasta lograr mejorar su proceso académico, se ha considerado gamificar MECOPROG en el sistema interactivo educativo.

En la Figura 5.37 se observa la interfaz de la actividad receta, del sistema interactivo educativo Alcodey, las recetas con las que los estudiantes pueden realizar un proceso lógico secuencial, para resolver los algoritmos propuestos, son:

- Desayuno.
- Tortilla de patata.
- Hamburguesa.
- Ensalada.

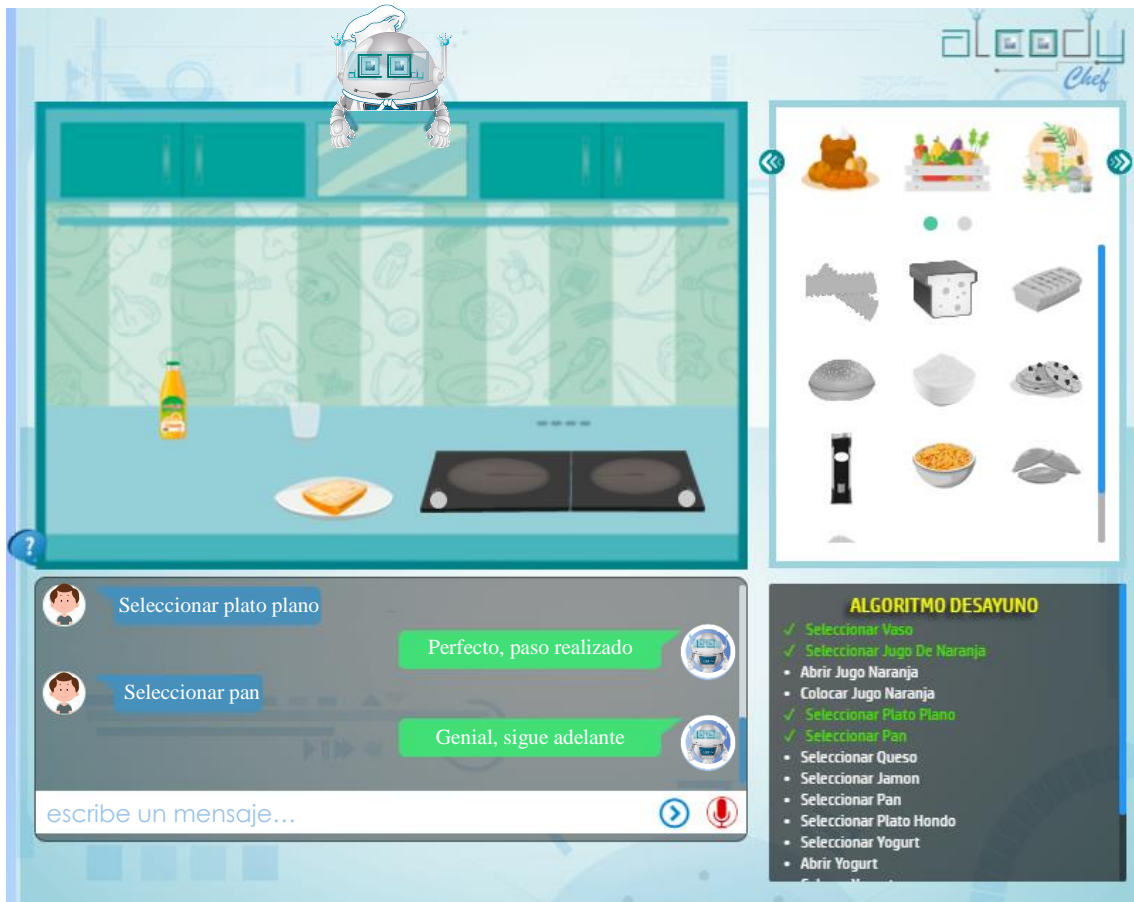


Figura 5.37 Gamificación de MECOPROG, Alcodey (fuente: propia)

En la Figura 5.38 se puede observar la distribución de elementos en la interfaz de la actividad recetas.

1 El compañero de aprendizaje en esta interfaz es AlcodeyChef. 2 Espacio de interacción y ejecución de la receta, en estas actividades se incorpora la posibilidad de usar el ratón, usando el arrastrar y soltar. 3 Icono de ayuda, que permite visualizar un video de explicativo.

4 Las instrucciones del algoritmo se pueden ingresar por el chat. 5 Alcodey responde los mensajes a las instrucciones enviadas. 6 En relación al co-diseño y la gamificación, se dan mensajes amigables y de motivación. 7 Funcionalidad del chat del gestor de diálogo.

8 Las recetas en Alcodey, cuentan con 6 categorías: carnes y embutidos; lácteos y líquidos; utensilios y vajillas; harinas y pan; frutas y vegetales; y, esencias y aderezos. 9 Cada una de estas categorías cuenta con elementos respectivos.

10 Algoritmo propuesto para la resolución de la receta. 11 A medida que se van realizando las instrucciones estas se colocan de color verde. 12 Los pasos que faltan para completar la receta se muestran de color blanco, el orden de las instrucciones no son obligatorios, pero si deben ser lógicos.

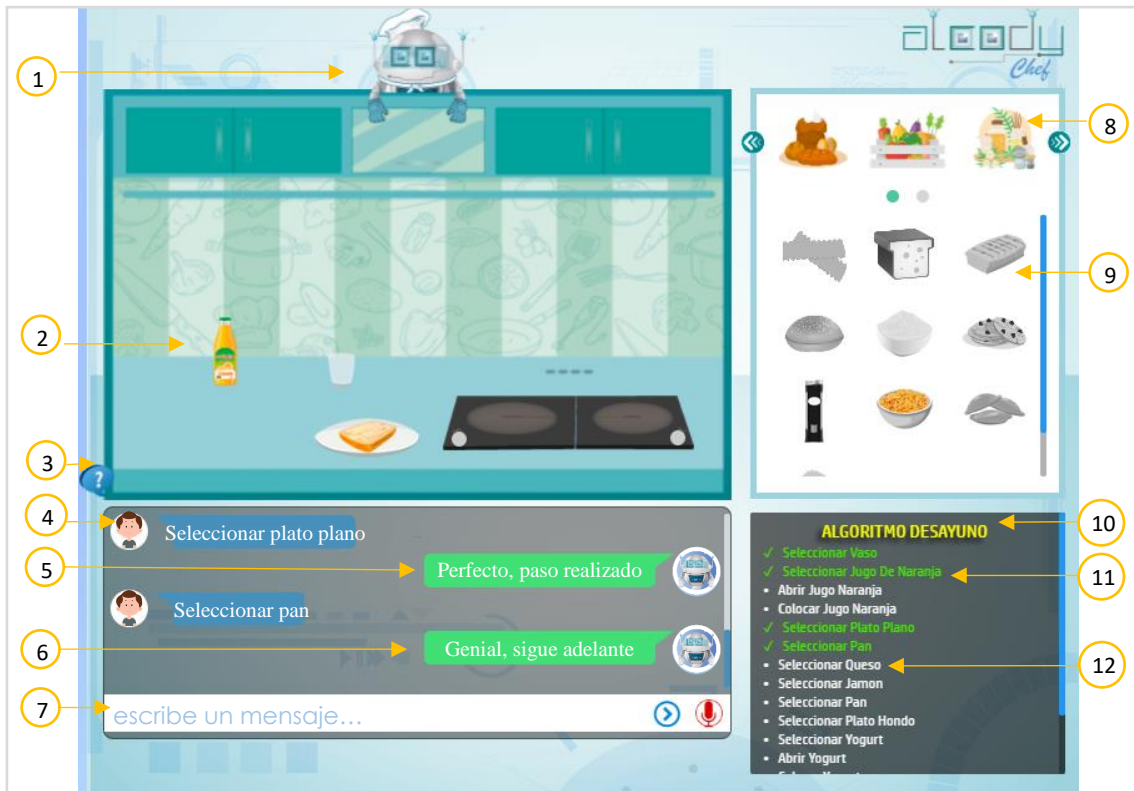


Figura 5.38 Interfaz de las recetas en Alcodey (fuente: propia)

5.6.3.7 Medallero

La Figura 5.39 muestra el medallero de Alcodey. 1 El nombre del perfil de usuario. 2 Las medallas de las actividades y recetas del sistema interactivo educativo, estas se presentan de dos maneras: gris (actividad no superada) y a color (actividad no superada). 3 El porcentaje de avance.



Figura 5.39 Medallero de actividades en Alcody (fuente: propia)

5.6.3.8 Editor de programación

En la Figura 5.40 se visualiza el editor de código del sistema interactivo educativo Alcody, y de igual manera se describe la interfaz del proceso de ejecución (ver Figura 5.41).

1 Título del cuestionario. 2 Pregunta propuesta. 3 Espacio de edición. 4 Numeración del código 5 Color predefinido en las instrucciones que forman parte del listado de tokens. 6 El apóstrofe permite colocar comentarios en el código. 7 Icono de Alchat que visualiza el chat del gestor de diálogo. 8 Botón de ejecución. 9 Avanzar o retroceder de pregunta.

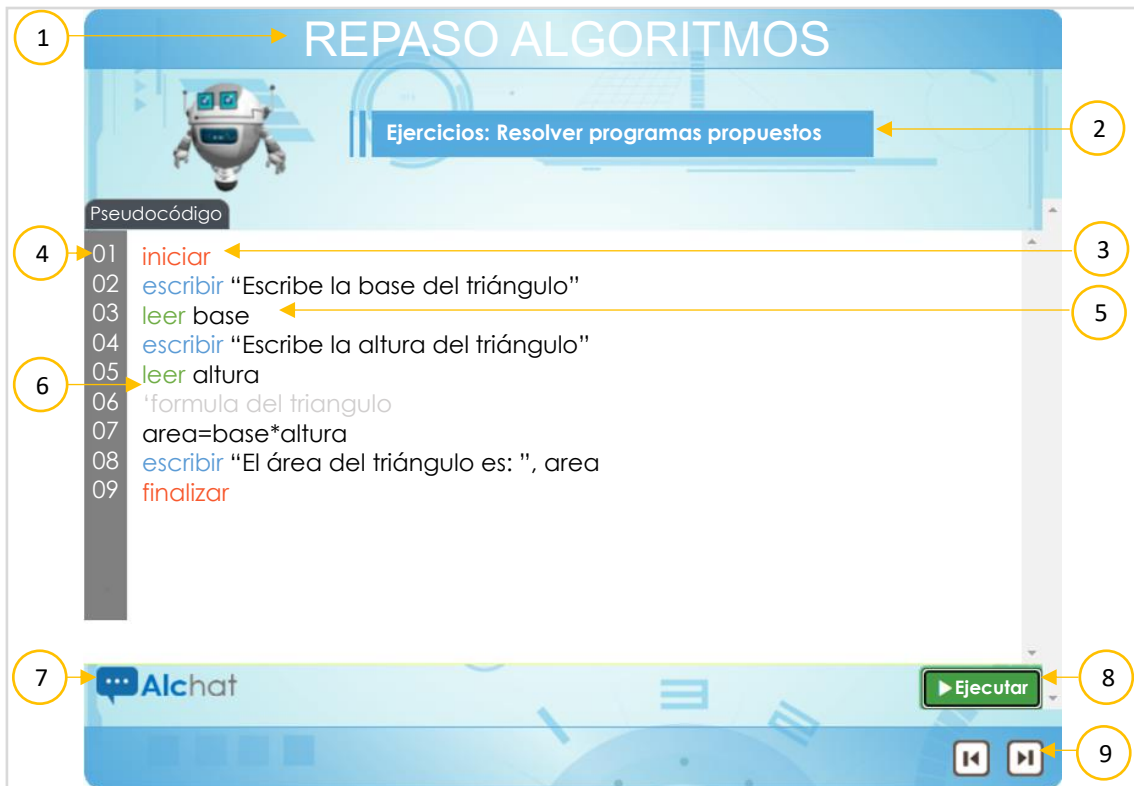


Figura 5.40 Editor de código en Alcodey (fuente: propia)



Figura 5.41 Proceso de ejecución en Alcodey (fuente: propia)

En la Figura 5.42 se visualiza un ejemplo de los mensajes predictivos del depurador de Alcodey, en este sentido también se muestra un ejemplo de los mensajes del depurador de código de Alcodey al momento de intentar ejecutar el código (ver Figura 5.43).

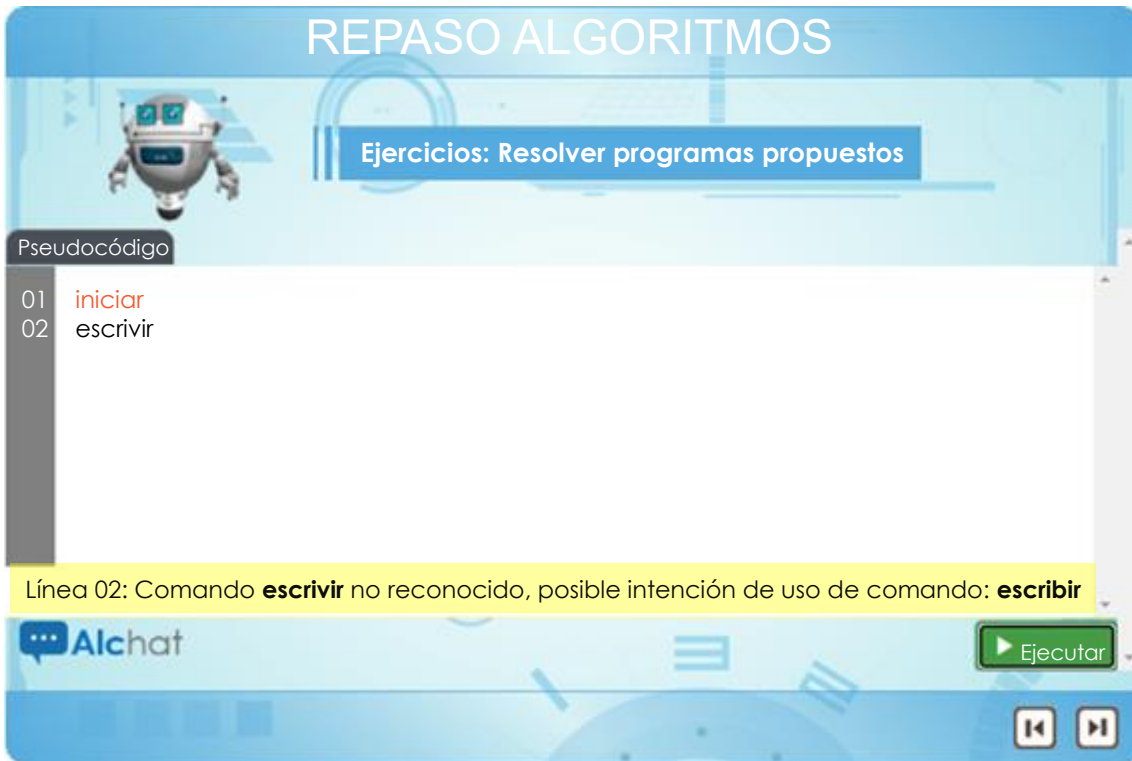


Figura 5.42 Mensajes predictivos del depurador de Alcodey (fuente: propia)

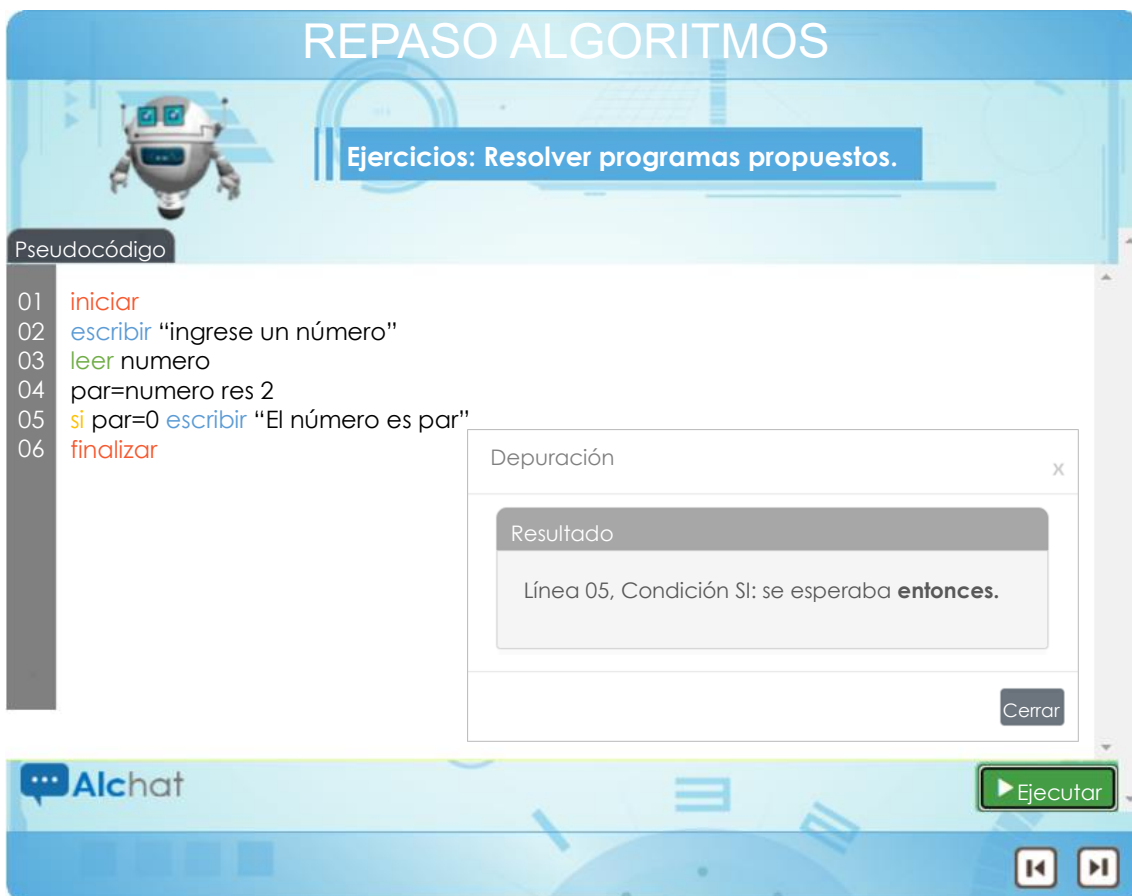


Figura 5.43 Mensajes del depurador de Alcodey (fuente: propia)

En la Figura 5.44 se visualiza el listado de cuestionarios que ha culminado los estudiantes:
 1 Título del informe. 2 Listado de cuestionarios. 3 Acceso directo para ingresar al cuestionario. 4 Filtro para activar/desactivar cuestionarios valorados.

GRUPO	USUARIO	TEST	INTENTO	FECHA	NOTA	REVISADO
GRUPO1	USUARIO 1	PROGRAMACIÓN	1	10-01-2019	8,00	✓ LINK
GRUPO1	USUARIO 2	PROGRAMACIÓN	2	10-01-2019	9,50	✓ LINK
GRUPO1	USUARIO 3	PROGRAMACIÓN	3	10-01-2019	8,00	✓ LINK
GRUPO1	USUARIO 4	PROGRAMACIÓN	1	10-01-2019	8,00	X LINK
GRUPO1	USUARIO 5	PROGRAMACIÓN	2	10-01-2019	8,10	✓ LINK
GRUPO1	USUARIO 6	PROGRAMACIÓN	1	10-01-2019	8,40	X LINK
GRUPO1	USUARIO 7	PROGRAMACIÓN	2	10-01-2019	7,90	✓ LINK
GRUPO1	USUARIO 8	PROGRAMACIÓN	3	10-01-2019	9,10	✓ LINK

Solo no revisados

Figura 5.44 Listado cuestionarios en Alcodey (fuente: propia)

En la Figura 5.45 se observa la resolución de cuestionarios por parte de los estudiantes:
 1 Título del cuestionario. 2 Detalle de pregunta. 3 Resolución por parte del estudiante. 4 Corrección del docente. 5 Puntuación. 6 Opciones de texto enriquecido. 7 Botones de grabación y movimiento.

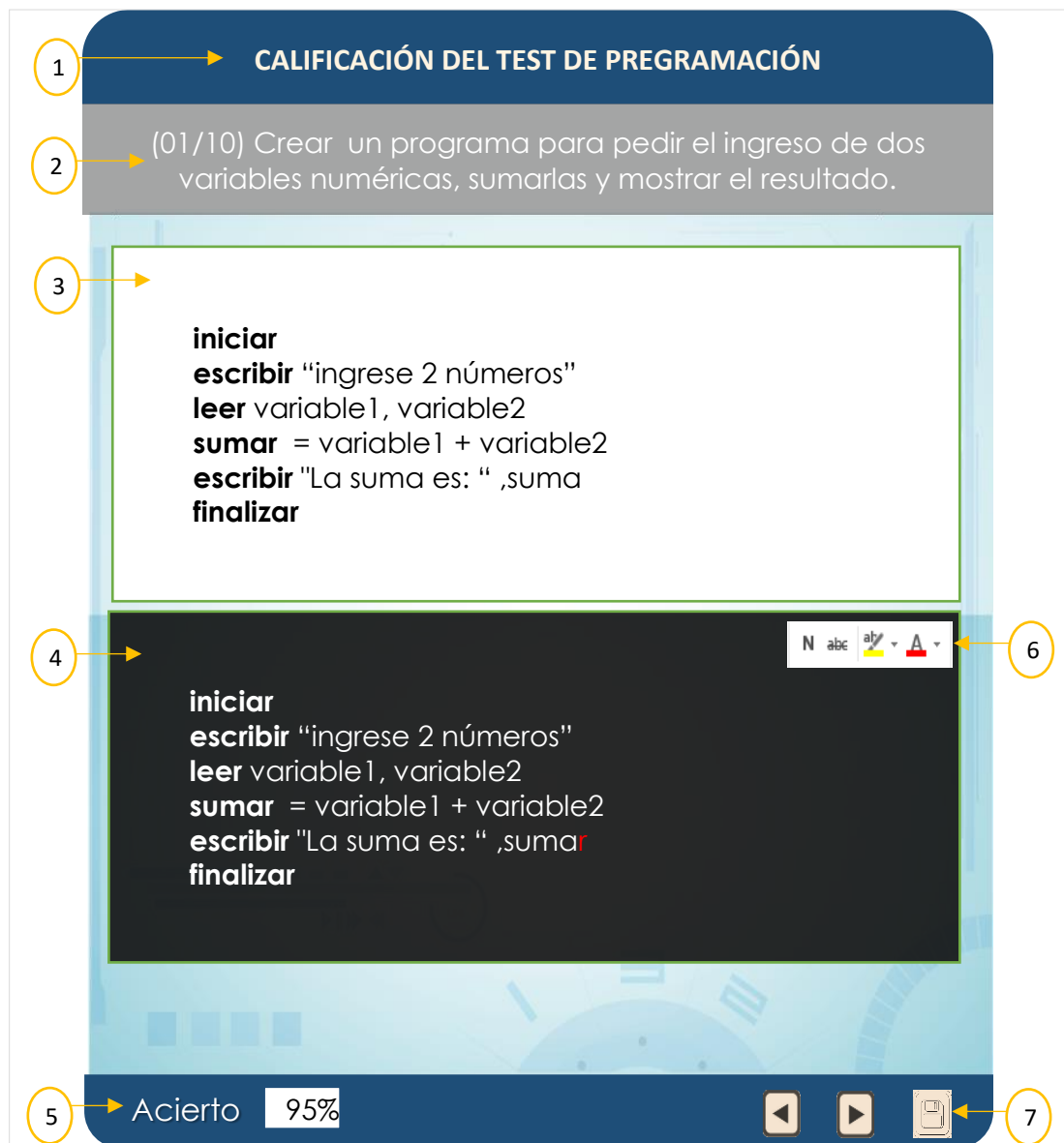


Figura 5.45 Resolución y calificación de preguntas (fuente: propia)

5.6.3.9 Encuestas de satisfacción

En la Figura 4.46, se observa en ejemplo de encuesta en Alcodey. 1 Identificación de usuario. 2 Preguntas de verdadero o falso. 3 Preguntas tipo escala de satisfacción.

ENCUESTA ALCODY

Jose

¿Crees que puedes concentrarte en casa usando Alcody?

Si

¿Estas motivado para seguir usando Alcody?

1 2 3 4 5 6 7 8 9 10

¿Estas satisfecho con su interacción con Alcody?

Si

Siguiente

Figura 5.46 Ejemplo de encuesta en Alcody (fuente: www.alcody.site/resultados)

Capítulo 6. Conclusiones

Este capítulo describe la consecución de los objetivos que se plantearon en el Capítulo 1, resaltando las principales contribuciones realizadas durante el proceso de investigación al área de Interacción Persona-Ordenador con la metodología MEDIE_GEDILEC y a la enseñanza de la programación con el compañero ALCODY creado con ésta metodología, para terminar con las principales líneas de trabajo futuro.

6.1 Objetivos cumplidos

Se procede a detallar los resultados derivados del trabajo de investigación:

- **Diseñar una metodología para la gestión del diálogo de los compañeros de aprendizaje que enseñan programación en Educación Primaria.**

Se satisface este objetivo en el Capítulo 4 con MEDIE_GEDILEC, propuesta metodológica para crear compañeros de aprendizaje que enseñen programación en Educación Primaria.

El Capítulo 5 aplica la metodología para crear el compañero de aprendizaje Alcody.

Las fases MEDIE_GEDILEC son: comunicación y co-diseño con docentes y estudiantes; análisis de diálogos; gestor de diálogo; editor de código; validación de interfaz y de funcionalidad; y sesiones prácticas.

- **Proponer un algoritmo que guíe la interacción y el diálogo para la enseñanza de conceptos de programación inicial.**

Se alcanza el objetivo propuesto en referencia a la adaptación de la interfaz de los prototipos mediante Storytelling y el análisis del diálogo definiendo parámetros que guíen el proceso (Ocaña et al., 2019).

En este sentido, en la Subsección 5.3.2 Desarrollo del gestor de diálogo, se muestra en la Figura 5.13 el esquema de funcionalidad del gestor de diálogo (AlcodyGDiag) que inicia con la entrada por parte del estudiante.

De esta manera las entradas (mensajes enviados por los estudiantes) son procesados por un gestor de diálogo a fin de proporcionar una respuesta adecuada.

- **Proponer algoritmos que permitan apoyar a la metodología propuesta, permitiendo la edición, ejecución y depuración del pseudocódigo.**

- Enmarcado en la resolución de este objetivo, la Subsección 5.4.2 Intérprete del compañero de aprendizaje, muestra en la Figura 5.18 el esquema de ejecución del código fuente (AlcodyGEjec), proceso de ejecución del código fuente en Alcody.

La Subsección 5.4.4 Proceso de depuración, describe en la Figura 5.23 el esquema de funcionamiento de ejecución y depuración (AlcodyGProg), con el que cuenta el sistema interactivo educativo de Alcody.

- **Contribuir al área de la enseñanza de programación en Educación Primaria.**

Se ha comprobado cómo estudiantes de Educación Primaria pueden aprender a programar usando un entorno de programación con un compañero de aprendizaje como Alcody.

En un experimento realizado con 137 niños de entre 10-12 años que usaron Alcody durante 4 meses se registró una mejora significativa de sus notas en un pre-test y post-test de programación con 5 ejercicios involucrando conceptos como entrada/salida, condicionales y bucles (Ocaña et al., 2020 y Morales et al., 2021).

A este respecto, el Capítulo 5 detalla la validación de la metodología, y en la Figura 5.2 se muestra que Alcody consta de: Gestión de usuarios; contenidos de aprendizaje; gestor de diálogo; y editor de código.

- **Incrementar los niveles de aprendizaje, satisfacción y motivación en los niños en el momento de aprender a programar.**

Los estudiantes pudieron aumentar significativamente sus puntuaciones de aprendizaje, de una puntuación media del pre-test fue de 0,88 en septiembre de 2019, a una puntuación media de 7,56 en un primer post-test realizado en febrero de 2020 y alcanzando una puntuación media de 8,01 en el segundo post-test en junio de 2020.

De igual manera el uso del compañero de aprendizaje Alcody obtuvo nivel de satisfacción y motivación en los estudiantes superior al 90%, tanto en sesiones presenciales como en línea (Ocaña et al., 2020).

6.2 Contribuciones

El trabajo de investigación, muestra en la Figura 6.1 las principales contribuciones:

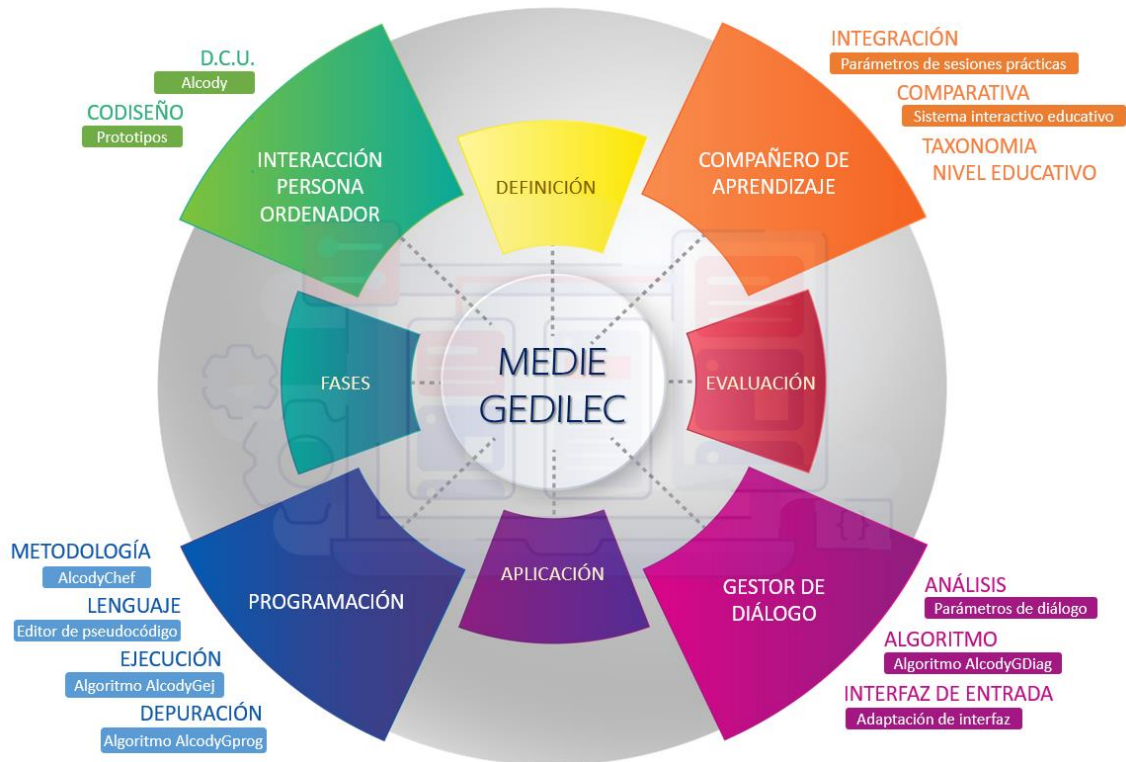


Figura 6.1 Contribuciones del trabajo de investigación (fuente: propia)

En la línea de investigación de Interacción Persona-Ordenador, se ha detectado una carencia en la existencia de metodologías para el diseño, desarrollo e integración de compañeros de aprendizaje para la enseñanza de programación en Educación Primaria.

En este sentido la principal contribución es el presentar una **propuesta metodológica denominada MEDIE_GEDILEC**, que ha sido validada mediante la implementación del compañero de aprendizaje Alcody que enseña programación a niños de Educación Primaria.

Siguiendo esta propuesta metodológica, un docente o investigador estará en condiciones de implementar un compañero de aprendizaje.

Se presentan características de los rasgos que aproximan la propuesta de diseño del compañero de aprendizaje, con el diseño centrado en las necesidades de los niños de Educación Primaria.

La Figura 5.3 muestra la visión de los docentes en relación al compañero de aprendizaje. La Figura 5.6 visualiza el resultado del co-diseño con niños del personaje Alcody.

Las principales contribuciones al área de compañeros de aprendizaje son:

La Tabla 3.5 donde se consideran cinco características relevantes, como parte de una estrategia de tutoría estructurada y que representan partes clave del proceso de aprendizaje (Crow et al., 2018). De esta forma, la Tabla 6.1 incluye al compañero de aprendizaje Alcody, e incluye un parámetro adicional para ver si se destina el aprendizaje en Educación Primaria.

COMPAÑEROS DE APRENDIZAJE	QU	CP	PL	LE	SO	EP
J-LATTE	-	✓	-	-	-	-
CPP-TUTOR	-	-	-	-	✓	-
ASK-ELLE	-	-	-	-	-	-
ANNETE	-	✓	✓	-	✓	-
JEPPY	-	-	✓	✓	✓	-
ALCODY	✓	-	✓	✓	✓	✓

Tabla 6.1 Características de compañeros de aprendizaje (fuente: adaptación Crow et al., 2018)

En la Figura 5.2 se muestra de manera global el sistema interactivo educativo con el compañero de aprendizaje, Alcody. Con los módulos (gestión de usuarios, contenidos de aprendizaje, gestor de diálogo y editor de código).

En relación a la integración, la Figura 4.23 describe la propuesta del proceso en las modalidades presencial y en línea de las sesiones prácticas.

Por otra parte, en cuanto al **Gestor de diálogo**, se cuenta con:

- La Tabla 5.6 describe las directrices para los parámetros a las preguntas planteadas en el análisis del diálogo entre el estudiante y Alcody.
- La Figura 5.10 muestra la interfaz de comunicación entre el estudiante y Alcody. Esta interfaz se adaptó de la experiencia del desarrollo de prototipos en papel de los niños relacionada con la forma de comunicarse con el ordenador.
- En la Figura 5.13 se observa el algoritmo del esquema de funcionalidad del gestor de diálogo (AlcodyGDiag) que gestiona el diálogo entre el estudiante y Alcody.

En relación a la **Programación**, se cuenta con:

- La Figura 5.37 muestra a AlcodyChef, el compañero de aprendizaje se viste como cocinero gamificando la metodología MECOPROG basada en el uso de metáforas de cocina para enseñar programación.
- La Figura 5.40 presenta el editor de código del sistema interactivo educativo Alcody.
- La Figura 5.18 muestra el proceso de ejecución del código fuente (AlcodyGEjec) que se basa en las fases de análisis léxico, sintáctico y semántico.
- La Figura 5.23 describe el esquema de funcionamiento del depurador (AlcodyGProg), con el que cuenta el sistema interactivo educativo de Alcody.

6.3 Trabajo futuro

Como primer punto del trabajo futuro será conseguir obtener un mayor tiempo de exposición del compañero de aprendizaje en escuelas de Educación Primaria para obtener más datos y continuar mejorando.

Otra línea de trabajo futuro es la realización de un estudio para investigar la relación entre la mejora de las puntuaciones de aprendizaje, gamificación y niveles de satisfacción de los alumnos, incluso cuando no tienen experiencia previa en Scratch o en programación.

Por otra parte se pretende incorporar la funcionalidad de puntuación automática para las preguntas de los cuestionarios tratados.

En el mismo sentido de aprender a programar usando un entorno de programación basado en texto, en el futuro se considera incorporar al sistema interactivo educativo, una versión fundamentada en Logo, para experimentar con niños de entre 8 a 10 años.

Se espera que MEDIE_GEDILEC pueda ser utilizada por más investigadores, diseñadores y desarrolladores que quieran desarrollar compañeros de aprendizaje para la enseñanza de programación en Educación Primaria.

Bibliografía

A

- Abelson, H., & DiSessa, A. A. (1986). *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press.
- Abelson, H., Dybvig, R. K., Haynes, C. T., Rozas, G. J., Adams, N. I., Friedman, D. P., & Wand, M. (1998). Revised 5 report on the algorit
- AbuShawar, B., & Atwell, E. (2015). ALICE chatbot: trials and outputs. *Computación y Sistemas*, 19(4), 625-632.
- Abu-Naser, S. S. (2009). Evaluating the effectiveness of the CPP-Tutor, an Intelligent Tutoring System for students learning to program in C++.
- Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of learning group publication*, 5(3), 438.
- Adey, P. (1999). *The Science of Thinking, and Science for Thinking: A Description of Cognitive Acceleration through Science Education (CASE)*. Innodata Monographs 2.
- Adom, D., Yeboah, A., & Ankrah, A. K. (2016). Constructivism philosophical paradigm: implication for research, teaching and learning. *Global Journal of Arts Humanities and Social Sciences*, 4(10), 1-9.
- Aguiar, E. V. B., Tarouco, L. M. R., & Reategui, E. (2014, January). Supporting problem-solving in mathematics with a conversational agent capable of representing gifted students' knowledge. In *2014 47th Hawaii International Conference on System Sciences* (pp. 130-137). IEEE.
- Albacete, P., Jordan, P., Lusetich, D., Chounta, I. A., Katz, S., & McLaren, B. M. (2018, June). Providing proactive scaffolding during tutorial dialogue using guidance from student model predictions. In *International Conference on Artificial Intelligence in Education* (pp. 20-25). Springer, Cham.
- Aljohani, M. (2017). Principles of “constructivism” in foreign language teaching. *Journal of Literature and Art Studies* 97-107.
- Andersen, L. B., Danholt, P., Halskov, K., Hansen, N. B., & Lauritsen, P. (2015). Participation as a matter of concern in participatory design. *CoDesign*, 11(3-4), 250-261.
- Arvin, F., Espinosa, J., Bird, B., West, A., Watson, S., & Lennox, B. (2019). Mona: an affordable open-source mobile robot for education and research. *Journal of Intelligent & Robotic Systems*, 94(3-4), 761-775.

- Altin, H., & Pedaste, M. (2013). Learning approaches to applying robotics in science education. *Journal of baltic science education*, 12(3), 365.
- Ausubel, D. P., Novak, J. D., & Hanesian, H. (1968). Educational psychology: A cognitive view.
- Ausubel, D., Hanesian, H., & Novak, J. (1983). *Psicología Educativa. Un punto de vista cognoscitivo*. Trillas 2ª. México.
- Ayuso, A. J. R., & Rioja, I. H. (2005). *Libro blanco de tecnologías del habla*. Universidad de Granada.

B

- Bacigalupo, D. A., Warburton, W. I., Draffan, E. A., Zhang, P., Gilbert, L., & Wills, G. B. (2010, July). A formative eAssessment co-design case study. *In Advanced Learning Technologies (ICALT), 2010 IEEE 10th International Conference on* (pp. 35-37). IEEE.
- Barnes, D. J. (2002). Teaching introductory Java through LEGO MINDSTORMS models. *ACM SIGCSE Bulletin*, 34(1), 147–151.
- Barshay, J. (2014). Getting boys-and girls-interested in computer coding. *Education*.
- Becker, B. A., Glanville, G., Iwashima, R., McDonnell, C., Goslin, K., & Mooney, C. (2016). Effective compiler error message enhancement for novice programming students. *Computer Science Education*, 26(2-3), 148-175.
- Beran, T. N., Ramirez-Serrano, A., Kuzyk, R., Fior, M., & Nugent, S. (2011). Understanding how children understand robots: Perceived animism in child–robot interaction. *International Journal of Human-Computer Studies*, 69(7-8), 539-550.
- Berger, P. L., Berger, P. L., & Luckmann, T. (1966). *The social construction of reality: A treatise in the sociology of knowledge*. Anchor.
- Berland, M., & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology*, 24(5), 628-647.
- Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education*, 3(2), 115-138.
- Berliner, D. C., & Calfee, R. C. (2013). *Handbook of educational psychology*. Routledge.
- Bickmore, T., & Cassell, J. (2001, March). Relational agents: a model and implementation of building user trust. *In Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 396-403).

- Bliss, J. (2002). The relevance of Piaget to research into children's conceptions. In *Children's informal ideas in science* (pp. 34-58). Routledge.
- Bloom, B. S. (1956). Taxonomy of educational objectives: The classification of educational goals. *Cognitive domain*.
- Bodner, G. M., Klobuchar, M., & Geelan, D. (2001). The Many Forms of Constructivism. *Journal of Chemical Education*, 78, 1107.
- Bozkurt, G. (2017). Social Constructivism: Does It Succeed in Reconciling Individual Cognition with Social Teaching and Learning Practices in Mathematics? *Journal of Education and Practice*, 8(3), 210-218.
- Brin, D. (2006). Why Johnny can't code. *Salon.com*.
- Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87(4), 834-860.
- Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (Vol. 1, p. 25).
- Bruner, J. S. (1966). *Toward a theory of instruction* (Vol. 59). Harvard University Press.
- Burr, Vivien (2003) *Social Constructionism*. London: Sage.
- Burr, V., & Dick, P. (2017). Social constructionism. In *The Palgrave handbook of critical social psychology* (pp. 59-80). Palgrave Macmillan, London.
- Busjahn, T., & Schulte, C. (2013). The use of code reading in teaching programming. In *Proceedings of the 13th Koli Calling international conference on computing education research* (pp. 3-11).

C

- Calao, L. A., Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with scratch. In *Design for teaching and learning in a networked world* (pp. 17-27). Springer, Cham.
- Calvo, I., Cabanes, I., Quesada, J., & Barambones, O. (2017). A multidisciplinary PBL approach for teaching industrial informatics and robotics in engineering. *IEEE Transactions on Education*, 61(1), 21-28.
- Carrier, D. (1974). Three kinds of imagination. *The Journal of Philosophy*, 70(22), 819-831.

- Cashman, T. J., Gunter, R. E., Gunter, G. A., Shelly, G. B., & Hall, M. P. EDU 2022/Grissom/Spring 2008.
- Chambliss, J. (1996), History of Philosophy of Education In J. Chambliss (ed.), *Philosophy of Education, An Encyclopedia*, Garland Publications, New York, pp. 461- 472.
- Chase, D. Chin, M. Oppedizzo, D. Schwartz. “Teachable agents and the Protégé effect: Increasing the effort towards learning”, *Journal of Science Educational Technology*, 18, 334-352, 2009.
- Chaudhary, V., Agrawal, V., Sureka, P., & Sureka, A. (2016). An experience report on teaching programming and computational thinking to elementary level children using lego robotics education kit. In *2016 IEEE Eighth International Conference on Technology for Education (T4E)* (pp. 38-41).
- Chen, N. S., Quadir, B., & Teng, D. C. (2011). A Novel approach of learning English with robot for elementary school students. In *International Conference on Technologies for E-Learning and Digital Entertainment* (pp. 309-316). Springer, Berlin, Heidelberg.
- Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers & education*, 59(2), 661-686.
- Crow, T., Luxton-Reilly, A., & Wuensche, B. (2018). Intelligent tutoring systems for programming education: a systematic review. In *Proceedings of the 20th Australasian Computing Education Conference* (pp. 53-62).
- Crespo, M., & Cabrera, B. D. (2020). Perspectivas de las tecnologías de Chatbot y su aplicación a las entrevistas de evaluación del lenguaje. *Pragmalingüística*, (2), 100-113.

D

- Dagdilelis, V., Sartatzemi, M., & Kagani, K. (2005). Teaching (with) robots in secondary schools: some new and not-so-new pedagogical problems. In *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)* (pp. 757-761).
- Damsa, A., & Damsa, D. (2014). Gamification implementation Model—An approach of linking psychology and game thinking. In *Workshop on Application of Robotics for Enhanced Security* (pp. 13-14).

- Danilava, S., Busemann, S., & Schommer, C. (2012). Artificial Conversational Companions: A Requirements Analysis. *Proceedings of 4th International Conference on Agents and Artificial Intelligence* (pp. 282-289).
- Dann, W. P., Cooper, S., & Pausch, R. (2006). *Learning to Program with Alice, Brief Edition*. Prentice-Hall, Inc.
- Day, M., Penumala, M. R., & Gonzalez-Sanchez, J. (2019). Annete: An Intelligent Tutoring Companion Embedded into the Eclipse IDE. In *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)* (pp. 71-80). IEEE.
- Denny, P., Luxton-Reilly, A., & Carpenter, D. (2014, June). Enhancing syntax error messages appears ineffectual. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 273-278).
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011, September). From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments* (pp. 9-15).
- Deterding, S., Sicart, M., Nacke, L., O'Hara, K., & Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. In *CHI'11 extended abstracts on human factors in computing systems* (pp. 2425-2428).
- Dewey, J. (1938) *Experience and education*, New York: MacMillan.
- Driver, R., & Easley, J. (1978). Pupils and paradigms: a review of literature related to concept development in adolescent science students. *Studies in Science Education*, 5, 61-84.
- Driver, R., & Erickson, G. (1983). Theories-in-action: some theoretical and empirical issues in the study of students' conceptual frameworks in science. *Studies in Science Education*, 10, 37-60.
- Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1), 57-73.

E

- Ernest, P. (1994) 'Social Constructivism and the Psychology of Mathematics Education', in Ernest, P. (Ed.) *Constructing Mathematical Knowledge: Epistemology and Mathematical Education*, London: The Falmer Press pp. 62-72.
- Escobar Ubreva, Á., León Vergara, M., Fernández Bermejo, M., & Buiza Camacho, B. (2010). Manual de acreditación de páginas web sanitarias, una apuesta por la calidad y la accesibilidad. *No Solo Usabilidad*, (9).

- Evangelopoulou, O., & Xinogalos, S. (2018). MYTH TROUBLES: An open-source educational game in scratch for Greek Mythology. *Simulation & Gaming*, 49(1), 71-91.

F

- Fagin, B., & Merkle, L. (2003). Measuring the effectiveness of robots in teaching computer science. *Acm sigcse bulletin*, 35(1), 307-311.
- Falkner, K., Vivian, R., & Falkner, N. J. (2014, June). Identifying computer science self-regulated learning strategies. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 291-296).
- Feijoo Garcia, P. G., & De la Rosa, F. (2016). RoBlock–web app for programming learning.
- Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63, 87-97.
- Fensham, P. J. (2004). *Defining an identity: The evolution of science education as a field of research* (Vol. 20). Springer Science & Business Media.
- Ferguson, C. J., & Garza, A. (2011). Call of (civic) duty: Action games and civic behavior in a large sample of youth. *Computers in Human Behavior*, 27(2), 770-775.
- Finch, D. K., & Edwards, S. H. (2020). *Using a Pedagogical Agent to Support Students Learning to Program*. American Society for Engineering Education.
- Fino, C. N. (2017). Constructionism and the shifting from didactics to mathematics. *International Journal of Development Research*, 7(10), 16250-16255.
- Fitzgerald, T. (2013). GUInform: Interactive Fiction for GUI Prototyping.
- Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013, June). Designing ScratchJr: Support for early childhood learning through computer programming. In *proceedings of the 12th international conference on interaction design and children* (pp. 1-10).
- Fletcher, G. H., & Lu, J. J. (2009). Education Human computing skills: rethinking the K-12 experience. *Communications of the ACM*, 52(2), 23-25.
- Ford Jr, J. (2009). *Getting Started with Game Maker*. Nelson Education.
- Frawley, W. (1997). Vygotsky and Cognitive Science: Language and the Unification of the Social and Computational Mind. *Harvard University Press, 79 Garden Street, Cambridge, MA 02138-1423*.

- Freeman, A., Becker, S. A., & Cummins, M. (2017). NMC/CoSN horizon report: 2017 K. The New Media Consortium.
- Fryer, L. K., Ainley, M., Thompson, A., Gibson, A., & Sherlock, Z. (2017). Stimulating and sustaining interest in a language course: An experimental comparison of Chatbot and Human task partners. *Computers in Human Behavior*, 75, 461-468.

G

- Gerdes, A., B. Heeren, J. Jeuring, and L. T. van Binsbergen (2016). Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback. *IJAIED*, (pp 1–36).
- Ghose, S., & Barua, J. J. (2013). Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor. *In 2013 international conference on informatics, electronics and vision (ICIEV)* (pp. 1-5).
- Gergen, M. M., & Davis, S. N. (Eds.). (2013). Toward a new psychology of gender: A reader. *Routledge*.
- Glasersfeld, E. (1989). Constructivism in education. *The international encyclopedia of education-research and studies*, 162-163.
- Glasersfeld, E. (1990). Environment and communication. *Transforming children's mathematics education: International perspectives*, 30-38.
- Gilbert, J. K., & Watts, D. M. (1983). Concepts, misconceptions and alternative conceptions: changing perspectives in science education. *Studies in Science Education*, 10(1), 61-98.
- Gilbert, J. K., Osborne, R. J., & Fensham, P. J. (1982). Children's science and its consequences for teaching. *Science Education*, 66(4), 623-633.
- Gilbert, J. K. (1995). Studies and Fields: Directions of Research in Science Education. *Studies in Science Education*, 25, 173 - 197.
- Griol, D., & Callejas, Z. (2013). An architecture to develop multimodal educative applications with chatbots. *International Journal of Advanced Robotic Systems*, 10(3), 175.
- Goldberg, A., & Kay, A. (1976). *Smalltalk-72: Instruction Manual*. Palo Alto: Xerox Corporation.
- Granollers, T., Lorés, J., & Perdrix, F. (2002). Modelo de proceso de la Ingeniería de la Usabilidad. Integración de la ingeniería del Software y la de la Usabilidad. *Proc. Work. Investig. sobre nuevos Paradig. interacción en entornos Colab. Apl. a la gestión y difusión del Patrim.* 2, 11-12.

- Grant, M. M. (2011). Learning, beliefs, and products: Students' perspectives with project-based learning. *Interdisciplinary Journal of Problem-Based Learning*, 5(2), 6.
- Graesser, A.; D'Mello, S.; Craig, S.; Witherspoon, A.; Sullins, J.; McDaniel, B.; Gholson, B. The relationship between affective states and dialog patterns during interactions with AutoTutor. *J. Interact. Learn. Res.* 2008, 19, 293–302.
- Graesser, A. C., Cai, Z., Morgan, B., & Wang, L. (2017). Assessment with computer agents that engage in conversational dialogues and trialogues with learners. *Computers in Human Behavior*, 76, 607-616.
- Griffiths, C. (2004). Language-learning strategies: Theory and research. AIS St Helens, *Centre for Research in International Education*.
- Greg Nelson L., Xie, B., & Ko, A. J. (2017, August). Comprehension first: evaluating a novel pedagogy and tutoring system for program tracing in CS1. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 2-11).
- Green, C. S., & Bavelier, D. (2012). Learning, attentional control, and action video games. *Current biology*, 22(6), R197-R206.
- Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1), 38-43.
- Gärdenfors, P. and Johansson, P. (2005). 'Introduction to Cognition, Education and Communication Technology' in Gärdenfors, P. and Johansson, P. (Eds.) *Cognition, Education, and Communication Technology*, New Jersey: Lawrence Erlbaum Associates, pp 1-20.

H

- Haake, M. (2009). "Embodied Pedagogical Agents: From Visual Impact to Pedagogical Implications." Institutionen för designvetenskaper, Lunds Tekniska Högskola, Lunds universite. Lund University. 978-91-628-7804-7.
- Han, S., & Bhattacharya, K. (2001). Constructionism, learning by design, and project based learning. *Emerging perspectives on learning, teaching, and technology*. Retrieved April, 29, 2007.
- Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.
- Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. In *Frontiers in Education Conference (FIE)*, 2016 IEEE (pp. 1-9).
- Honebein, P. C. (1996). Seven goals for the design of constructivist learning environments. *Constructivist learning environments: Case studies in instructional design*, 11-24. ISO 690.

- Hein, G. E. (1991). Constructivist learning theory. *Institute for Inquiry*.
- Hoffmann, L., & Krämer, N. C. (2013). Investigating the effects of physical and virtual embodiment in task-oriented and conversational contexts. *International Journal of Human-Computer Studies*, 71(7-8), 763-774.
- Holland, J., Mitrovic, A., & Martin, B. (2009). J-LATTE: a Constraint-based Tutor for Java.
- Holden, W. R. (2004). Facilitating listening comprehension: Acquiring successful strategies. *Bulletin of Hokuriku University*, 28(3), 257-266.
- Holzner, B. (1972). Reality construction in society.
- Hoban, G., Nielsen, W., & Carceller, C. (2010). Articulating constructionism: Learning science through designing and making" Slowmations"(student-generated animations).
- Hong, J. C., Yu, K. C., & Chen, M. Y. (2011). Collaborative learning in technological project design. *International Journal of Technology and Design Education*, 21(3), 335-347.

J

- Jaques, P. A., & Vicari, R. (2007). A BDI Approach to Infer Student's Emotions in an Intelligent Learning Environment. *Computers and Education*, 49, 360-384.
- Jaworski, B. (1996). Investigating mathematics teaching: A constructivist enquiry (No. 5). *Taylor & Francis*.
- Jenkins, T. (2002, August). On the difficulty of learning to program. *In Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (Vol. 4, No. 2002, pp. 53-58).
- Jha, A. K. (2012). Epistemological and pedagogical concerns of constructionism: Relating to the educational practices. *Creative Education*, 3(02), 171.
- Jojoa, E. M. J., Bravo, E. C., & Cortés, E. B. B. (2009). Tool for experimenting with concepts of mobile robotics as applied to children's education. *IEEE Transactions on Education*, 53(1), 88-95.
- Jones, M. G., & Brader-Araje, L. (2002). The impact of constructivism on education: Language, discourse, and meaning. *American Communication Journal*, 5(3), 1-10.
- Johnson, L., Becker, S. A., Estrada, V., & Freeman, A. (2014). NMC horizon report: 2014 K (pp. 1-52). The New Media Consortium.

- Jost, J. T., & Kruglanski, A. W. (2002). The estrangement of social constructionism and experimental social psychology: History of the rift and prospects for reconciliation. *Personality and Social Psychology Review*, 6, 168–187.

K

- Kahn Jr, P. H., Kanda, T., Ishiguro, H., Freier, N. G., Severson, R. L., Gill, B. T., Ruckert, J. H., & Shen, S. (2012). “Robovie, you’ll have to go into the closet now”: Children’s social and moral relationships with a humanoid robot. *Developmental Psychology*, 48(2), 303-314.
- Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons.
- Kay A. C. (1993). The Early History of Smalltalk. *SIGPLAN Not*, 28 (3), 69–95.
- Kay A., Rose K., Ingalls D., Kaehle T., Maloney J. & Wallace S. (1997) “Etoys & SimStories.” *Walt Disney Imagineering*.
- Kafai, Y. (2006). Constructionism, R. K. Sawyer (Edited), *The Cambridge handbook of the learning sciences*, Cambridge University Press, 35-46.
- Kelley, J.F. (1984). An iterative design methodology for user-friendly natural language information applications, *ACM Transactions on Information Systems*, 2, 26-41.
- Kelly, G., Kelly, G. A., & Kelly, G. A. (1963). A theory of personality: The psychology of personal constructs (No. 152).
- Kim, Y. (2016). The role of agent age and gender for middle-grade girls. *Computers in the Schools*, 33(2), 59-70.
- Kim, Y., & Baylor, A. L. (2006). A social-cognitive framework for pedagogical agents as learning companions. *Educational technology research and development*, 54(6), 569-596.
- Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C. N. (2015). Robotics to promote elementary education pre-service teachers' STEM engagement, learning, and teaching. *Computers & Education*, 91, 14-31.
- Kozima, H., & Nakagawa, C. (2007). A Robot in a playroom with preschool children: Longitudinal field practice. In *Proceedings of 16th IEEE International Conference Robot & Human Interactive Communication (ROMAN 2007)* (pp. 1058-1059).
- Kumar, M. (2012) cognitive constructivism in inclusive education. *Journal of Global Research & Analysis*, 4, 30.

- Kumar, J. (2013). Gamification at work: Designing engaging business software. In International conference of design, user experience, and usability (pp. 528-537). Springer, Berlin, Heidelberg.
- Kucuk, S., & Sisman, B. (2017). Behavioral patterns of elementary students and teachers in one-to-one robotics instruction. *Computers & Education*, 111, 31-43.
- Kuniavsky, M. (2003). *Observing the user experience: a practitioner's guide to user research*. Elsevier.
- Kurtz, Thomas E. (1981). "BASIC". In Wexelblat, Richard (ed.). *History of programming languages*. History of Programming Languages I. New York: ACM. pp. 515–537.
- Ko, A. J., Myers, B. A., & Aung, H. H. (2004, September). Six learning barriers in end-user programming systems. In 2004 *IEEE Symposium on Visual Languages-Human Centric Computing* (pp. 199-206). *IEEE*.
- Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., & Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, 43(3), 1-44.

L

- Landauer, T.; Foltz, P.; Laham, D. An introduction to Latent Semantic Analysis. *Discourse Process*. 1998, 25, 259–284.
- Luerssen, M., & Powers, D. M. W. (2011) Chapter Number Embodied Conversational Agents for Education in Autism.
- Leonhardt, M., Dutra, R. L. S. D., Granville, L. Z., & Tarouco, L. M. R. (2005). DOROTY: An Extension in the Architecture of a ChatterBot for Academic and Professional Training in the Field of Network Management. *IFIP World Conference on Computers in Education*. Cape Town.
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J. & Werner, L. (2011). *Computational thinking for youth in practice*. *Acm Inroads*, 2(1), 32-37
- Lester, S. Converse, S. Kahler, S. Barlow, B. Stone, R. Bhogal. "The Persona effect: affective impact of animated pedagogical agents", *Presented at the SIGCHI conference on Human factors in computing systems*, 1997.
- Liang, H., Chang, J., Kazmi, I. K., Zhang, J. J., & Jiao, P. (2017). Hand gesture-based interactive puppetry system to assist storytelling for children. *The Visual Computer*, 33(4), 517-531.

- Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., & Simon, B. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4), 119-150.
- Lister, R., Fidge, C., & Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *ACM SIGCSE Bulletin*, 41(3), 161-165.
- Lodi, M., Malchiodi, D., Monga, M., Morpurgo, A., & Spieler, B. (2018, August). Learning to program in a constructionist way. In *Constructionism 2018: Constructionism, Computational Thinking and Educational Innovation* (pp. 888-911).
- Loksa, D., & Ko, A. J. (2016). The role of self-regulation in programming problem solving process and success. In *Proceedings of the 2016 ACM conference on international computing education research* (pp. 83-91).
- Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016). *Programming, problem solving, and self-awareness: effects of explicit guidance*. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 1449-1461).
- Lorés, J., Granollers, T., & Lana, S. (2002). Introducción a la interacción persona-ordenador. *Universidad de Lleida*.

M

- Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2016). CMX: The effects of an educational MMORPG on learning and teaching computer programming. *IEEE Transactions on Learning Technologies*, 10(2), 219-235.
- MacMillan, D. (2011). Gamification': A growing business to invigorate stale websites. *Bloomberg Businessweek*. Retrieved December, 7, 2015.
- Manzini, E. (2015). *Cuando todos diseñan: Una introducción al diseño para la innovación social*. Madrid: Gráficas Muriel.
- Marache-Francisco, C., & Brangier, E. (2013). Process of gamification. *Proceedings of the 6th Centric*, 126-131.
- Martinez, I. M. (1996). The importance of language learning strategies in foreign language teaching. *Cuadernos of English Philology*, 5(1), 103-120.
- Mathieu, M. J. (2014). Introducción a la programación. *Grupo Editorial Patria*.
- Matthews, M. R. (Ed.). (1998). Constructivism in science education: A philosophical examination. *Springer Science & Business Media*.

- Maturana, H. R. (1978). Biology of Language: The Epistemology of Reality in G. Miller & E. Lenneberg (Eds.) *Psychology and Biology of Language and Thought*.
- McGonigal, J. (2011). *Reality is broken: Why games make us better and how they can change the world*. Penguin.
- Mckerrell, S. (2016). Social constructionism in music studies. *Popular Music*, 35(3), 425.
- McGonigal, J. (2011). *Reality is broken: Why games make us better and how they can change the world*. Penguin.
- Meang, U.K (2006). Comparison of L2 listening and reading comprehension strategies: A case study of three middle school students. *The Journal of Curriculum & Evaluation*, 9(2), 471-500.
- Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, 23(3), 239-264.
- Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R. (eds.): *Robotics in Education: Research and Practices for Robotics STEM Education*. Springer, Berlin (2017)
- Mergendoller, J. R., Maxwell, N. L., & Bellisimo, Y. (2006). The effectiveness of problem-based instruction: A comparative study of instructional methods and student characteristics. *Interdisciplinary Journal of Problem-based Learning*, 1(2), 49-69.
- Mills, B., Evens, M., & Freedman, R. (2004, April). Implementing directed lines of reasoning in an intelligent tutoring system using the atlas planning environment. *In International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004*. (Vol. 1, pp. 729-733). IEEE.
- Moggridge, B., & Atkinson, B. (2007). *Designing interactions* (Vol. 17). Cambridge, MA: MIT press.
- Morales-Rodríguez, M., & Domínguez-Martínez, J. (2011). Agentes conversacionales como un sistema de diálogo. *Memorias del V Encuentro de Investigadores del ITCM*.
- Morales-Urrutia, E. K., Ocaña, J. M., Pérez-Marín, D., & Tamayo, S. (2017, Octubre). A first proposal of Pedagogic Conversational Agents to develop Computational Thinking in children. *In Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality ACM*, 2, 1–6.

- Morales-Urrutia, E., Ocaña, J. M. C., Aguirre, J. L. S., & Pérez-Marín, D. (2019). Interfaz de usuario enfocado en el co-diseño con niños. *Revista Ibérica de Sistemas e Tecnologías de Informação*, (E22), 147-160.
- Morales-Urrutia, E. K., Ocaña, J. M., & Pérez-Marín, D. (2020). How to Integrate Emotions in Dialogues With Pedagogic Conversational Agents to Teach Programming to Children. In *Innovative Perspectives on Interactive Communication Systems and Technologies* (pp. 66-91). IGI Global.
- Morales-Urrutia, E. K., Ch, Ocaña, J. M., Pérez-Marín, D., & Pizarro-Romero, C. (2020, July). Promoting learning and satisfaction of children when interacting with an emotional companion to program. In *International Conference on Artificial Intelligence in Education* (pp. 220-223). Springer, Cham.
- Morales-Urrutia, E. K., Ocaña, J. M., Pérez-Marín, D., & Pizarro, C. (2020). ¿Pueden los niños aprender a programar usando un entorno de programación basado en texto con un agente compañero? *Informática Educativa Comunicaciones*, 32.
- Morales-Urrutia, E. K., Ocaña, J. M., Pérez-Marín, D., & Pizarro, C. (2021). Can mindfulness help Primary Education students to learn how to program with an emotional learning companion? *IEEE Access*, 9, 6642-6660.
- Montessori, M. (1952). Kinder sind anders (Original: Il segreto dell'infanzia).
- Moreno-León, J., & Robles, G. (2016, April). Code to learn with Scratch? A systematic literature review. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 150-156).
- Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: the good, the bad, and the quirky--a qualitative analysis of novices' strategies. *ACM SIGCSE Bulletin*, 40(1), 163-167.
- Mustafa, D., & Kalender, S. (2007). Applying the Subject" Cell" through Constructivist Approach during Science Lessons and the Teacher's View. Online Submission, 2(1), 3-13.

N

- Németh, T. (2015). English knight: Gamifying the EFL classroom. *Unpublished Master's Thesis*, Pázmány Péter Katolikus Egyetem Bölcsészeti és Társadalomtudományi Kar, Piliscsaba, Hungary. Available online: <https://ludus.hu/gamification/> (accessed on 28 January 2019).
- Nica, E., & Potcovaru, A. M. (2015). The social sustainability of the sharing economy. *Economics, Management and Financial Markets*, 10(4), 69.

- Fino, C. N. (2017). Constructionism and the shifting from didactics to mathematics. *International Journal of Development Research*, 7(10), 16250-16255.
- Norman, D. (2002). Emotion & design: attractive things work better. *Interactions*, 9(4), 36-42.
- Noss, R., & Hoyles, C. (2017). Constructionism and microworlds. In *Technology enhanced learning* (pp. 29-35). Springer, Cham.
- Nunes, M. A., Dihl, L. L., Fraga, L. M., Woszezenki, C. R., Oliveira, L., Francisco, D. J. da Glória Notargiacomo, M. (2002). Animated pedagogical agent in the intelligent virtual teaching environment. *Digital Education Review*, (4), 53-61.

O

- Olson, C. B., & Land, R. (2007). A cognitive strategies approach and reading and writing instructions for English language learners in secondary school. *Research in the Teaching English*, 41(3), 269-303.
- Osborne, R. J., & Wittrock, M. C. (1985). The generative learning model and its implications for science education. *Studies in Science Education*, 12, 59-87.
- O'Connor, M. C. (1998) Can We Trace the "Efficacy of Social Constructivism?" *Review of Research in Education*, Vol.23, pp 25-71.
- Ocaña, J. M., Morales-Urrutia, E., Pérez-Marín, D., & Tamayo, S. (2019). Gestión del diálogo de un Agente Conversacional Pedagógico para aprender a programar. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E19), 239-251.
- Ocaña, J. M., Morales-Urrutia, E. K., Pérez-Marín, D., & Tamayo-Moreno, S. (2019). How to Create a Pedagogic Conversational Agent for Teaching Computer Science. *In Advanced Online Education and Training Technologies* (pp. 114-134). IGI Global.
- Ocaña, J. M., Morales-Urrutia, E. K., Pérez-Marín, D., & Pizarro, C. (2020). Can a learning companion be used to continue teaching programming to children even during the COVID-19 pandemic? *IEEE Access*, 8, 157840-157861.

P

- Papadakis, S., & Orfanakis, V. (2016). The combined use of Lego Mindstorms NXT and App Inventor for teaching novice programmers. In *International Conference EduRobotics 2016* (pp. 193-204). Springer, Cham.
- Papadakis, S., & Orfanakis, V. (2018). Comparing novice programming environments for use in secondary education: App Inventor for Android vs. Alice. *International Journal of Technology Enhanced Learning*, 10(1-2), 44-72.

- Palincsar, A. S. (1998). Social constructivist perspectives on teaching and learning. *Annual Review of Psychology*, 49, 345–375.
- Parke, R. D., & Clarke-Stewart, K. A. (2003). The effects of parental incarceration on children: Perspectives, promises, and policies. *Prisoners once removed: The impact of incarceration and reentry on children, families, and communities*, 189-232.
- Passos, E. B., Medeiros, D. B., Neto, P. A., & Clua, E. W. (2011, November). Turning real-world software development into a game. *In 2011 Brazilian Symposium on Games and Digital Entertainment* (pp. 260-269). *IEEE*.
- Pelling, N. (2011). The (short) prehistory of “gamification”.... Funding Startups (& other impossibilities). *Journal of Nano Dome*.
- Pérez-Marín, D., Hijón-Neira, R., & Martín-Lope, M. (2017). Propuesta de Metodología basada en Metáforas para la Enseñanza de la Programación a Niños. *Versión Abierta Español–Portugués*, 32.
- Pérez-Marín, D., Hijón-Neira, R., Romero, A., & Cruz, S. (2019). Is the use of Makey Makey Helpful to Teach Programming Concepts to Primary Education Students? *International Journal of Online Pedagogy and Course Design (IJOPCD)*, 9(2), 63-77.
- Perkins, D. N., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. *In at Empirical Studies of Programmers, 1st Workshop, Washington, DC* (pp. 213-229).
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2), 1-11.
- Papert, S. (1980). *Mindstorms: Children. Computers and powerful ideas*. Basic Books.
- Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & education*, 52(1), 1-12.
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2), 1-11.
- Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. BasicBooks, 10 East 53rd St., New York, NY 10022-5299.
- Perez, J. E., Dinawanao, D. D., & Tabanao, E. S. JEPPY: An Interactive Pedagogical Agent to Aid Novice Programmers in Correcting Syntax Errors.
- Pezzullo, L. G., Wiggins, J. B., Frankosky, M. H., Min, W., Boyer, K. E., Mott, B. W... & Lester, J. C. (2017, June). “Thanks Alisha, Keep in Touch”: Gender Effects

and Engagement with Virtual Learning Companions. *In International Conference on Artificial Intelligence in Education* (pp. 299-310). Springer, Cham.

- Picard, R. W., Papert, S., Bender, W., Blumberg, B., Breazeal, C., Cavallo, D., & Strohecker, C. (2004). Affective learning—a manifesto. *BT Technology Journal*, 22(4), 253-269.
- Phillips, D. C. (1995). The good, the bad, and the ugly: The many faces of constructivism. *Educational researcher*, 24(7), 5-12.
- Piaget, J. (1970) *Structuralism*, New York: Basic Books.
- Piaget, J. (1973). The affective unconscious and the cognitive unconscious. *Journal of the American Psychoanalytic Association*, 21(2), 249-261.
- Plaza, P., Sancristobal, E., Carro, G., Blazquez, M., García-Loro, F., Muñoz, M. & Castro, M. (2019). STEM and educational robotics using scratch. In 2019 *IEEE Global Engineering Education Conference (EDUCON)* (pp. 330-336).
- Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction Design: Beyond Human-Computer Interaction*, New York: John Wiley & Sons.
- Prensky, M. (2003). Digital game-based learning Computers in Entertainment (CIE). *Listen to the natives Educational Leadership*.

Q

- Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1), 1-24.

R

- Redecker, C. (2017). *European framework for the digital competence of educators: DigCompEdu* (No. JRC107466). Joint Research Centre (Seville site).
- Resnick, M. (2009). Making Programming Universally Accessible and Useful. *Proposal to*.
- Resnick, M. (2012, August). Mother's day, warrior cats, and digital fluency: Stories from the scratch online community. In *Proceedings of the constructionism 2012 conference: Theory, practice and impact* (pp. 52-58).
- Rickel, J., & Johnson, W. L. (1999, July). Virtual humans for team training in virtual reality. In *Proceedings of the ninth international conference on artificial intelligence in education* (Vol. 578, p. 585).

- Rigóczki, Damsa, Andrei and Györgyi-Ambró, Kristóf (2017) *Gamification on the edge of educational sciences and pedagogical methodologies*. *Journal of Applied Technical and Educational Sciences*, 7 (4). pp. 79-88
- Rigóczki, C. (2018). Constructivist Environmental Education in Urban Walking Lane. *Journal of Applied Technical and Educational Sciences*, 8(3), 68-82.
- Rogoff, B. (1999). Cognitive development through social interaction: Vygotsky and Piaget. *Learners, learning and assessment*, 69-82.
- Rob, M., & Rob, F. (2018). Dilemma between constructivism and constructionism. *Journal of International Education in Business*.
- Ryokai, K., Vaucelle, C., & Cassell, J. (2003). Virtual peers as partners in storytelling and literacy learning. *Journal of computer assisted learning*, 19(2), 195-208.

S

- Sáez-López, J. M., Sevillano-García, M. L., & de los Ángeles Pascual-Sevillano, M. (2019). Aplicación del juego ubicuo con realidad aumentada en Educación Primaria. *Comunicar*, 27(61), 71-82.
- Sanders, E. B. N. (1992). Converging perspectives: product development research for the 1990s. *Design management journal*, 3(4), 49-54.
- Sanders, E. B. N., & Stappers, P. J. (2008). Co-creation and the new landscapes of design. *Co-design*, 4(1), 5-18.
- Schulte, C. (2008, September). Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching. In *Proceedings of the Fourth international Workshop on Computing Education Research* (pp. 149-160).
- Schulte, C., Clear, T., Taherkhani, A., Busjahn, T., & Paterson, J. H. (2010). An introduction to program comprehension for computer science educators. In *Proceedings of the 2010 ITiCSE working group reports* (pp. 65-86).
- Schroeder, N. L., Adesope, O. O., & Gilbert, R. B. (2013). How effective are pedagogical agents for learning? A meta-analytic review. *Journal of Educational Computing Research*, 49(1), 1-39.
- Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, 29(2-3), 136-176.
- Segre, S. (2016). Social constructionism as a sociological approach. *Human studies*, 39(1), 93-99.
- Solomon, J. (1994). The rise and fall of constructivism. *Studies in Science Education*, 23, 1-19.

- Spohrer, J. C., & Soloway, E. (1986). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM*, 29(7), 624-632.
- Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(4), 1-64.

T

- Taber, K. S. (2016). Constructivism in education: Interpretations and criticisms from science education. In *Handbook of research on applied learning theory and design in modern education* IGI Global. (pp. 116-144).
- Tamayo Moreno, S. (2017). Propuesta de metodología para el diseño e integración en el aula de un agente conversacional pedagógico desde educación secundaria hasta educación infantil.
- Tamayo-Moreno, S., & Pérez-Marín, D. (2017). Designing and evaluating pedagogic conversational agents to teach children. *International Journal of Educational and Pedagogical Sciences*, 11(3), 521-526.
- Tamayo-Moreno, S., & Pérez-Marín, D. (2014). Diseño y gestión de diálogos educativos en agentes pedagógicos conversacionales para uso escolar. *Aplicaciones multidisciplinares de sistemas de diálogo*.
- Teague, D., & Lister, R. (2014, June). Programming: reading, writing and reversing. *In Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 285-290).
- Tegos, S., Demetriadis, S., & Karakostas, A. (2015). Promoting academically productive talk with conversational agent interventions in collaborative learning settings. *Computers & Education*, 87, 309-325.
- Tegos, S., & Demetriadis, S. (2017). Conversational agents improve peer learning through building on prior knowledge. *Journal of Educational Technology & Society*, 20(1), 99-111.
- Tsur, M. (2017). *Scratch Microworlds: introducing novices to scratch using an interest-based, open-ended, scaffolded experience* (Doctoral dissertation, Massachusetts Institute of Technology).
- Toh, L. P. E., Causo, A., Tzuo, P. W., Chen, I. M., & Yeo, S. H. (2016). A review on the use of robots in education and young children. *Journal of Educational Technology & Society*, 19(2), 148-163.

- Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education, 120*, 64-74.

V

- Van Merriënboer, J. J., & Krammer, H. P. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. *Instructional Science, 16*(3), 251-285.
- Varney, M. W., Janoudi, A., Aslam, D. M., & Graham, D. (2011). Building young engineers: TASEM for third graders in Woodcreek Magnet Elementary School. *IEEE transactions on education, 55*(1), 78-82.
- Vasek, M. (2012). Representing expressive types in blocks programming languages.
- Ventura, M., Shute, V., & Zhao, W. (2013). The relationship between video game use and a performance-based measure of persistence. *Computers & Education, 60*(1), 52-58.
- Vygotsky, (1934). *For a discussion of Vygotsky's argument for his perspective that learning leads development.* pp. 194–212 in Vygotsky (1934/1987b).
- Vygotsky, L. (1978). *Mind in society: The development of higher psychological processes.* Cambridge, MA: Harvard University Press.
- Vygotsky, L. S. (1979). *The development of higher forms of attention in childhood.* *Soviet Psychology, 18*(1), 67-115.
- Vygotsky, L. S. (1999). Tool and sign in the development of the child. *The collected works of LS Vygotsky, 6*, 3-68.
- Voss, J. F., Wiley, J., & Carretero, M. (1995). Acquiring intellectual skills. *Annual Review of Psychology, 46*(1), 155-181.

W

- Wallace, R. (1995). Artificial linguistic internet computer entity (alice). *City.*
- Wei, C. W., & Hung, I. (2011). A joyful classroom learning system with robot learning companion for children to learn mathematics multiplication. *Turkish Online Journal of Educational Technology-TOJET, 10*(2), 11-23.
- Weizenbaum, J. (1983). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM, 26*(1), 23-28.

- Wertsch, J. V. (1985). *Vygotsky and the social formation of mind*. Harvard university press.
- Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education (TOCE)*, 18(1), 1-20.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
- Werbach, K., & Hunter, D. (2012). *For the win: How game thinking can revolutionize your business*. Wharton digital press.
- Weng-yi Cheng, R., Shui-fong, L., & Chung-yan Chan, J. (2008). When high achievers and low achievers work in the same group: The roles of group heterogeneity and processes in project-based learning. *British Journal of Educational Psychology*, 78, 205–221.
- Wilson, A., & Moffat, D. C. (2010, September). Evaluating Scratch to Introduce Younger Schoolchildren to Programming. *In PPIG* (p. 7).

X

- Xia, L., & Zhong, B. (2018). A systematic review on teaching and learning robotics content knowledge in K-12. *Computers & Education*, 127, 267-282.
- Xie, T., & Luo, L. (2017). Impact of prompting agents on task completion in the virtual world. *International Journal of Online and Biomedical Engineering (iJOE)*, 13(06), 35-48.

Y

- Yeves-Martínez, P., & Pérez-Marín, D. (2019). Prof. Watson: A Pedagogic Conversational Agent to Teach Programming in Primary Education. *In Multidisciplinary Digital Publishing Institute Proceedings* (Vol. 31, No. 1, p. 84).
- Yee, N., & Bailenson, J. (2007). The Proteus effect: The effect of transformed self-representation on behavior. *Human communication research*, 33(3), 271-290.
- Yu-Kai Chou (2016). Actionable Gamification: beyond points, badges and leaderboards. Octalysis Media: Fremont. CA. *Revista Internacional de Organizaciones= International Journal of Organizations*, (18), 137-144.

Z

- Zhou, L. (2007), Natural language interfaz for information management on mobile devices. *Behavior & Information Technology*, 26, 197-207.

- Zichermann, G., & Cunningham, C. (2011). *Gamification by design: Implementing game mechanics in Web and mobile apps*. “O'Reilly Media, Inc.”
- Zille, H., Kottenhahn, A., & Mostaghim, S. (2017). Dynamic Distance Minimization Problems for dynamic multi-objective optimization. *In 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings* (pp. 952–959).

Referencias Web

HTTP1: Editor de código Ideone. Última fecha de consulta: 28 de mayo de 2021.

<https://www.ideone.com/>

HTTP2: Editor de código CodePad. Última fecha de consulta: 28 de mayo de 2021.

<http://www.codepad.org/>

HTTP3: Editor de código Python. Última fecha de consulta: 28 de mayo de 2021.

<https://www.python.org/>

HTTP4: Editor de código WritePhpOnline. Última fecha de consulta: 28 de mayo de 2021.

<https://www.writephponline.com/>

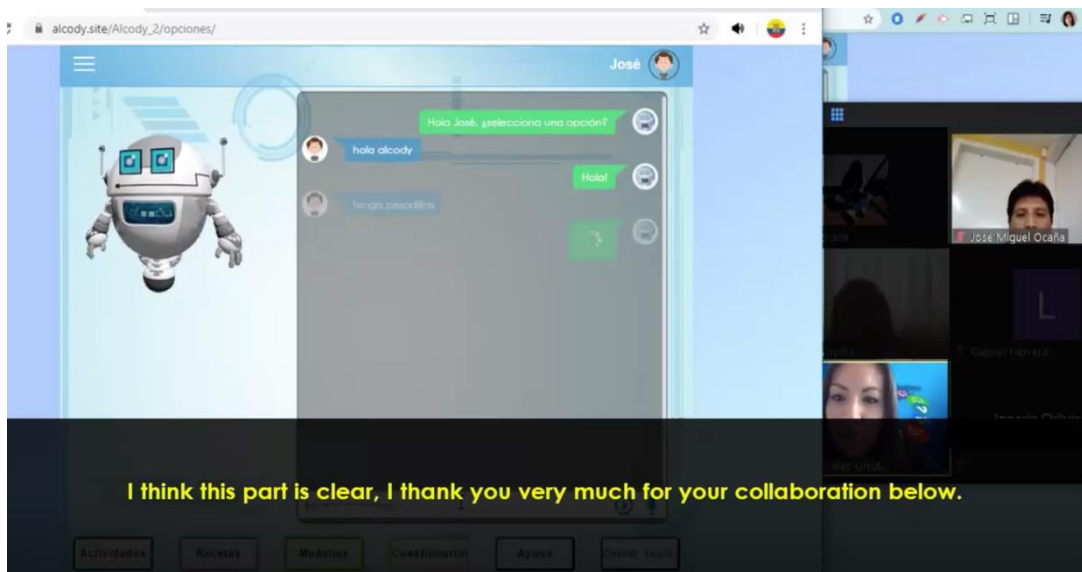
HTTP5: Editor de código Browxy. Última fecha de consulta: 28 de mayo de 2021.

<https://www.browxy.com/>

Anexo

MATERIAL AUDIOVISUAL

LINK VIDEO 1: <https://ieeexplore.ieee.org/abstract/document/9314156>



LINK VIDEO 2: <https://ieeexplore.ieee.org/abstract/document/9178762>

