

UNIVERSIDAD TÉCNICA DE MACHALA
FACULTAD DE INGENIERÍA CIVIL
MAESTRÍA EN SOFTWARE

**METODOLOGÍA DE SELECCIÓN DE LENGUAJES DE
PROGRAMACIÓN PARA DESARROLLAR SOFTWARE EN UN
AMBIENTE DE ENSEÑANZA-APRENDIZAJE**

ING. JUAN PABLO ARAUJO CORTEZ

**“PROPUESTA METODOLÓGICA Y TECNOLÓGICA AVANZADA EN
OPCIÓN AL TÍTULO DE MAGISTER EN SOFTWARE”**

TUTORA: ING. JENNIFER CÉLLERI. MSC

COTUTOR: ING. JORGE MAZA. MSC

MACHALA 2021

DEDICATORIA

En primer lugar, a mi creador, Dios mi buen padre celestial por la salud, vida y sabiduría para lograr cumplir cada una de mis metas.

A mis amados padres por ser el apoyo incondicional en el transcurso de esta travesía de estudio, por sus oraciones a la distancia, sus consejos y su amor sin medida a la hora de apoyarme.

A mi amada esposa por ser un pilar en mi vida personal y profesional. A todas las personas que confiaron en mí y me brindaron su apoyo en el transcurso de esta mis estudios de la maestría.

AGRADECIMIENTO

A DIOS todo poderoso, por la vida, la salud y dones del espíritu santo. A mi tutora, por ser una guía en el proceso de titulación, por la dedicación y paciencia en el transcurso del desarrollo de la tesis, por el apoyo en cada uno de los procesos difícil de arduo trabajo y por impartir sus conocimientos para cumplir esta meta.

A cada uno de los docentes que formaron parte de este camino del conocimiento, por su dedicación y compartir sus conocimientos que fueron fundamentales para mi formación académica y profesional.

RESPONSABILIDAD DE AUTORÍA

Por medio de la presente declaro ante el Comité Académico de la maestría en software de la Universidad Técnica de Machala, que el trabajo de titulación, titulado “Metodología de selección de lenguajes de programación para desarrollar software en un ambiente de enseñanza-aprendizaje”, de mi propia autoría, no contiene material escrito por otra persona al no ser referenciado debidamente en el texto, parte de ella o en su totalidad no ha sido aceptada para el otorgamiento de cualquier otro diploma de una institución nacional o extranjera.

Juan Pablo Araujo Cortez

CI 0104708235

Machala, 18 de octubre de 2021

REPORTE TURNITIN

INFORME DE ORIGINALIDAD

6%

INDICE DE SIMILITUD

6%

FUENTES DE INTERNET

3%

PUBLICACIONES

2%

TRABAJOS DEL
ESTUDIANTE

FUENTES PRIMARIAS

1

www.espae.espol.edu.ec

Fuente de Internet

<1%

2

archive.org

Fuente de Internet

<1%

3

elempleodehoy.blogspot.com

Fuente de Internet

<1%

4

idoc.pub

Fuente de Internet

<1%

5

docplayer.es

Fuente de Internet

<1%

6

worldwidescience.org

Fuente de Internet

<1%

7

nahoul.me

Fuente de Internet

<1%

8

buscaintegrada.ufrj.br

Fuente de Internet

<1%

9

gestion.pe

Fuente de Internet

<1%

10	di.umons.ac.be Fuente de Internet	<1 %
11	www.theibfr.com Fuente de Internet	<1 %
12	itsacayucan.edu.mx Fuente de Internet	<1 %
13	prezi.com Fuente de Internet	<1 %
14	doaj.org Fuente de Internet	<1 %
15	www.growkudos.com Fuente de Internet	<1 %
16	bcbl.eu Fuente de Internet	<1 %
17	www.redisd.org Fuente de Internet	<1 %
18	www.scilit.net Fuente de Internet	<1 %
19	Submitted to Universidad Santo Tomas Trabajo del estudiante	<1 %
20	jornadas-innovaciondocente.uca.es Fuente de Internet	<1 %
21	Amy J.C. Trappey, Charles V. Trappey, Jheng-Long Wu, Jack W.C. Wang. "Intelligent	<1 %

compilation of patent summaries using machine learning and natural language processing techniques", Advanced Engineering Informatics, 2020

Publicación

22	www.yumpu.com Fuente de Internet	<1 %
23	investigacion.ucsm.edu.pe Fuente de Internet	<1 %
24	rodna.bn.gov.ar Fuente de Internet	<1 %
25	www.springerprofessional.de Fuente de Internet	<1 %
26	www.uazuay.edu.ec Fuente de Internet	<1 %
27	onlinelibrary.wiley.com Fuente de Internet	<1 %
28	repositorio.utmachala.edu.ec Fuente de Internet	<1 %
29	ninive.ismm.edu.cu Fuente de Internet	<1 %
30	www.3ciencias.com Fuente de Internet	<1 %
31	crossmark.crossref.org Fuente de Internet	<1 %

32	Submitted to Universidad Catolica De Cuenca Trabajo del estudiante	<1 %
33	developer.ibm.com Fuente de Internet	<1 %
34	explora.unex.es Fuente de Internet	<1 %
35	www.um.es Fuente de Internet	<1 %
36	suche.thulb.uni-jena.de Fuente de Internet	<1 %
37	www.lsi.us.es Fuente de Internet	<1 %
38	Javier Cornejo-Reyes, Jessica Herrera-Urgiles, Patricio Pacheco-Quezada, Yamila Rodriguez-Lopez. "Pachamanta", a multi-thematic platform for teaching friendly living with the environment.", 2020 IEEE World Conference on Engineering Education (EDUNINE), 2020 Publicación	<1 %
39	espirituemprededortes.com Fuente de Internet	<1 %
40	vufind.katalog.k.utb.cz Fuente de Internet	<1 %

Excluir citas Activo

Excluir bibliografía Activo

Excluir coincidencias < 20 words

CERTIFICACIÓN DEL TUTOR

Por medio de la presente apruebo que el trabajo de titulación, titulado “Metodología de selección de lenguajes de programación para desarrollar software en un ambiente de enseñanza-aprendizaje” del autor Juan Pablo Araujo Cortez en opción al título de Master en Software, sea presentado al Acto de Defensa.

Ing. Jennifer Céleri Pacheco,
MscCI 0704259373

Machala 18 de octubre de 2021

CESIÓN DE DERECHO DE AUTORÍA

Yo, JUAN PABLO ARAUJO CORTEZ, en calidad de autor del presente trabajo titulado “METODOLOGÍA DE SELECCIÓN DE LENGUAJES DE PROGRAMACIÓN PARA DESARROLLAR SOFTWARE EN UN AMBIENTE DE ENSEÑANZA-APRENDIZAJE” autorizo a la UNIVERSIDAD TECNICA DE MACHALA, la publicación y distribución en el repositorio DigitalInstitucional de la Universidad Técnica de Machala.

El autor declara que el contenido que se publicará es de carácter académico y enmarca en las disposiciones definidas por la Universidad Técnica de Machala.

Ing. Juan Pablo Araujo Cortez

CI: 0104708235

Machala, 18 de octubre de 2021

RESUMEN

Actualmente existe una gran cantidad de lenguajes de programación, pero al momento de seleccionar el lenguaje adecuado se presentan varios factores que dificultan esa elección, más aún cuando se busca implementarlo en un ambiente de enseñanza-aprendizaje. El presente estudio pretende solucionar esta problemática a través del desarrollo de una metodología. Esta será la encargada de tomar en consideración las tendencias actuales de programación y valorar los criterios de evaluación de expertos para una construcción más eficiente. Partiendo de una revisión sistemática de la literatura. Se lleva a cabo una metodología de investigación experimental y correlacional. Se complementa el diseño de la investigación con un enfoque mixto (cualitativo-cuantitativo). A esto se suma para la recolección y análisis de datos, técnicas tales como entrevistas, encuestas, observación y revisión de documentación. Todo el proceso está articulado a construir una metodología de selección basado en los parámetros de análisis, tendencias actuales de programación y criterios técnicos.

Es fundamental en el estudio el destacar que se presenta en contraste de dos escenarios. El primer escenario plantea el no usar la metodología y el segundo con la implementación de la misma. Todo esto en el fin de comparar resultados finales para una interpretación efectiva. Con esto, se busca el fortalecer el proceso de formación tecnológica de los institutos superiores de Azuay y Cañar. Este trabajo se centra en la carrera de desarrollo de software como eje principal del presente estudio. Con el estudio se logró establecer una metodología de selección de lenguajes de programación orientados a objetos para desarrollar software en un ambiente de enseñanza aprendizaje presencial. Basado en tendencias actuales de programación y evaluando la implementación de la misma.

ABSTRACT

Currently there are a large number of programming languages, but when selecting the appropriate language there are several factors that make that choice difficult, even more so when it is sought to implement it in a teaching-learning environment. The present study aims to solve this problem through the development of a methodology. This will be in charge of taking into account current programming trends and assessing the evaluation criteria of experts for a more efficient construction. Starting from a systematic review of the literature. An experimental and correlational research methodology is carried out. The research design is complemented with a mixed approach (qualitative-quantitative). To this is added for the collection and analysis of data, techniques such as interviews, surveys, observation and documentation review. The whole process is articulated to build a selection methodology based on the analysis parameters, current programming trends and technical criteria.

It is essential in the study to highlight that it is presented in contrast to two scenarios. The first scenario raises not using the methodology and the second with its implementation. All this in order to compare final results for an effective interpretation. With this, it seeks to strengthen the technological training process of the higher institutes of Azuay and Cañar. This work focuses on the software development career as the main axis of this study. With the study, it was possible to establish a methodology for selecting object-oriented programming languages to develop software in a face-to-face teaching-learning environment. Based on current programming trends and evaluating its implementation.

Tabla de contenido

DEDICATORIA

AGRADECIMIENTO	3
RESPONSABILIDAD DE AUTORÍA	4
REPORTE TURNITIN	5
CERTIFICACIÓN DEL TUTOR.....	10
CESIÓN DE DERECHO DE AUTORÍA.....	11
RESUMEN.....	12
ABSTRACT.....	13
Tema	20
Introducción	20
Capítulo 1	26
1. Generalidades.	26
1.1. Antecedentes Históricos de la Investigación	26
1.1.1. Institutos Tecnológicos Superiores del Ecuador.....	28
1.1.2. Planteamiento del problema	32
1.1.3. Pregunta de Investigación.....	33
1.1.4. Proceso de búsqueda	33
1.1.5. Criterios de inclusión y exclusión	33
1.1.6. Cadena de búsqueda.....	33
1.1.7. Resultado de la revisión	34
1.2. Antecedentes Conceptuales de la Investigación	40
1.2.1. Hipótesis de la investigación.....	40
1.2.2. Red de categorías de las variables	40
1.2.3. Fundamentación teórica de la variable Independiente	40
1.2.3.1. Software.....	40
1.2.3.2. Desarrollo de Software.....	41
1.2.3.3. Lenguajes de Programación.....	43
1.3. Fundamentación teórica de la variable Dependiente.....	45
1.3.1. Programación Orientada a Objetos.....	45
1.3.1.1. Herencia.....	46
1.3.1.2. Encapsulamiento.....	46
1.3.1.3. Abstracción	46

1.3.1.4.	Polimorfismo	46
1.3.2.	Lenguajes de Programación Visuales (VPL) y lenguajes de programación textuales (TPL) 47	
1.3.2.1.	Lenguajes de Programación Visuales (VPL) y lenguajes de programación textuales (TPL) consisten en elementos icónicos que pueden interactuar manipulado de acuerdo con alguna gramática espacial [27] 47	
1.3.3.	Selección de lenguajes de programación	48
1.3.4.	Lineamientos para elaborar la metodología.....	52
1.4.	Antecedentes Contextuales de la Investigación.....	56
1.4.1.	Delimitación del contexto de estudio	56
1.4.2.	Propuesta de solución y contribuciones	56
1.4.3.	Organización del documento.....	57
Capítulo 2	58
2.	Generalidades.	58
2.1.	Tipo de estudio o investigación realizada	58
2.2.	El paradigma o enfoque en el cual se realizó.....	58
2.3.	Diseño de investigación mixto (Cualitativo-Cuantitativo).....	58
2.4.	Cálculo de la población y muestra	59
2.5.	Métodos teóricos con los materiales utilizados.	60
2.6.	Métodos empíricos con los materiales utilizados.	61
2.6.1.	Definiciones para la búsqueda	62
2.6.1.1.	Preguntas de investigación	62
2.6.1.2.	Alcance de la revisión	62
2.6.1.3.	Criterios de Inclusión y Exclusión	63
2.6.1.4.	Conductas de búsqueda	64
2.6.2.	Proceso de Búsqueda.....	64
2.6.3.	Discusión de resultados	64
2.7.	Técnicas estadísticas para el procesamiento de datos obtenidos.	65
Capítulo 3	66
3.	Generalidades.	66
3.1.	Aplicación de técnicas de recolección de datos	66
3.1.1.	Observación	66
3.1.2.	Revisión de documentación	67
3.1.3.	Análisis de herramientas por asignatura.....	77
3.2.	Entrevistas	77

3.3.	Encuestas	87
3.3.1.	Escenario 1.....	88
3.4.	Metodología de Selección	95
3.4.1.	Fases de la metodología de selección de lenguajes de programación orientada a objetos. 95	
3.4.2.	Formato de Ficha de fases	104
3.5.	Aplicación de metodología.....	104
3.5.1.	Escenario 2.....	105
	Capítulo 4	108
4.	Generalidades.	108
4.1	Metodología de selección	108
4.2.	Tendencias actuales de programación	109
4.3.	Recolectar y tabular información.....	109
4.4.	Evaluar la metodología	111
	Conclusiones	116
	Recomendaciones	117
	Bibliografía.....	118

Lista de ilustraciones

Ilustración 1 Campo amplio de las Tics	17
Ilustración 2 Oferta académica Institutos Superiores Tecnológicos	18
Ilustración 3 Carreras Tecnologías de la Información y la Comunicación	19
Ilustración 4 Azuay, Tecnologías de la Información y la Comunicación	20
Ilustración 5 Cañar, Tecnologías de la Información y la Comunicación	20
Ilustración 6 Fundamentos de la Programación Orientada a Objetos (POO)	28
Ilustración 7 Rank Languages by popularity	31
Ilustración 8 Índice Tiobe, ranking noviembre 2020	33
Ilustración 9 Proceso de evaluación	40
Ilustración 10 Cálculo de la muestra.	41
Ilustración 11 Técnicas de recolección de datos	43
Ilustración 12 Proceso de mapeo.	44
Ilustración 13 Proceso de enseñanza-aprendizaje	49
Ilustración 14 Malla curricular 1. Juan Bautista Vásquez / Azuay (JBV/AZ)	43
Ilustración 15 Malla curricular 2. Enrique Noboa Arizaga (ENA)	44
Ilustración 16 Malla curricular 3. Integración Andina (INAN)	45
Ilustración 17 Malla curricular 4. Sudamericano (SUDA)	46
Ilustración 18 Jerarquía de roles IES	56
Ilustración 19 Años de existencia de la carrera de desarrollo de software.	63
Ilustración 20 Tipo de software empleado en el proceso de enseñanza-aprendizaje	63
Ilustración 21 Lenguajes de programación enseñados.	64
Ilustración 22 Documentos o lineamientos formales de selección de lenguajes de programación	65
Ilustración 23 Considerar uso de una metodología.	66
Ilustración 24 Escenario 1, sin implementación de metodología.	67
Ilustración 25 Experiencia con algún lenguaje de programación.	69
Ilustración 26 Conocimiento de las tendencias actuales de programación.	70
Ilustración 27 Dar a conocer herramientas usadas en el proceso	70
Ilustración 28 Dar a conocer parámetros de evaluación	71
Ilustración 29 Temáticas de evaluación.	72
Ilustración 30 Grafico estadístico. Medición sin metodología	74
Ilustración 31 Metodología de selección de lenguajes de programación orientados a	

objetos	76
Ilustración 32 Fase de diseño del proyecto	77
Ilustración 33 Índice Tiobe, tendencias de programación.	78
Ilustración 34 Índice PYPL, tendencias de programación.	78
Ilustración 35 Índice GITHUB, tendencias de programación.	79
Ilustración 36 Fase de tendencias	80
Ilustración 37 Fase de implementación	81
Ilustración 38 Fase de seguimiento	82
Ilustración 39 fase de cierre	82
Ilustración 40 Escenario 2, con la implementación de metodología.	84
Ilustración 41 Objetivo general. Metodología de selección	87
Ilustración 42 Aceptación sin una implementación de Metodología	88
Ilustración 43 Aceptación con la implementación de Metodología	89

Lista de tablas

Tabla 1 Elementos de las Tics	16
Tabla 2 Resultado de la revisión	23
Tabla 3 Senescyt / Sniese. En línea. Ultimo ingreso: abril 2011	25
Tabla 4 Senescyt / Sniese. En línea. Ultimo ingreso: abril 2011	26
Tabla 5 Criterios de Selección	35
Tabla 6 Análisis de resultados obtenidos	47
Tabla 7 Estadística descriptiva	48
Tabla 8 Asignaturas candidatas	48
Tabla 9 Asignaturas candidatas finales para aplicar al estudio.	48
Tabla 10 Contenidos mínimos Fundamentos de Programación	51
Tabla 11 Contenidos mínimos Programación Orientada a Objetos	55
Tabla 12 Elementos considerados en las entrevistas.	57
Tabla 13 Síntesis de entrevistas.	61
Tabla 14 Escenarios de estudio.	66
Tabla 15 Preguntas encuesta	69
Tabla 16 Evaluación del lenguaje de programación usado	72
Tabla 17 Rúbrica de evaluación de prácticas de programación	74
Tabla 18 Indicador numérico de aceptación	75
Tabla 19 Formato de ficha de fases de metodología	83
Tabla 20 Aplicación de metodología de selección	85
Tabla 21 Pasos del método Delphi	90

Tema:

“Metodología de selección de lenguajes de programación para desarrollar software en un ambiente de enseñanza-aprendizaje”.

Introducción

En 1990, el Consejo de Educación Superior (CES) implementó el Reglamento para crear y poner en funcionamiento Institutos Tecnológicos Superiores, los que se encargan de brindar educación en el ámbito tecnológico [1]. Cabe destacar las transformaciones en cuanto al fortalecimiento a nivel técnico y tecnológico que se han dado hasta el momento en los Institutos de Educación Superior en Ecuador y esencialmente en los Institutos Superiores Tecnológicos [2]. Por otra parte, se debe indicar que en la reforma del Artículo 18 de la Ley Orgánica de Educación Superior (LOES) se permite, a partir del 2 de agosto de 2019, que los Institutos Tecnológicos otorguen títulos de tercer nivel. En Ecuador existe un total de 241 institutos técnicos y tecnológicos clasificados entre particular autofinanciada, particular cofinanciada y pública con oferta académica vigente. En el área de Tecnologías de la información y la comunicación (TIC) existen 76 ofertas académicas que rigen con base a la modalidad de financiamiento y campo de conocimiento [3].

En la actualidad, las Tecnologías de la información y la comunicación, con el uso de computadoras y la forma de programarlas han evolucionado de una forma vertiginosa [4]. Hay que comprender que programar en un lenguaje de programación (LP) es el conjunto de instrucciones similares al inglés que incluye un conjunto de reglas y sintaxis para unir las instrucciones y crear comandos [5].

Por otra parte, cabe recalcar que el software de una computadora es un producto intangible y contiene una secuencia o instrucciones construida a través de un lenguaje de programación que se enfoca en cumplir un requerimiento del usuario [6]. En el ámbito del desarrollo de software, específicamente el área de la docencia en los institutos de educación superior, es necesario entender el contexto académico de uso y la selección de los lenguajes de programación en los diferentes Institutos Tecnológicos. De esta forma se pueden plantear variables clave que permitan definir una metodología

con resultados. El desarrollar en el estudiante una mentalidad lógica y sistemática al momento de usar los lenguajes de programación idóneos, con lo cual se podrá alcanzar competencias y destrezas adecuadas en el proceso de enseñanza-aprendizaje. Una metodología de selección permite determinar el lenguaje de programación idóneo para implementar en un entorno de enseñanza-aprendizaje para desarrollar software [7].

La problemática parte de que actualmente existe gran diversidad de lenguajes de programación orientados a objetos para el desarrollo de software. Esto dificulta la selección del lenguaje idóneo para aplicarse en un ambiente de enseñanza aprendizaje en los Institutos de Educación Superior. A esto, se suma el no contar con una metodología de selección de lenguajes de programación para los Institutos de Educación Superior de Azuay y Cañar. Finalmente, y siguiendo el análisis del problema, se puede recalcar que las tendencias de programación varían acorde a diferentes parámetros como son las necesidades de las aplicaciones, desarrolladores, entre otros factores.

A través del presente estudio, se plantea una solución a esta problemática. Los docentes y estudiantes se verían en gran parte beneficiados a través de una metodología definida que les permita identificar y establecer el lenguaje de programación para impartir en las respectivas asignaturas que manejen el proceso de desarrollo de software. Se debe destacar que particularmente, la enseñanza-aprendizaje de la programación de computadoras no es una actividad sencilla para los docentes ni para los estudiantes [8]. En el ámbito educativo es esencial fortalecer el proceso de enseñanza-aprendizaje de los Institutos de Educación Superior de Azuay y Cañar, con la cual se cuenta información para tabular y analizar. A la hora de seleccionar un lenguaje de programación idóneo para el proceso, enfocado en la información recolectada.

El estudio se enfoca en establecer una metodología de selección de lenguajes de programación orientados a objetos para desarrollar software en un ambiente de enseñanza aprendizaje presencial. En esencia, esta metodología se centra en la selección de lenguajes de programación usados en las carreras que involucran el desarrollo de software específicamente en las asignaturas de programación. De esta manera, se debe mencionar que es fundamental para los docentes de educación superior en la carrera de desarrollo de software, buscar, identificar y establecer lineamientos claves que ayuden a entender cómo definir las herramientas idóneas para el desarrollo de software.

Actualmente en la provincia de Azuay existen siete Institutos entre técnicos y

tecnológicos, pero los que ofertan la carrera de desarrollo de software son: Instituto Tecnológico Superior Sudamericano, Instituto Superior Tecnológico del Azuay, Instituto Tecnológico Superior Integración Andina; por otra parte en la provincia de Cañar se cuenta con dos Institutos que ofertan la carrera de desarrollo de software, los cuales son: Instituto Tecnológico Superior Juan Bautista Vásquez y el Instituto Tecnológico Superior Enrique Noboa Arizaga [3].

La población de análisis de este trabajo se enfoca en los cinco institutos de educación superior de Azuay y Cañar, los cuales ofertan la carrera de desarrollo de software. De esta manera se empezará con plantear un análisis y establecer parámetros de los lenguajes de programación más usados, basados en las tendencias actuales de lenguajes de programación. Estas tendencias vienen marcadas por ranking de uso basado en distintos parámetros en comunidad de programadores a nivel global. Entre los indicadores más destacados se puede mencionar Tiobe y Pypl. Estos rankings permiten el tomar con partida los lenguajes de programación más populares de la comunidad de desarrolladores, aportando a la construcción del estudio. Con esto, se pretende contrastar los rankings de tendencias de programación actuales con los usados en el proceso de enseñanza- aprendizaje. Esto conlleva a la construcción de un cuadro comparativo en base a parámetros técnicos que encamine a entender y estructurar de mejor manera el estudio. A esto se suma el aportar con cuál es el lenguaje de programación, más efectivo a la hora de iniciar un proceso de aprendizaje en el desarrollo de software [9]. A esto se suma que la carrera de desarrollo de software, tienen en su malla la asignatura de tendencias actuales de programación.

La recolección de información se la realizará con base en la aplicación de técnicas como entrevistas y encuestas dirigidas a docentes y profesionales del área de desarrollo de software. A esto se suma el proponer la metodología para la selección de lenguajes de programación con base en criterios técnicos y análisis. Finalmente, se realizará una evaluación de la metodología propuesta de selección de lenguajes de programación en la cual se contrastaron los resultados comparativos de antes de implementar la metodología y posterior implementación de la misma [10].

El trabajo inicia con referencias de índices de clasificación a nivel global, lo que permite tener una perspectiva general del uso de lenguajes de programación. El usar los lenguajes de programación con base en tendencias de desarrollo actuales, fortalece el proceso de enseñanza-aprendizaje. Estos rankings presentados permiten obtener criterios

de diversos profesionales en el ámbito de desarrollo a nivel mundial y fortalece las bases del proyecto para construir una metodología robusta y eficiente a la hora de seleccionar un lenguaje de programación.

El desarrollo de la investigación busca fortalecer el ámbito de enseñar un lenguaje de programación en un proceso de enseñanza aprendizaje de carreras de desarrollo de software. El determinar el lenguaje a usar permitirá el aprendizaje más eficiente para los estudiantes conforme el avance de cada ciclo de estudio y permitirá el uso del mismo en una rama específica del desarrollo.

El documento consta de cuatro capítulos y se encuentra estructurado de la siguiente manera: (i) en el capítulo uno se aborda el análisis bibliográfico y fundamentos teóricos de la investigación de los cuales son punto de partida para argumentar científicamente el estudio, (ii) el capítulo dos presenta las técnicas de recolección de información a través de encuestas, entrevistas y la metodología de investigación, (iii) el tercer capítulo define la relevancia de información recolectada, y la propuesta del desarrollo de la metodología de selección, (iv) en el cuarto capítulo se establecen una comparativa y se contrasta resultados; además de las conclusiones y recomendaciones de la investigación.

Capítulo 1

1. Generalidades.

En este capítulo se detallan la introducción, planteamiento del problema, hipótesis, antecedentes, justificación e importancia, objetivos, alcance, metodología de la investigación y herramientas usadas.

1.1. Antecedentes Históricos de la Investigación.

En la actualidad, aprender a utilizar un lenguaje de programación es una alternativa viable que provee de medios tecnológicos a los desarrolladores de software para automatizar procesos. A la hora de implementar el aprendizaje de un lenguaje de programación orientado a objetos, existen varias alternativas en el mercado del desarrollo de software. Los Institutos Superiores Tecnológicos ofertan a través de la carrera Desarrollo de Software esta área académica de estudio. A la hora de implementar el aprendizaje de un lenguaje de programación en un ambiente de enseñanza aprendizaje presencial no existe un modelo o los lineamientos adecuados que permitan refinar los criterios de selección de un lenguaje de programación.

Hoy en día, el uso de lenguajes de programación ha dado un salto importante al centrar procesos en el uso de nuevas tecnologías permitiendo surgir disponibilidad para recursos de Tecnología de la información y la comunicación y sus ámbitos de aplicación. Por consiguiente, el aprendizaje de Programación de Algoritmos muestra, quizás, uno de los niveles de mayor dificultad en los procesos de Desarrollo de software [6]. Visto desde el punto tecnológico, la infraestructura de la Tecnología de la Información y la Comunicación, se enfoca en hardware y aplicaciones de software necesarias para automatizar procesos de una empresa. Todo esto conlleva a la importancia de entender esta estructura, la cual se encuentra fusionada al proceso de desarrollar software, por ende, es claro el identificar el proceso en que se encuentran situados hitos claves para el presente estudio como se muestra en la Tabla 1.

Elementos de Tics	
Hardware	Computadoras, servidores, medios de almacenamiento, aparatos móviles, impresoras, equipos de redes, entre otros.
Software	Aplicaciones para productividad, negocios, redes, sistemas, seguridades, apps móviles.
Servicios	Instalación, integración, desarrollo personalizado y servicios de administración.
Infraestructura	Red troncal de internet, redes de telecomunicaciones, centros de datos en la nube.
Información	Datos, documentos, voz, video, imágenes.
Negocios Digitales	Comercio, comunicación, colaboración, automatización, gobernanza.

Tabla 1 Elementos de las Tics Fuente: CompTIA

Inicialmente el software conlleva un proceso de aprendizaje con los futuros profesionales desarrolladores. A esto se añade la formación que tienen los estudiantes de la carrera de desarrollo de software. Este trabajo correctamente llevado con las herramientas y recursos adecuados fortalecen de manera significativa el aprendizaje y asegura mejores profesionales en el ámbito [17]. De esta forma el software se desarrolla por profesionales que se prepararon académicamente para realizar un producto de calidad. Las carreras de Desarrollo de Software que ofertan Institutos Superiores Tecnológicos se centran en determinar la herramienta idónea; específicamente centrados en el lenguaje de programación orientado a objetos que se debe usar en su proceso de enseñanza aprendizaje.

La educación superior tecnológica se enfoca en promover el talento de los futuros profesionales en el desarrollo de software motivando el desarrollo de la investigación, innovación y transferencia tecnología de la Senescyt, encaminados a cumplir las normas para obtener el cumplimiento objetivos institucionales [1].

El Gobierno Nacional está empeñado en implementar un sistema de educación superior inclusivo y de calidad. En ese contexto, la Senescyt ha impulsado acciones que promuevan un sistema de acceso meritocrático, en igualdad de oportunidades y con garantía de la libre elección para las y los aspirantes, conforme lo determinado en la Ley

Orgánica de Educación Superior. La Senescyt junto con instituciones de educación superior del Ecuador han implementado estrategias que permitieron elevar los cupos en comparación a otras convocatorias en el ámbito de Tics [1].

1.1.1. Institutos Tecnológicos Superiores del Ecuador

Según la Senescyt, hasta el 2018 existían un total de 241 institutos de educación superior con la oferta académica vigente. De estos, 112 públicos, 120 particulares y 9 particulares cofinanciados [1].

Es fundamental partir de esta información estadística que permita tener una visión clara del panorama de la investigación, lo cual permita entender e identificar la creación de una metodología de selección de lenguaje de programación para un proceso de enseñanza- aprendizaje. La información estadística determina el campo amplio de Tics, y define esencialmente el enfoque a las carreras que manejen el proceso de desarrollo de software como se puede apreciar en la Ilustración 1.

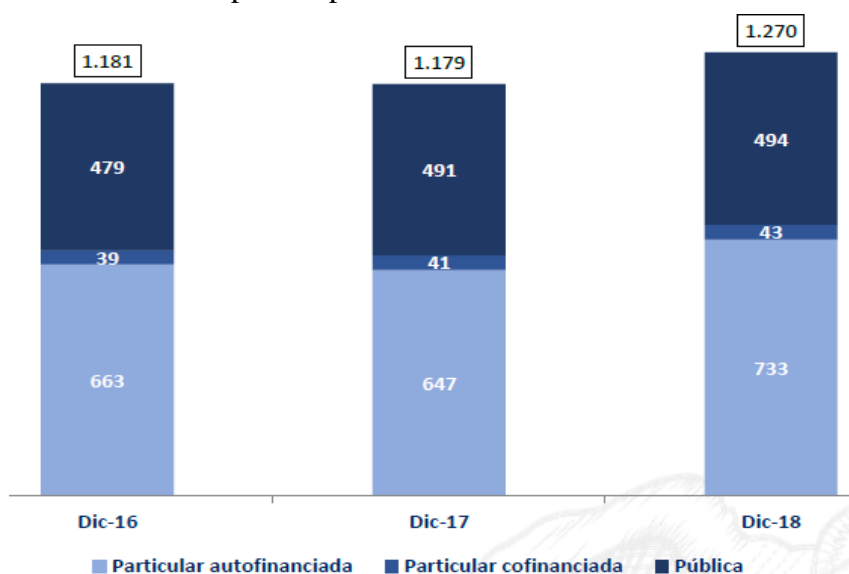


Ilustración 1 Campo amplio de las Tics Fuente: SNIESE

Es importante el conocer que oferta académica en institutos tecnológicos que se presenta en la Ilustración 2.

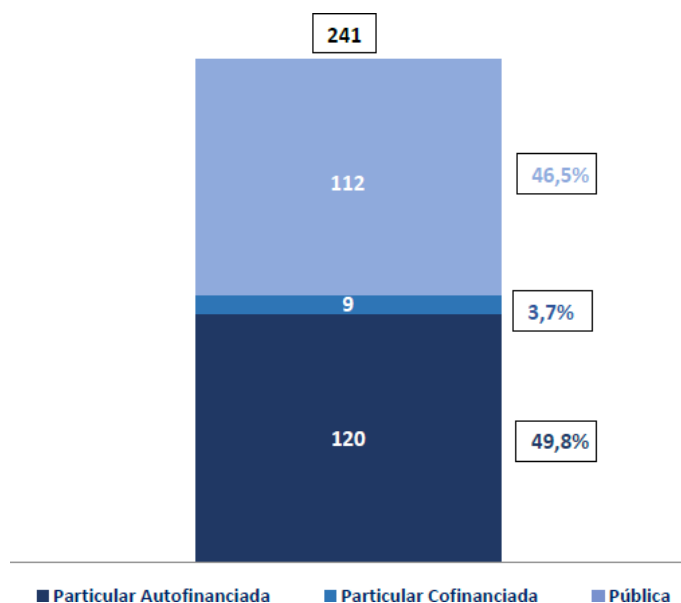


Ilustración 2 Oferta académica Institutos Superiores Tecnológicos

Fuente: SNIESE

Las Instituciones de Educación Superior Tecnológica cuentan con la evaluación realizada por el Consejo de Aseguramiento de la Calidad de la Educación Superior, que los determina como: “Acreditado”, “En proceso de Acreditación”, “En proceso de Acreditación Condicionado”, “En proceso de Acreditación Fuertemente Condicionado” y “No Acreditado”. Además, es importante dar a conocer que se reconoce como profesionales con título de tercer nivel tecnológico a estudiantes que se están formando en determinada área o que culminaron su formación en años pasados.

El artículo 118 de la LOES indica: "Los institutos superiores técnicos y tecnológicos pueden otorgar títulos de tercer nivel tecnológico superior; y, los institutos superiores que tengan la condición de instituto superior universitario podrán otorgar además los títulos de tercer nivel tecnológico superior universitario y posgrados tecnológicos; se priorizará la oferta técnico-tecnológica en estos institutos frente a la oferta de las universidades y escuelas politécnicas" [2].

Cabe indicar que la oferta académica de carreras vigentes, según modalidad de financiamiento y campo de conocimiento amplio en institutos tecnológicos se presenta enfatizando la línea de investigación de proyecto que se encamina hacia las Tecnologías de la Información y la comunicación, en la cual se encuentra enmarcada la carrera de

tecnología superior en desarrollo de software que es el punto clave del estudio.

La figura a continuación indica la oferta académica en diferentes ámbitos. Es claro identificar el área de Tecnologías de la Información y la Comunicación, en la cual se enfoca el presente estudio. Además, se puede evidenciar la cantidad de institutos superiores que ofertan carreras tecnológicas en la cual destacan en primera instancia los públicos seguido de los particulares y finalmente los particulares cofinanciados. Con esto permite alcanzar una visión general y clara de la acogida y crecimiento del ámbito de las Tics que aportan al estudio, a continuación, se presenta en la Ilustración 3.



Ilustración 3 Carreras Tecnologías de la Información y la Comunicación

Fuente: SNIIESE

Es trascendental indicar que la investigación se centra en las provincias de Azuay y Cañar, en este contexto, en la provincia de Azuay, la oferta académica de carreras vigentes con las que se cuenta en el ámbito de Tecnologías de la Información y la Comunicación (Tics) se refleja en la Ilustración 4.

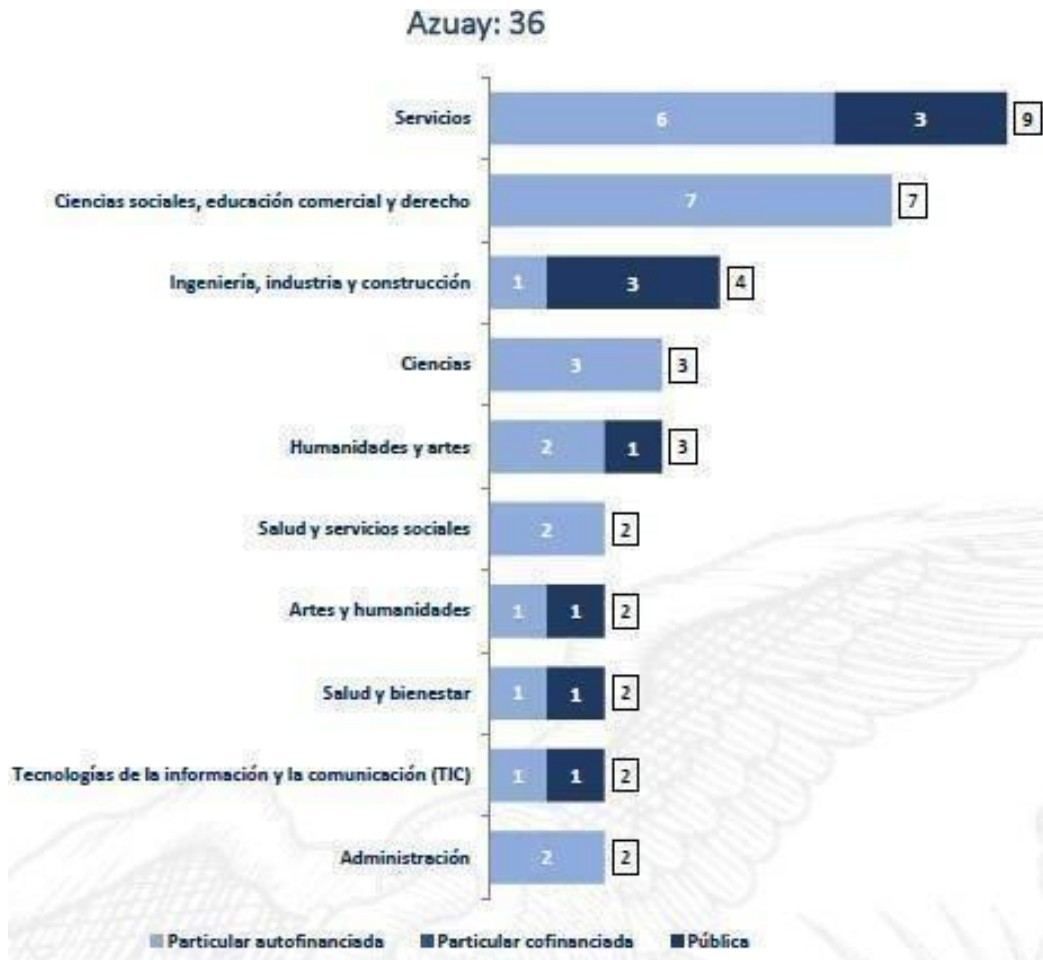


Ilustración 4 Azuay, Tecnologías de la Información y la Comunicación

Fuente: SNIесе

Sumado a esto, se indica por también que en la provincia de Cañar se cuenta con los siguientes datos estadísticos presentados en la Ilustración 5.

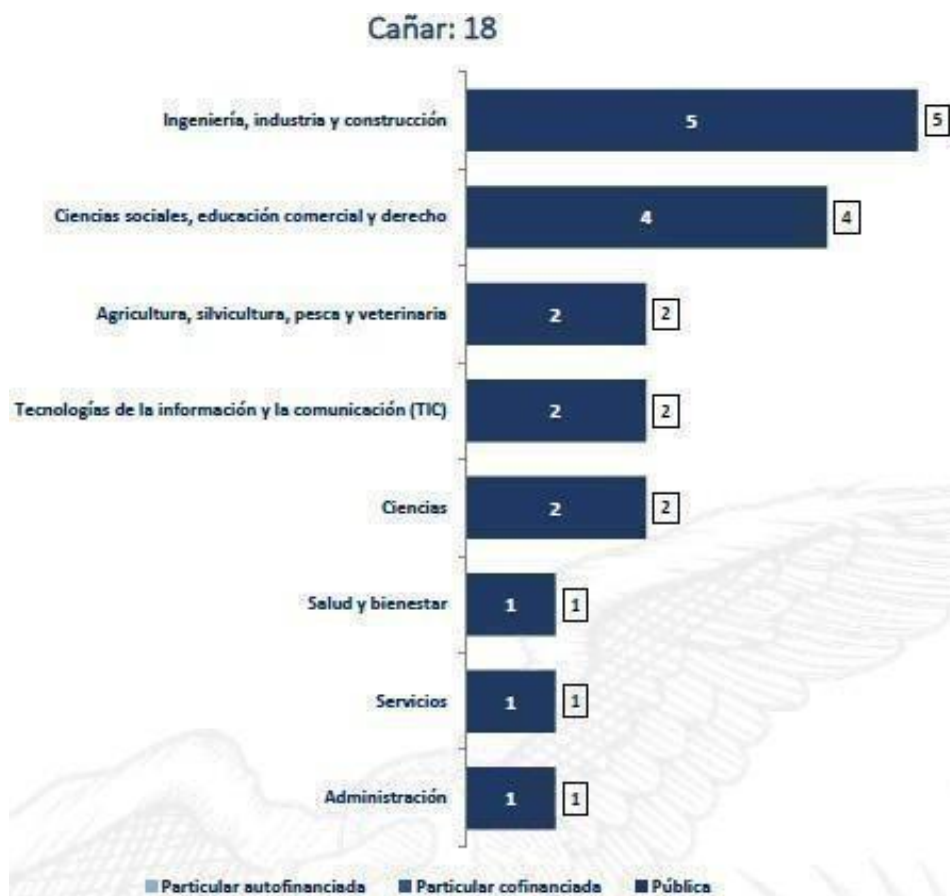


Ilustración 5 Cañar, Tecnologías de la Información y la Comunicación

Fuente: SNIESE

El presente estudio tomará datos de los Institutos Superiores Tecnológicos de las provincias de Azuay y Cañar con las que mayoritariamente se cuenta con acceso a la información. Además, en Azuay existen tres institutos, los cuales ofertan la carrera de desarrollo de software, entre ellos: Instituto Tecnológico Superior Sudamericano, Instituto Superior Tecnológico del Azuay, Instituto Tecnológico Superior Integración Andina. Por otra parte, en la provincia de Cañar se encuentran Institutos Superiores que ofertan la carrera de desarrollo de software entre los cuales están: Instituto Tecnológico Superior Juan Bautista Vásquez y el Instituto Tecnológico Superior Enrique Noboa Arizaga [3].

1.1.2. Planteamiento del problema

Actualmente la diversidad de lenguajes de programación orientados a objetos para el desarrollo de software existentes, dificulta la selección del idóneo para aplicar en un

ambiente de enseñanza aprendizaje en los institutos de educación superior. Las tendencias de programación cambiantes que se adecuan las necesidades del cliente y programadores. A esto, además se suma, que no existe un modelo o metodología de selección de lenguajes de programación en los Institutos de Educación Superior de Azuay y Cañar.

En este contexto inicial se plantean los siguientes objetivos:

General:

- ✓ Desarrollar una metodología de selección de los lenguajes de programación para desarrollo de software, aplicado a un ambiente de enseñanza-aprendizaje presencial para los institutos de educación superior de las provincias de Azuay y Cañar.

Específicos:

- ✓ Establecer los lenguajes de programación más usados, basado en las tendencias de desarrollo de software actuales para la construcción de un cuadro comparativo en base a parámetros técnicos.
- ✓ Recolectar información en base a técnicas como entrevistas y encuestas, enfocadas a docentes y profesionales del área de desarrollo de software.
- ✓ Proponer la metodología para la selección de lenguajes de programación en base a criterios técnicos y análisis de recolección recolectada.
- ✓ Evaluar la metodología propuesta de selección de lenguajes de programación para el desarrollo de software.

1.1.3. Pregunta de Investigación

¿Cómo realizar la selección de un lenguaje de programación orientado a objetos idóneo para el desarrollo de software para implementarlo en un ambiente de enseñanza-aprendizaje presencial?

1.1.4. Proceso de búsqueda

El proceso de búsqueda se realizó en base al método científico y los procesos que conllevada una de sus fases de manera sistemática y metódica. Se enfocó en indagar en fuentes secundarias como bases de datos de contenido de artículos científicos. Entre estas bases de datos se puede destacar Scopus, Web of Science, IEEE, Google Scholar.

1.1.5. Criterios de inclusión y exclusión

En este contexto hay que aclarar que los criterios de exclusión permitieron limitar la búsqueda significativamente, ya que se limita el estudio esencialmente a los lenguajes de programación con el paradigma de orientación a objetos.

1.1.6. Cadena de búsqueda

La construcción y definición de la cadena de búsqueda, se estructuró en base a palabras clave y variables relacionadas al objeto de estudio. Pero es trascendental destacar que no se encontró un tema investigado relacionado a este estudio, lo que se logró encontrar son estudios interesantes que aportan a las variables dependientes e independientes. En tal virtud se realizaron cadenas de búsqueda relacionadas con: Tendencias de Programación, Programación Orientada a Objetos, Lenguajes de programación específicos (Java, Python, C#), Proceso de enseñanza Aprendizaje de Programación.

1.1.7. Resultado de la revisión

Aplicando la Revisión Sistemática de la Literatura, se destaca que se obtuvo un total de 59 investigaciones, las cuales aportan el presente estudio. Cada uno de estos artículos forma parte de este estudio sumando con temas y puntos clave que permiten definir de mejor manera el curso de la investigación, los resultados se presentan a continuación en la Tabla 2.

NUMERO	TIPO	AÑO	AUTORES	TITULO
1	journalArticle	2019	Arslan, Sibel; Öztürk, Celal	A Comparative Study of Automatic Programming Techniques
2	journalArticle	2020	Gmys, Jan; Carneiro, Tiago; Melab, Nouredine; Talbi, El-Ghazali; Tuytens, Daniel	A comparative study of high-productivity high-performance programming languages for parallel metaheuristics
3	book	2016	Feo, John T.	A comparative study of parallel programming languages: the Salishan problems
4	journalArticle	2018	Eichhorn, Helge; Cano, Juan Luis; McLean, Frazer; Anderl, Reiner	A comparative study of programming languages for next-generation astrodynamics systems
5	conferencePaper	2015	Nanz, Sebastian; Furia, Carlo A.	A Comparative Study of Programming Languages in Rosetta Code
6	journalArticle	2016	Ferrara, Pietro	A generic framework for heap and value analyses of object-oriented programming languages
7	journalArticle	2020	Carreño, Yolanda Soler Pellicer / Karina Virginia Mero Suárez / Edwin Joao Merchán	Ambiente integrado de visualización de estructuras de datos para la enseñanza-aprendizaje de la programación de computadoras
8	journalArticle	2018	Cabada, Ramón Zatarain; Estrada, María Lucía Barrón; Hernández, Francisco González; Bustillos, Raúl Oramas; Reyes-García, Carlos Alberto	An affective and Web 3.0-based learning environment for a programming language

9	journalArticle	2015	Salas-Zárate, María del Pilar; Alor-Hernández, Giner; Valencia-García, Rafael; Rodríguez-Mazahua, Lisbeth; Rodríguez-González, Alejandro; López Cuadrado, José Luis	Analyzing best practices on Web development frameworks: The lift approach
10	journalArticle	2020	Luk, Gigi; Pliatsikas, Christos; Rossi, Eleonora	Brain changes associated with language development and learning: A primer on methodology and applications
11	journalArticle	2019	Majd, Amirabbas; Vahidi-Asl, Mojtaba; Khalilian, Alireza; Baraani-Dastjerdi, Ahmad; Zamani, Bahman	Code4Bench: A multidimensional benchmark of Codeforces data for different program analysis techniques
12	journalArticle	2018	Sánchez, Milton Rafael Valarezo Pardo / Joofre Antonio Honores Tapia / Antonio Steeven Gómez Moreno / Luis Fernando Vincés	Comparación de tendencias tecnológicas en aplicaciones web
13	journalArticle	2018	Themistokleous, Carlos Castaño-Garrido / Urtza Garay-Ruiz / Sotiris	De la revolución del software a la del hardware en educación superior
14	journalArticle	2020	Henwood, Tom; Channon, Sue; Penny, Helen; Robling, Mike; Waters, Cerith S.	Do home visiting programmes improve children's language development? A systematic review
15	journalArticle	2020	Latorre, Gabriel	Educación y autonomía: Alfabetización en lectura e interpretación de datos y formación en programación
16	journalArticle	2019	Vera Mosquera, Jorge Francisco / Argüello Fiallos	El aprendizaje de la programación de computadoras para futuros docentes informáticos
17	journalArticle	2020	Ferreira Szpiniak, Ariel / Rojo	Enseñanza de la programación
18	journalArticle	2020	Carrillo, Gonzalo Martín Rodríguez	Enseñanza de la programación de computadoras para principiantes: un contexto histórico
19	journalArticle	2019	Vázquez, Rosa Elizabeth Molina Izurieta / Renzo Rogelio Padilla Gómez / Maikel Yelandi Leyva	Estudio y propuesta metodológica, para la enseñanza-aprendizaje de la programación informática en la educación superior
20	journalArticle	2019	Rodríguez, Rafael Saltos Rivas / Pavel Novoa-Hernández / Rocío Serrano	Evaluación de la presencia de competencias digitales en las Instituciones de Educación Superior en América Latina

21	journalArticle	2020	Rojas-López, Arturo / García-Peñalvo	Evaluación del pensamiento computacional para el aprendizaje de programación de computadoras en educación superior: Assessment of computational thinking skills to predict student learning and retention in the subject programming computer in higher education
22	journalArticle	2020	McCartney, Jane; Boschmans, Shirley-Anne	Evaluation of an intervention to support the development of clinical problem solving skills during a hospital-based experiential learning program for South African pharmacy students
23	journalArticle	2018	Mahmoud, Mohamed Yousri; Felty, Amy P.	Formal Meta-level Analysis Framework for Quantum Programming Languages
24	journalArticle	2019	Solarte, Oswaldo / Villegas	Fortaleciendo la motivación y mejorando el rendimiento de estudiantes de un curso introductorio de programación: Un enfoque de enseñanza integrado
25	journalArticle	2019	Tsai, Chun-Yen	Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy
26	journalArticle	2020	Aka, Natsuki	Incidental learning of a grammatical feature from reading by Japanese learners of English as a foreign language
27	journalArticle	2019	Garro-Aburto, Yolvi Ocaña-Fernández / Luis Alex Valenzuela-Fernández / Luzmila Lourdes	Inteligencia artificial y sus implicaciones en la educación superior
28	journalArticle	2020	Trappey, Amy J. C.; Trappey, Charles V.; Wu, Jheng-Long; Wang, Jack W. C.	Intelligent compilation of patent summaries using machine learning and natural language processing techniques
29	journalArticle	2020	Gao, Kaifeng; Mei, Gang; Piccialli, Francesco; Cuomo, Salvatore; Tu, Jingzhi; Huo, Zenan	Julia language in machine learning: Algorithms, applications, and open issues
30	journalArticle	2018	Solís Sierra, Marlene N.; Duarte Morante, Pedro L.; Solís Sierra, Marlene N.; Duarte Morante, Pedro L.	La educación superior tecnológica y la empleabilidad

31	journalArticle	2018	Tejera, Keila Irene Diaz / Martin	La ensenanza de la programacion. Una experiencia en la formacion de profesores de Informatica/The teaching of programming: An experience in the training of Informatics teachers/O ensino da programacao. Uma experiencia na formacao de professores da Informatica
32	journalArticle	2018	Muñoz Pentón, María Amelia / Fierro Martín	La enseñanza de la programación. Una experiencia en la formación de profesores de Informática
33	journalArticle	2018	Drljača Margić, Branka; Vodopija-Krstanović, Irena	Language development for English-medium instruction: Teachers' perceptions, reflections and learning
34	journalArticle	2020	Soltaninejad, Nasibe; Jalilevand, Nahid; Kamali, Mohammad; Mohamadi, Reyhane	Language therapy outcomes in deaf children with cochlear implant using a new developed program: A pilot study
35	journalArticle	2020	Rojas, Esperanza Manrique	Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo
36	conferencePaper	2017	Khaleel, Firas Layth; Ashaari, Noraidah Sahari; Wook, Tengku Siti Meriam Tengku; Ismail, Amirah	Methodology for developing gamification-based learning programming language framework
37	journalArticle	2020	Román, Leobardo López	Metodología para el Desarrollo de la Lógica de la Programación Orientada a Objetos
38	journalArticle	2020	Román, Leobardo López	Metodologías para la Enseñanza Aprendizaje de la Programación Estructurada y Orientada a Objetos (Artículo Invitado)
39	journalArticle	2020	Toasa, Henry Recalde / Paúl Francisco Baldeón Egas / Miguel Alfredo Gaibor Saltos / Renato M.	Minería de datos con R para información académica en Instituciones de Educación Superior
40	journalArticle	2018	Cuba-Sayco, Julio Vera / Klinge Villalba-Condori / Sonia Castro	Modelo de sistema de recomendación basado en el contexto a partir del análisis de código estático para el desarrollo del Pensamiento Computacional: Caso de Programación Web
41	journalArticle	2015	Ruiz, Jesús García; López, Marisol Hernández; Brito, José Antonio Loaiza	Pensamiento sistémico y desarrollo de competencias, en el aprendizaje de los lenguajes de programación
42	journalArticle	2017	Monsálvez, José Carlos García	Python como primer lenguaje de programación textual en la Enseñanza Secundaria

43	book	2015	Shein, Esther	Python for beginners
44	journalArticle	2018	Monjelat, Natalia / Rodríguez	Repensando la programación como formación práctica en Ingeniería: Un estudio de caso en primer año
45	conferencePaper	2015	Benedetto, Marcelo Gabriel; Carabio, Ana Lía Ramona; Alvez, Carlos E.; Fernández, Miguel; Etchart, Graciela; Cabrera, Sergio Alberto; Falappa, Marcelo Alejandro; Cobo, María Laura; Martínez, Diego C.; Benítez, Horacio Duval	Selección de lenguajes orientados a objetos para un estudio comparativo y análisis de rendimiento
46	journalArticle	2020	Zucco, Chiara; Calabrese, Barbara; Agapito, Giuseppe; Guzzi, Pietro H.; Cannataro, Mario	Sentiment analysis for mining texts and social networks data: Methods and tools
47	journalArticle	2018	Gaddis, Tony	Starting out with Python
48	journalArticle	2017	Deliktas, Derya; Ustun, Ozden	Student selection and assignment methodology based on fuzzy MULTIMOORA and multichoice goal programming
49	journalArticle	2018	Singh, Dharendra Pratap; Joshi, Ishan; Choudhary, Jaytrilok	Survey of GPU Based Sorting Algorithms
50	journalArticle	2015	Dolgopolas, Vladimiras; Dagienė, Valentina; Minkevičius, Saulius; Sakalauskas, Leonidas	Teaching Scientific Computing: A Model-Centered Approach to Pipeline and Parallel Programming with C
51	journalArticle	2019	Kartushina, Natalia; Martin, Clara D.	Third-language learning affects bilinguals' production in both their native languages: A longitudinal study of dynamic changes in L1, L2 and L3 vowel production
52	journalArticle	2019	Weintrop, David; Wilensky, Uri	Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms
53	journalArticle	2017	Stevens, Kimberly A.; Ronan, Prof Kevin; Davies, Gene	Treating conduct disorder: An effectiveness and natural language analysis study of a new family-centred intervention program

54	journalArticle	2020	Santana, Carlos Ariel / Coni	Un metalenguaje de programación orientado al diseño de interfaces gráficas
55	journalArticle	2018	Chen, Zhifei; Chen, Lin; Ma, Wanwangying; Zhou, Xiaoyu; Zhou, Yuming; Xu, Baowen	Understanding metric-based detectable smells in Python software: A comparative study
56	journalArticle	2019	Donnelly, Lane F.; Grzeszczuk, Robert; Guimaraes, Carolina V.; Zhang, Wei; Bisset III, George S.	Using a Natural Language Processing and Machine Learning Algorithm Program to Analyze Inter-Radiologist Report Style Variation and Compare Variation Between Radiologists When Using Highly Structured Versus More Free Text Reporting
57	journalArticle	2020	Pérez Narváez, Hamilton Omar / Roig Vila	Uso de SCRATCH en el aprendizaje de Programación en Educación Superior
58	journalArticle	2020	Champredonde, Raúl	Visual davinci : Lenguaje y ambiente visual para la enseñanza de programación en el curso de ingreso y en el primer año de las carreras de Informática
59	journalArticle	2020	Brito, Y. S. Pellicer / M. C. Zea / R. A. Pérez / Y. C. Blanco / M. G. L.	Visualización dinámica, una opción para la enseñanza aprendizaje de la programación de computadoras

Tabla 2 Resultado de la revisión

1.2. Antecedentes Conceptuales de la Investigación.

1.2.1. Hipótesis de la investigación

Una metodología de selección, permitirá determinar el lenguaje de programación idóneo para el desarrollo de software en un ambiente de enseñanza-aprendizaje presencial.

El proceso de implementación de una metodología de selección de lenguaje de programación, a través de la evaluación de la misma, permite validar la hipótesis de la investigación. Este proceso se realiza en dos etapas. La primera sin usar una metodología y la segunda con la implementación de la misma. De esta manera permite cotejar resultados previos y post implementación de la metodología.

Por otro parte, se plantean hipótesis específicas tales como:

HE1 Los lenguajes de programación más usados en base a tendencias de programación actuales

HE2 Instrumentos de recolección de información más idóneos para el estudio

HE3 Parámetros de evaluación adecuados para la metodología

1.2.2. Red de categorías de las variables

El estudio plantea variables dependientes, en primer plano están las tendencias de programación actuales y a continuación la aceptación por parte del estudiante. Es importante mencionar que en el primer punto se basa en revisar páginas de índices de desarrolladores a nivel global para determinar las tendencias de programación y en el segundo punto el determinar el grado de aceptación por parte del estudiante. En ambos casos se busca obtener resultados cualitativos para el estudio y obtener un mejor análisis.

Además, se plantea la variable independiente enfocada a los lenguajes de programación orientados a objetos idóneos para el desarrollo de software.

1.2.3. Fundamentación teórica de la variable Independiente

1.2.3.1. Software

El software se lo puede definir como el conjunto de instrucciones a ser usadas por el computador. Actualmente el software se puede encontrar en la mayoría de dispositivos electrónicos y pudiendo existir de forma intangible con un conjunto de líneas de código no atados a ningún medio en específico. Entre la producción de software se destacan los sistemas operativos, software de gestión, software empresarial, software de seguridad, entre otros.

Los programas o software para un computador o dispositivo móvil, es un producto intangible que en su interior lleva un conjunto de instrucciones secuenciales lógicas, que cumplen una determinada función. El software tiene gran acogida y alcance, ya que los avances tecnológicos y automatización de procesos se generan a diario a nivel mundial.

Algunos años atrás, jamás se hubiera podido imaginar que el software se convertiría en una industria indispensable para los ámbitos personales y empresariales. Las aplicaciones tan grandes que recibe el software hoy en día en las ciencias, ingenierías, y que permiten la creación de tecnologías nuevas, por ejemplo, ingeniería genética y nanotecnología [18].

Hoy en día el software forma parte indispensable de los dispositivos que usan para acceder a información, y qué personas los usan en la cotidianidad [5].

1.2.3.2. Desarrollo de Software.

El crecimiento de empresas y la necesidad de automatizar procesos ha convertido el software en un elemento de transformación y crecimiento en el ámbito productivo. Organismos públicos señalan que la producción de aplicaciones en Ecuador ha crecido paulatinamente con el paso de los años.

Un estudio realizado por la Asociación Ecuatoriana de Software AESOFT, en el año 2012 indica que la concentración de carreras por instituto es mucho menor que en las y computación, por ese entonces se los muestra en la Tabla 3.

No.	Instituto	No. carreras disponibles por institución	% sobre el total
1	Instituto Tecnológico Superior Espiritu Santo	11	4%
2	Instituto Tecnológico Superior Sudamericano	10	4%
3	Instituto Tecnológico Superior Ibarra	8	3%
4	Instituto Tecnológico Superior Integración Andina	7	3%
5	Instituto Tecnológico Superior Los Andes	6	2%
6	Instituto Tecnológico Superior Raul Prebisch	6	2%
7	Instituto Tecnológico Superior Cordillera	5	2%
8	Instituto Tecnológico Superior Liceo Cristiano	5	2%
9	Instituto Tecnológico Superior Vicente Rocafuerte	5	2%
10	Instituto Tecnológico Superior Almirante Illingworth	4	1%
11	Instituto Tecnológico Superior David P Ausubel	4	1%
12	Instituto Tecnológico Superior Edupraxis	4	1%
13	Instituto Tecnológico Superior El Pacífico	4	1%
14	Instituto Tecnológico Superior Esca	4	1%
15	Instituto Tecnológico Superior Japon	4	1%
16	Los demás	190	69%

Tabla 3 Institutos Carreras tecnológicas Fuente: Sniese. En línea. Último ingreso: abril 2011

A nivel de Institutos Superiores es evidente, una mayor concentración en la oferta de carreras de análisis y programación, los cuales representaban el 53% de un total de las

No.	Carrera	%
1	Análisis de sistemas	35%
2	Programación de sistemas	18%
3	Informática	10%
4	Informática mención análisis de sistemas	5%
5	Informática mención programación de sistemas	3%
6	Diseño gráfico	2%
7	Informática empresarial	1%
8	Informática: programación y análisis de sistemas	1%
9	Sistemas de automatización	1%
10	Análisis de sistema	1%
11	Analista de sistemas	1%
12	Analistas de sistemas	1%
13	Diseño multimedia	1%
14	Informática para el control aeronáutico	1%
15	Informática y programación de gestión	1%
15	Las demás	19%

carreras que se ofertan a nivel nacional. A esto además se evidencia la demanda de carreras informáticas más ofertadas por los institutos superiores, en

referencia al porcentaje sobre el total que se muestra en la Tabla 4.

Tabla 4 Demanda Carreras Fuente: Sniese. En línea. Último ingreso: abril 2011

Toda la información estadística expuesta, se debe tener en cuenta factores como avances tecnológicos que con el pasar de los años se cierran carreras y por otra parte se crean nuevas o se rediseña y mejora las existentes. Un claro ejemplo es la carrera de Análisis de Sistemas,

la cual actualmente se encuentra en proceso de cierre de la misma. A esto añadir que el cerrar esa carrera ha dado apertura a rediseñar y crear una nueva como la Desarrollo de Software. En la actualidad se destacan las carreras de Analistas de Sistemas y los Institutos Tecnológicos Superiores se mantienen con la visión de explotar el desarrollo de software ofertando la carrera de Tecnología Superior en Desarrollo de Software.

En las sesiones de laboratorio de desarrollo de software se enseña un lenguaje de programación, estableciendo problemas a pequeña escala, comenzando con problemas fáciles y gradualmente volviéndose más complejo [12].

1.2.3.3. Lenguajes de Programación

Es un lenguaje formal, interpretado como un conjunto de instrucciones con una combinación de reglas de sintaxis y semántica, que permiten indicar órdenes que deseamos que realice una computadora de manera específica [19].

Las computadoras y la forma de programarlas han evolucionado de una forma vertiginosa con el paso del tiempo [13]. En la actualidad existen diversas herramientas de desarrollo de software y cuando se habla de lenguajes de programación en específico, que es el tema central de estudio de esta investigación, la lista aún sigue siendo muy extensa. Hay que reconocer que los lenguajes de programación que forman parte de esta lista se enfocan en diferentes ámbitos de trabajo ya que cada uno tiene su mayor afinidad con un campo de acción específico para solventar las necesidades tanto de los usuarios como de los desarrolladores.

En general, el aprendizaje de un lenguaje de programación, puede tornarse una tarea no tan sencilla pues inmiscuye el análisis, razonamiento y lógica del paradigma de programación en el que destaca la sintaxis y semántica del innato del lenguaje [15].

Las computadoras y la forma de programarlas han evolucionado de una forma vertiginosa con el paso del tiempo [13]. Se debe entender que hoy en día, no se ha llegado a un consenso en cuanto a un método para utilizar a la hora de programar computadoras. Esto se debe a que no existe un único método que permita la resolución de un algoritmo, así como unos lineamientos didácticos para ingresar al mundo de la programación de computadoras [20].

Es esencial destacar que los métodos de enseñanza actualmente se fundamentan en los paradigmas de programación como son el funcional y el imperativo. Dentro de un paradigma podemos evidenciar que enseñan a programar en un lenguaje en específico, aplicando el uso de reglas de sintaxis o semántica. Se debe mencionar que cada lenguaje tiene su aporte desde diferente enfoque, puesto que cada uno tiene sus propias características y funcionalidades para determinada área de análisis y desarrollo [21].

Adentrados en el contexto, podemos encaminar la selección de los lenguajes de programación acorde a una metodología que permita esta selección con base en parámetros técnicos y análisis de los mismos. Cabe indicar que la investigación se centra en la metodología de selección de lenguajes de programación para desarrollar software, particularmente en los que se encuentran enfocados en la programación orientada a objetos. Esto permite limitar la investigación de manera significativa y analizar estos lenguajes, enfocado en la aplicación y uso de los mismo en un ambiente de enseñanza-aprendizaje [22].

La enseñanza de los principios básicos del uso de algoritmos se lo lleva a cabo en primera

instancia en la Asignatura Introducción al Desarrollo de Software, esta asignatura se la imparte en el primer ciclo y tiene como objetivo que el alumno aborde el tema de resolución de problemas con el uso de un algoritmo, ya que el estudiante tenga desarrollado bien la resolución algorítmica, se procede a traducirlo hacia un lenguaje de programación específico, en este punto hay que recalcar que el lenguaje de programación viene dado por tradición de uso, por experiencia del docente o por otras circunstancias atenuantes; pero que no están basados implícitamente en una metodología de selección o lineamientos técnicos que determinen de manera eficiente el lenguaje de programación óptimo para un proceso de aprendizaje [23].

1.3. Fundamentación teórica de la variable Dependiente

1.3.1. Programación Orientada a Objetos.

Se expresa como un paradigma de programación en la cual se ven inmiscuidos términos como clases, objetos; es decir se asemejan a expresar cosas de la vida real. Para reutilizar el código de programadores y puedan ser usados por otros desarrolladores se vio necesario el crear la POO [24]. Por otra parte, no dejar de mencionar los pilares fundamentales de la POO objetos como son: herencia, polimorfismos, encapsulamiento y abstracción, los cuales forman parte esencial de la investigación para establecer la metodología de selección y se presenta a continuación en la Ilustración 6.

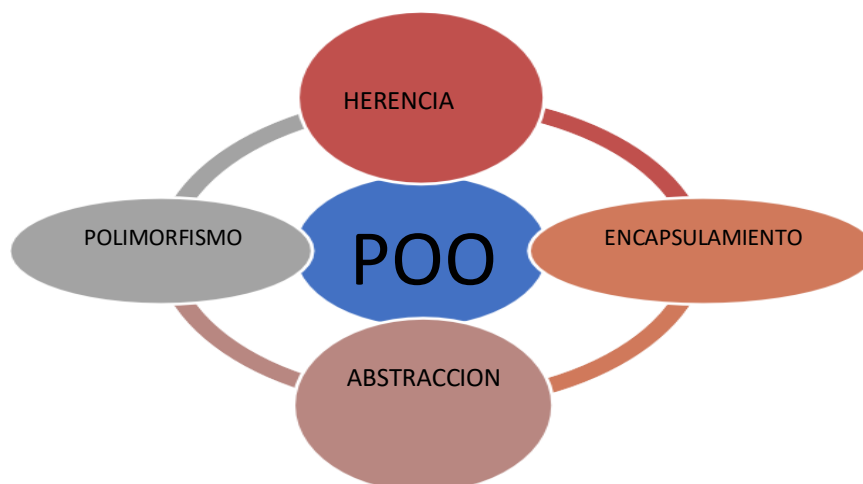


Ilustración 6 Fundamentos de la Programación Orientada a Objetos (POO)

1.3.1.1. Herencia.

Este concepto se expresa en que una clase hereda las funciones y características de otra existente.

1.3.1.2. Encapsulamiento.

Este concepto se aplica al aislar todo del resto de componentes y puede pasarse por desapercibido.

1.3.1.3. Abstracción.

Guarda relación con el encapsulamiento, representa sus características exteriores, pero aísla su complejidad del resto.

1.3.1.4. Polimorfismo.

Este concepto guarda el uso de varios objetos de diferentes clases, pero guardan una base en común, y el programador lo puede usar de diversas maneras [25].

En el transcurso de los últimos años, los autores eligen y encaminan el uso de POO como paradigma de enseñanza en las carreras del ámbito informático [26].

1.3.2. Lenguajes de Programación Visuales (VPL) y lenguajes de programación textuales(TPL)

1.3.2.1. Lenguajes de Programación Visuales (VPL) y lenguajes de programación textuales (TPL) consisten en elementos icónicos que pueden interactuar manipulado de acuerdo con alguna gramática espacial [27].

Varios estudios que comparan VPL y TPL muestran que no hay conclusiones que pongan en evidencia sobre sus ventajas relativas. Sin embargo, generalmente se admitió que los VPL son más productivos y motivadores para los principiantes. Por otro lado, los TPL son considerablemente más productivos para tratar problemas complejos y a gran escala y, de hecho, la mayoría de los idiomas son TPL [28].

Las asignaturas que involucran el proceso de programación, generalmente son impartidas con el uso de lenguajes de propósito general que resultan bastante complejos y más aún cuando se inicia un proceso de enseñanza-aprendizaje [29]. Algunos lenguajes de programación se centran en que un estudiante debe aprender varios conceptos teóricos para adentrarse en el mundo de la programación y por otra parte los lenguajes se enfocan en que un estudiante debe escribir numerosas cantidades de líneas de código.

Algunos autores plantean usar entornos de programación con interfaces gráficas debido a que reducen carga cognitiva necesaria para realizar tareas de programación [30]. La hipótesis de partida es una metodología de selección, permitirá determinar el lenguaje de

programación idóneo para el desarrollo de software en un ambiente de enseñanza-aprendizaje.

Se busca proponer una metodología centrada en la programación Orientada a objetos la cual permita el generar un proceso de enseñanza aprendizaje a lo largo del estudio de la carrera de desarrollo de software de manera efectiva. A través del uso de lenguajes de programación idóneos y acordes a solucionar problemas de manera efectiva con el uso de un lenguaje de programación.

1.3.3. Selección de lenguajes de programación

Un panorama claro se lo puede mencionar en las instituciones de educación superior en el país de México, los que han tomado un fuerte enfoque hacia la enseñanza de lenguajes de programación como Java, C, C++ o C#. Mencionados lenguajes son muy trascendentales a la hora de construir software de producción, pero que realmente no fueron diseñados para empezar un proceso de aprendizaje de un lenguaje de programación. Esto ha provocado que muchos estudiantes tengan la concepción de que la informática es una disciplina compleja y con un gran nivel de dificultades técnicas [31].

Otro claro ejemplo del uso de lenguajes de programación en un proceso de enseñanza aprendizaje es en España que plantean a Python como el primer lenguaje de programación para arrancar un proceso de adentrarse en el mundo de la programación [9], [32].

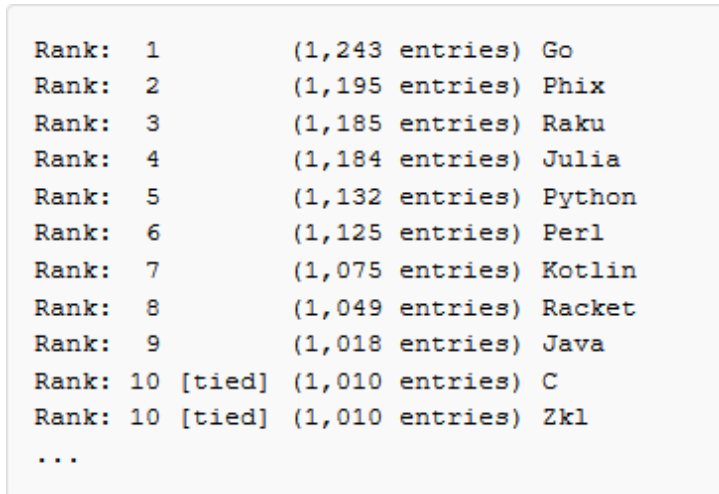
Existen, en la actualidad, diversos lenguajes de programación que permiten evidenciar el proceso de enseñanza aprendizaje a través del paradigma de orientación a objetos. Los lenguajes de programación guardan cada uno de ellos sus funcionalidad y aplicación en diferente ámbito del desarrollo de software, y lo que se pretende con el estudio, es el establecer una metodología para seleccionar el o los lenguajes idóneos para mejorar este proceso [33].

En la industria del software, están disponibles rankings que permiten determinar, los lenguajes de programación más populares utilizados para el en el desarrollo de software, estos rankings, se establecen a partir de información en la Web y basado en parámetros técnicos de análisis [34].

A todo esto, se plantea interrogantes tales como: ¿Cuál es el mejor lenguaje de programación? Preguntas sobre lenguajes de programación y las propiedades de sus

programas se solicitan con frecuencia, pero no se obtienen respuestas bien fundadas fácilmente disponibles [35].

Una base de selección para aplicar es Rosetta Code, el cual se presenta como un repositorio de código con una interfaz wiki. Este estudio se basa en una instantánea del repositorio tomada el 24 junio de 2020 y se presenta en la Ilustración 7 a continuación [36].



Rank: 1	(1,243 entries)	Go
Rank: 2	(1,195 entries)	Phix
Rank: 3	(1,185 entries)	Raku
Rank: 4	(1,184 entries)	Julia
Rank: 5	(1,132 entries)	Python
Rank: 6	(1,125 entries)	Perl
Rank: 7	(1,075 entries)	Kotlin
Rank: 8	(1,049 entries)	Racket
Rank: 9	(1,018 entries)	Java
Rank: 10 [tied]	(1,010 entries)	C
Rank: 10 [tied]	(1,010 entries)	Zkl
...		

Ilustración 7 Ranking Popularidad Lenguajes Fuente: Rosetta Code/Rank Languages by popularity

Por otra parte, también se puede tomar otra fuente de ranking de selección la comunidad de programadores: TIOBE, el cual es un indicador que determina la popularidad de los lenguajes de programación actuales. Es de mencionar que este índice se actualiza una vez por mes y todo esto se basa en ingenieros, cursos y desarrolladores externos de todo el mundo.

El tomar como referencias índices de clasificación a nivel global permite tener una perspectiva general del uso de lenguajes de programación y usar los lenguajes de programación en base a tendencias de desarrollo actuales. Estos rankings presentados permiten obtener criterios de diversos profesionales en el ámbito de desarrollo a nivel mundial y fortalece las bases del proyecto para construir una metodología mucho más robusta y eficiente a la hora de seleccionar un lenguaje de programación.

Los lenguajes de programación se presentan como una larga lista en la cual destacan el uso o funcionalidad específica de cada uno, pero esto es el punto de partida para establecer los lineamientos base para una metodología que determine el lenguaje a usar en un proceso de enseñanza aprendizaje. De esta forma el encaminar esfuerzos al iniciar y seguir un proceso

de aprender un lenguaje de programación acorde a las necesidades del mercado y tomando en cuenta el motivar el estudio de la carrera de desarrollo de software. Con el uso de herramientas tecnológicas y tendencias de desarrollo basadas en el constante cambio del manejo de información automatizada.

Es interesante el conocer que varios lenguajes de programación incorporan en su marco de trabajo, frameworks que aportan significativamente al desarrollador [37].

Otro dato importante a mencionar es la popularidad, soporte de análisis de datos, volumen de datos que puede manejar, velocidad de compilación, expresividad, recomendaciones de los programadores y costo financiero razonable promedio. Los lenguajes de programación Python, R, Java, SQL, Scala y C se utilizan como alternativas y a continuación se presenta en la Ilustración 8 [38].

Nov 2020	Nov 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.21%	+0.17%
2	3	▲	Python	12.12%	+2.27%
3	1	▼	Java	11.68%	-4.57%
4	4		C++	7.60%	+1.99%
5	5		C#	4.67%	+0.36%
6	6		Visual Basic	4.01%	-0.22%
7	7		JavaScript	2.03%	+0.10%
8	8		PHP	1.79%	+0.07%
9	16	▲	R	1.64%	+0.66%
10	9	▼	SQL	1.54%	-0.15%
11	14	▲	Groovy	1.51%	+0.41%
12	21	▲	Perl	1.51%	+0.68%
13	20	▲	Go	1.36%	+0.51%
14	10	▼	Swift	1.35%	-0.31%
15	11	▼	Ruby	1.22%	-0.04%
16	15	▼	Assembly language	1.17%	+0.14%
17	19	▲	MATLAB	1.10%	+0.21%
18	13	▼	Delphi/Object Pascal	0.86%	-0.28%
19	12	▼	Objective-C	0.84%	-0.35%
20	32	▲	Transact-SQL	0.82%	+0.44%

Ilustración 8 Índice Tiobe, ranking noviembre 2020 Fuente: TIOBE index ranking 2020

Para el desarrollo de software es fundamental contar con varias alternativas de herramientas que hagan efectiva su utilización y sean sencillos de aprender [39]. Además, es valioso realizar una comparativa de lenguajes de programación, basado en parámetros de análisis expuestos en un artículo científico en el cual destaca la comparación entre C, C++, C# y java [40].

1.3.4. Parámetros para elaborar la metodología.

La teoría del aprendizaje se utiliza para guiar a los educadores y desarrolladores de software educativo para comprender el enfoque de aprendizaje dirigido que se puede adoptar y adaptar en un desarrollo de software [41].

Partir de estudios que guarden relación con los lenguajes de programación orientados a objetos, permite un estudio comparativo y de análisis. El cual se centra en información sacada de índices Tiobe, GitHub y PYPL, los cuales aportan a este trabajo para determinar resultados efectivos [34].

Un aporte muy importante en el estudio, se establece en base a criterios para seleccionar un lenguaje de programación para cursos de principiantes, el cual define algunos lineamientos técnicos que aportan a la creación de la metodología que se propone [39].

Con base en este estudio se analizará y se buscará establecer los posibles criterios para la selección de un lenguaje de programación, considerando parámetros como se describen a continuación en la Tabla 5.

Costo financiero razonable

Disponibilidad académica del estudiante

Aceptación Académica

Disponibilidad de libros / material Bibliográfico

Etapas de ciclo de vida

Aceptación de la industria

Comercialización

Requisitos de sistema

Dependencia del Sistema Operativo

Propietario / Código abierto

Entorno de Desarrollo

Facilidad de depuración

Facilidad de aprendizaje de conceptos

fundamentales

Soporte para código seguro

Funciones avanzadas para cursos de

programación

Manejo de Scripts

Soporte para desarrollo web

Soporte para desarrollo móvil

Apoyo al enfoque de enseñanza

Soporte orientado a objetos

Disponibilidad de soporte

Instructor y entrenamiento de personal

Nivel de experiencia anticipado

Tabla 5 Criterios de Selección

Los criterios fueron derivados de analizar más de sesenta documentos relevantes para la selección del idioma y justificado por una breve revisión del apoyo literario en cada uno de los criterios [42]. En este contexto se busca el analizar cada uno de los lineamientos de manera técnica para identificar el enfoque de cada uno de ellos y permita determinar el diseño de una metodología de selección de lenguajes de programación.

A continuación, se presenta la descripción de los criterios de manera más específica:

Costo financiero razonable. – El primer criterio enmarca el determinar el precio para adquirir el lenguaje de programación y el entorno de desarrollo en el que se maneja el mismo.

Disponibilidad académica del estudiante. – Establecer si existe una versión de la herramienta enfocada al ámbito educativo, mas no a la puesta en producción.

Aceptación Académica. – Esto se enfoca en establecer la popularidad de un lenguaje en otras instituciones académicas.

Disponibilidad de libros / material Bibliográfico. – Esto determina si un lenguaje posee la suficiente información de referencia ya sea formato digital o físico para usarla.

Etapas de ciclo de vida. – Tener presente conceptos que afectan el crecimiento, maduración y descomposición de los lenguajes de programación con el pasar de los años.

Aceptación de la industria. – Tomar en cuenta La penetración que tiene o puede tener un lenguaje de programación a nivel de los negocios del medio actuales [43].

Comercialización. – Permite entender la empleabilidad de los graduados que se desenvuelven con el uso de el o los lenguajes de programación en determinado ambiente laboral.

Requisitos de sistema. – Es indispensable tener en cuenta los requisitos a nivel de hardware como de software para la implementación de un sistema en determinado equipo.

Dependencia del Sistema Operativo. – A tomar en cuenta siempre la dependencia de un lenguaje de programación en cuanto al sistema operativo, con respecto temas de compatibilidad y desarrollo.

Propietario / Código abierto. – Específicamente esto se basa en la responsabilidad y sobre todo la evolución de los lenguajes de programación y su ambiente de desarrollo.

Entorno de Desarrollo. – El entorno de trabajo con el que cuenta el programador, desde un simple editor de texto hasta un Entorno de Desarrollo Integrado. (IDE).

Facilidad de depuración. – Entender las ventajas que ofrece un lenguaje a la hora de presentar y entender código para ser depurado.

Facilidad de aprendizaje de conceptos fundamentales. – Entender la curva de aprendizaje y

evolución de un lenguaje de programación.

Soporte para código seguro. – Establecer los lineamientos de seguridad a la hora de codificar un programa y que brinde las seguridades en tiempo de diseño y ejecución.

Funciones avanzadas para cursos de programación. – Evidenciar que un lenguaje de programación tiene las prestaciones nuevas para generar soportar un curso avanzado de programación.

Manejo de Scripts. – El uso de lenguajes completos en contenido y menos complejos a la hora de insertarlos en un ambiente de aprendizaje.

Soporte para desarrollo web. – El criterio se centra en una de las tendencias de desarrollo actual como son aplicaciones web y el nivel de soporte de lenguaje para este tipo de aplicación.

Soporte para desarrollo móvil. – Al igual que el criterio anterior, es fundamental que el lenguaje brinde prestaciones y soporte para el desarrollo de aplicaciones móviles.

Apoyo al enfoque de enseñanza. – Es válido el tener presente que un lenguaje de programación apoye el enfoque de enseñanza de conceptos básicos de manera clara, sencilla y efectiva para los estudiantes.

Soporte orientado a objetos. – En este criterio es evidente la necesidad de aplicación de conceptos de POO como son abstracción, polimorfismo, herencia y encapsulamiento.

Disponibilidad de soporte. – Este criterio determina el saber la disponibilidad de soporte a nivel local, dentro de la institución y soporte a través de foros, o soportes de documentación de la herramienta y profesionales del ámbito [42].

Instructor y entrenamiento de personal. – En este criterio se describe la capacitación de los docentes o instructores, personal de soporte técnico necesario para aprender el lenguaje de programación en específico [44].

Nivel de experiencia anticipado. – En este criterio permite el establecer los conocimientos previos de un estudiante en un lenguaje y las bases que conoce para facilitar mejor las comprensiones [45].

El proceso de enseñanza-aprendizaje de los lenguajes de programación no es del todo una actividad sencilla para los estudiantes al igual que para los docentes [8]. Entre los pilares clave para la introducción a la enseñanza de los lenguajes de programación, en primer lugar,

está el lenguaje escogido; además otro elemento la malla curricular, contenidos mínimos y las herramientas usadas durante el proceso [46].

1.4. Antecedentes Contextuales de la Investigación.

1.4.1. Delimitación del contexto de estudio.

El estudio se realiza en las provincias de Azuay y Cañar. Directamente en los Institutos Superiores Tecnológicos, los cuales ofrecen la carrera de Desarrollo de Software. Además, el enfoque de puesta en marcha de la Metodología en asignaturas relacionadas directamente con el desarrollo de software, las cuales son: Introducción al Desarrollo de Software, Fundamentos de Programación y Programación Orientada a Objetos [47]. Posterior a esto será evaluada por expertos con experiencia en el ámbito de desarrollo de software.

1.4.2. Propuesta de solución y contribuciones.

Lo interesante de la propuesta es desarrollar una metodología de selección de los lenguajes de programación para desarrollo de software, aplicado a un ambiente de enseñanza-aprendizaje presencial para los institutos de educación superior de las provincias de Azuay y Cañar. Permitiendo entender cuán importante es realizar este estudio. De esta forma aportar a fortalecer las falencias de los estudiantes a la hora de programar acorde a una buena selección del lenguaje de programación, enfocado a estudiantes que se inician en el proceso de adentrarse en el mundo de la programación [46].

A todo esto, se suma que el estudio tendrá como beneficiarios directos los estudiantes y docentes de los Institutos Superiores Tecnológicos de Azuay y Cañar, mismos que al contar con una metodología de selección, permitirá el fortalecer el proceso de enseñanza-aprendizaje con el uso de lenguajes de programación idóneos. A esto se suma que preparan profesionales de manera más efectiva para la inserción en el ámbito laboral [48].

Este análisis permite extraer información que aporte en el proyecto buscando el definir parámetros más concretos y efectivos a la hora de construir una metodología de selección [31]. Para la evaluación de la metodología se aplica el método Delphi el cual se muestra como un proceso prospectivo en la cual intervienen un grupo de expertos [49].

1.4.3. Organización del documento.

El documento se encuentra estructurado de la siguiente forma, en base a cuatro capítulos, en primer lugar, en el capítulo uno se aborda el análisis bibliográfico antecedentes

históricos, conceptuales, contextuales fundamentos teóricos de la investigación de los cuales se parte para argumentar científicamente el estudio. El capítulo dos presenta las técnicas de recolección de información a través de encuestas, entrevistas y la metodología de investigación. El tercer capítulo define la relevancia de información recolectada, y la propuesta del desarrollo de la metodología de selección. En el cuarto capítulo se establecen una comparativa y se contrasta resultados; además de las conclusiones y recomendaciones de la investigación.

Capítulo 2

2. Generalidades.

En este capítulo se describe detalladamente la metodología y materiales utilizados para la realización del presente estudio. Explicando además cada uno de los procedimientos para la recolección, análisis y tabulación de datos.

2.1. Tipo de estudio o investigación realizada.

La elección del tipo de estudio a la hora de realizar una investigación es una de las etapas más complejas en el proceso investigativo. Puesto que en esto se debe tomar en cuenta la información previa al respecto existente sobre el tema. En este contexto el estudio se enfoca en el ámbito experimental y correlacional, es decir permitirá el evaluar la metodología de selección de lenguajes de programación en un proceso de enseñanza-aprendizaje. A más de esto, se suma el contrastar los resultados obtenidos antes de implementar la metodología y otra posterior a realizarsu implementación.

2.2. El paradigma o enfoque en el cual se realizó.

Importante en este hito el definir el enfoque mixto que toma el estudio a la hora de analizar los datos recolectados. De esta forma se establece una visión clara a la hora de poner en práctica el enfoque idóneo en la población especificada que se muestra en la Ilustración 9.



Ilustración 9 Proceso de evaluación

2.3. Diseño de investigación mixto (Cualitativo-Cuantitativo)

Se desarrollo el diseño cuantitativo con el fin de recolectar y analizar los datos numéricos. Con la finalidad de probar la hipótesis con base en medidas numéricas y el análisis estadístico. Este diseño es ideal, que permite identificar tendencias, comprobar relaciones y obtener resultados generales. Todo esto con el objetivo de definir pautas de

comportamiento a la hora de implementar una metodología de selección de los lenguajes de programación para desarrollo de software. Con el fin de aplicarlo a un ambiente de enseñanza-aprendizaje presencial para los institutos de educación superior de las provincias de Azuay y Cañar. El enfoque cuantitativo permitirá entender e interpretar el nivel de factibilidad y eficiencia de la misma.

Al llevar a cabo un diseño de investigación mixto, en el presente estudio también se centra en un enfoque cualitativo para evaluar, ponderar e interpretar la información que se obtuvo de los diferentes recursos tales como entrevistas, encuestas, revisión bibliográfica, con el objetivo de establecer un significado trascendente.

2.4. Cálculo de la población y muestra.

Aplicando la calculadora de la muestra a través de las funciones de Excel, se obtienen los siguientes resultados. Partiendo que se tiene una población aproximada de 100 estudiantes de los 5 Institutos de Educación Superior, tres de la provincia de Azuay y dos de Cañar. Tomados de la carrera de Desarrollo de Software.

- ✓ Con un nivel de confianza del 95%
- ✓ Un margen de error de 5%
- ✓ Una población aproximada de 100

Se obtuvo el siguiente resultado expuesto en la Ilustración 10.

MARGEN DE ERROR MÁXIMO ADMITIDO	3,0%
TAMAÑO DE LA POBLACIÓN	100
Tamaño para un nivel de confianza del 95%	92
Tamaño para un nivel de confianza del 97%	93
Tamaño para un nivel de confianza del 99%	95
Volver a página de inicio	

Ilustración 10 Cálculo de la muestra.

De esta forma se puede evidenciar que la población total a ser encuestada para un óptimo resultado de recolección y tabulación está en los 92 a 95, que varían en el nivel de confianza.

Se debe además mencionar que se procederá a realizar la entrevista a los directores de la carrera de desarrollo de software de los cinco institutos superiores tecnológicos de Azuay y Cañar. Sumado a esto se procederá con la evaluación de la metodología por parte de expertos con amplia experiencia en el ámbito de desarrollar software.

2.5. Métodos teóricos con los materiales utilizados.

En el proyecto de investigación, ha habido avances que contribuyen a la ciencia desde diferentes ángulos. La investigación tiene un impacto importante en la sociedad, pero si la investigación no lleva a cabo procesos adecuados de recopilación y análisis de datos, no tendrá tal impacto. Además, es principalmente el proceso de realización de métodos y sistemas.

En la actualidad, existe una gran cantidad de métodos, tecnologías y medios que pueden ayudar a completar la ardua tarea de la recolección de datos. Por la diversidad de datos que se pueden obtener en los diferentes estudios realizados. Comprender los métodos más relevantes en el campo de la investigación puede determinar si la investigación tiene un impacto en la comunidad científica.

Para esta investigación, se recomiendan algunas técnicas para recolectar información trascendental, que se enfoca en la participación de docentes, estudiantes y orientación académica.

Estas técnicas se muestran en la siguiente Ilustración 11:

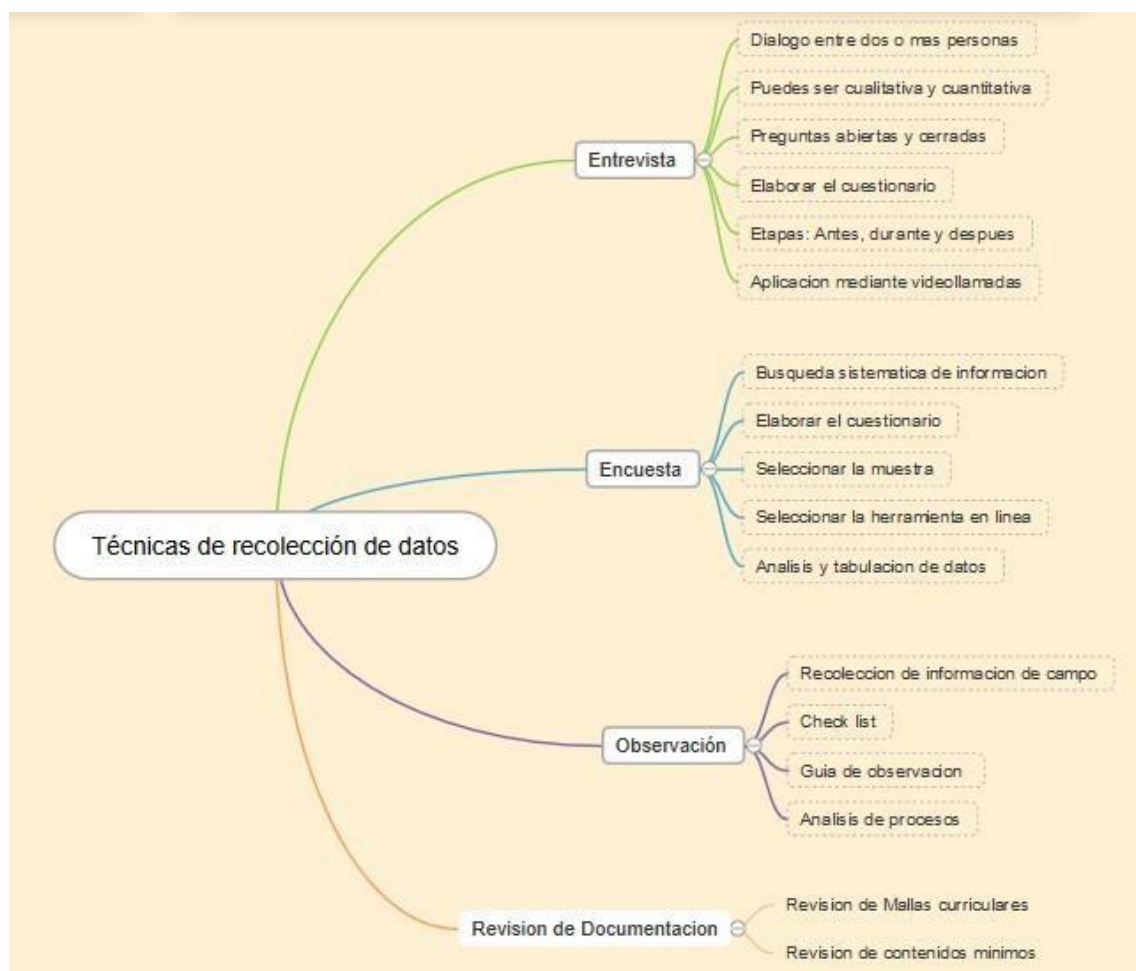


Ilustración 11 Técnicas de recolección de datos

Luego de la definición de técnicas e instrumentos de recolección de información, es fundamental dar a conocer las herramientas que se aplicarán para llevar a cabo los procesos correspondientes. En el caso de las entrevistas se las llevará a cabo en algunos casos de manera presencial y por otra parte a través de la plataforma Zoom. En cuanto a las encuestas se las realizará a través de la herramienta Google Forms para un mejor trato y tabulación de la información.

2.6. Métodos empíricos con los materiales utilizados.

El propósito de la revisión sistemática de la literatura (SRL) en este estudio es definir y evaluar la evidencia encontrada. RSL es un estudio secundario que utiliza un método determinista que permite la identificación, síntesis y correlación de la evidencia disponible relacionada con el objeto de investigación de una manera justa y repetible [33].

La búsqueda se realizó entre las publicaciones científicas indexadas en las bases de datos acorde a los lineamientos establecidos, dando resultados en los últimos 5 años. A continuación, se grafica el proceso llevado y se muestra en la Ilustración 12.

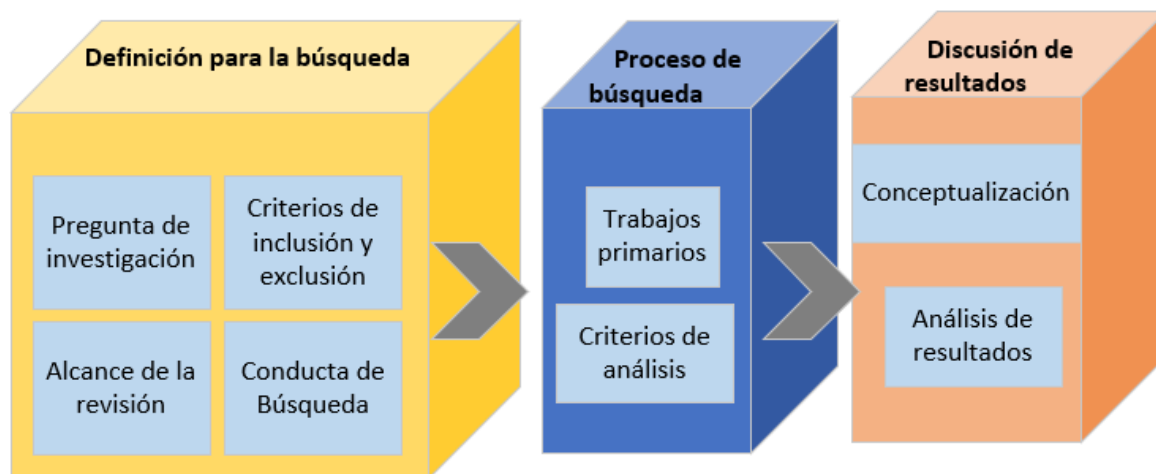


Ilustración 12 Proceso de mapeo.

2.6.1. Definiciones para la búsqueda.

Es fundamental el establecer las búsquedas que más aporten al estudio de manera significativa

2.6.1.1. Preguntas de investigación.

Para el presente estudio se plantean las siguientes preguntas de investigación.

RQ1. ¿Cómo realizar la selección de un lenguaje de programación orientado a objetos idóneo para el desarrollo de software?

RQ2. ¿Cuáles son los lenguajes de programación más usados según las tendencias de programación actuales?

RQ3. ¿Cuáles son los instrumentos de recolección idóneos, acorde a esta

investigación? RQ4. ¿Cuáles son los parámetros de evaluación para acorde a este estudio?

2.6.1.2. Alcance de la revisión

Este estudio se enfoca en realizar una Revisión Sistemática de la Literatura a través de varios motores de búsqueda que contienen librerías digitales.

Los parámetros para definir las cadenas de búsqueda fueron:

- ✓ Programming Languages + "for beginners"
- ✓ Programming Languages comparison.
- ✓ (Selección methodology) and Programming Languages + for beginners

Al tratarse de un tema el cual no se ajusta a un nicho de investigación previo, se plantearon tres cadenas de búsqueda, las cuales permitieron compactar diversos contenidos que aporten al objeto de estudio para la RSL.

Las cadenas de búsqueda se fueron ajustando a los parámetros de cada base de datos, con los cuales se obtuvieron de artículos 84%, conferencias 9%, libros 5%, tesis 2%. Estas investigaciones fueron tomadas en cuenta para el marco teórico del estudio y afianzar y respaldar la construcción y los lineamientos de la investigación.

2.6.1.3. Criterios de Inclusión y Exclusión.

Para filtrar la gran cantidad de estudios hallados, se realizó filtros en base a parámetros tales como:

- ✓ Se incluyeron estudios de los idiomas inglés y español.
- ✓ Se controló el rango de año de publicaciones entre 2015 y 2020, salvo casos excepcionales de libros y fundamentos teóricos necesarios. Exceptuando casos especiales de fundamentos teóricos importantes.
- ✓ Se tomaron en cuenta los estudios que guardan relación con la ingeniería de software, específicamente con el desarrollo de software.
- ✓ Se consideraron los estudios en base a las lecturas de abstract y conclusiones de la investigación.

2.6.1.4. Conductas de búsqueda.

Filtro Primario:

Revisión del título de los estudios encontrados y la relación con el objeto de estudio. Con los títulos seleccionados se procede a la revisión del abstract.

Filtro secundario:

Los estudios que aprobaron el primer filtro, ahora pasan a un proceso de análisis y lectura completa de su contenido.

2.6.2. Proceso de Búsqueda

2.6.2.1. Trabajos Primarios.

Posterior a la aplicación de los filtros primario y secundario, fueron definidos 87 estudios loscuales aportan con sus contenidos de manera diversa en la investigación.

2.6.2.2. Criterios de Análisis.

A todo este proceso de la RSL se suma criterios de análisis de los estudios tales como el año, para entrar en el contexto de la evolución de la temática planteada, de manera esencial en cuanto a los lenguajes de programación.

Entre otro de los aspectos se presenta esta el producto, el proceso y las personas que desempeñaron su esfuerzo para el logro eficiente de sus propósitos y éxito en las tareas.

El área de conocimiento fue un punto clave a la hora del proceso de búsqueda, se centró en las ciencias de la computación, en el área de ingeniería del software la cual conlleva el proceso de desarrollo de software.

2.6.3. Discusión de resultados

2.6.3.1. Conceptualización

Un resumen de los trabajos hallados, donde se puede entender que luego de seguir la RSL y contextualizar el objeto de estudio de esta investigación. Además, se puede evidenciar el enfoque de los estudios en cuanto a los lenguajes de programación y ambientes de aprendizaje.

2.6.3.2. Análisis de resultados

A continuación, se presenta la Tabla 6, en cuanto a los trabajos hallados clasificados por parámetros de búsqueda establecidos como son: Tipo de documento, año de publicación, autores, título y abstract. En concreto se logra obtener un total de 59 investigaciones para fortalecer y construir el marco teórico del estudio y se muestra parte de las investigaciones en la Tabla 6.

NUM	TIPO	AÑO	AUTORES	TITULO
1	journalArticle	2019	Arslan, Sibel; Öztürk, Celal	A Comparative Study of Automatic Programming Techniques
2	journalArticle	2020	Gmys, Jan; Carneiro, Tiago; Mel	A comparative study of high-productivity high-performance programming languages for parallel metaheuristics
3	book	2016	Feo, John T.	A comparative study of parallel programming languages: the Salishan problems
4	journalArticle	2018	Eichhorn, Helge; Cano, Juan Luis	A comparative study of programming languages for next-generation astrodynamics systems
5	conferencePaper	2015	Nanz, Sebastian; Furia, Carlo A.	A Comparative Study of Programming Languages in Rosetta Code
6	journalArticle	2016	Mondal, Manishankar; Roy, Cha	A comparative study on the intensity and harmfulness of late propagation in near-miss code clones
7	journalArticle	2015	Jordan, Howell; Botterweck, Go	A feature model of actor, agent, functional, object, and procedural programming languages
8	journalArticle	2016	Ferrara, Pietro	A generic framework for heap and value analyses of object-oriented programming languages
9	journalArticle	2019	Katz, Garrett E.; Davis, Gregory P	A programmable neural virtual machine based on a fast store-erase learning rule
10	journalArticle	2015	Bak, Christopher; Faulkner, Glyn	A Reference Interpreter for the Graph Programming Language GP 2
11	journalArticle	2019	Kang, Pilsung	Accelerating Stochastic Simulations on GPUs Using OpenCL
12	journalArticle	2019	Oga-Baldwin, W. L. Quint	Acting, thinking, feeling, making, collaborating: The engagement process in foreign language learning
13	journalArticle	2020	Carreño, Yolanda Soler Pellicer	AMBIENTE INTEGRADO DE VISUALIZACIÓN DE ESTRUCTURAS DE DATOS PARA LA ENSEÑANZA-APRENDIZAJE DE LA PRO

14	journalArticle	2018	Cabada, Ramón Zatarain; Estrad	An affective and Web 3.0-based learning environment for a programming language
15	conferencePaper	2019	Bany Abdelnabi, Ahmad A.	An Analytical Hierarchical Process Model to Select Programming Language for Novice Programmers for Data Analytic
16	conferencePaper	2015	Morocho, Villie; Colina-Morles,	Analysis of thermographic patterns using Open CV case study: A clinker kiln
17	journalArticle	2015	Salas-Zárate, María del Pilar; Alo	Analyzing best practices on Web development frameworks: The lift approach
18	journalArticle	2020	Luk, Gigi; Pliatsikas, Christos; Ro	Brain changes associated with language development and learning: A primer on methodology and applications
19	bookSection	2020	Qi, Zhenghan; Legault, Jennifer	Chapter Five - Neural hemispheric organization in successful adult language learning: Is left always right?
20	journalArticle	2019	Majd, Amirabbas; Vahidi-Asl, M	Code4Bench: A multidimensional benchmark of Codeforces data for different program analysis techniques

Tabla 6 Parcial de análisis de resultados obtenidos

2.7. Técnicas estadísticas para el procesamiento de datos obtenidos.

Análisis de Datos con Microsoft Excel.

Para la obtención de parámetros estadísticos que se planteen en el problema, se utiliza las funciones avanzadas de la herramienta para generar datos más claros y precisos.

Capítulo 3

3. Generalidades.

En este capítulo se describen los resultados obtenidos en el presente estudio realizado. Además, se fundamenta el aporte práctico y el proceso de elaboración.

3.1. Aplicación de técnicas de recolección de datos

Se define las técnicas idóneas que permiten recolectar y analizar los datos del estudio de manera eficiente.

3.1.1. Observación

Esta técnica permitió comprender y evidenciar el proceso de enseñanza-aprendizaje de un lenguaje de programación en los Institutos Superiores Tecnológicos de Azuay y Cañar. Con esta técnica no participante, se evidenció el proceso de enseñanza-aprendizaje. A través de la cual se pudo entender el proceso de manera más clara.

Se pudo constatar, además, a través de la observación, que el proceso de enseñanza-, inicialmente se identifican elementos tales como docente, recursos y estudiantes. Todo el proceso se realiza en un laboratorio de computación con todos los recursos a nivel de hardware y software. Entre los recursos se destaca el uso de herramientas como pizarrón, computadora, proyector y software específico de estudio. El proceso de enseñanza-aprendizaje tiene un enfoque teórico-práctico que inicialmente fortalece conceptos para posteriormente poner en práctica. Las asignaturas son impartidas por profesionales en el área de sistemas (Ingenieros de Sistemas) con experiencia en desarrollo de software. En todo este proceso se puede destacar la constante retroalimentación por parte del docente con los estudiantes a la hora de reforzar y adquirir nuevos conocimientos.

En cada uno de los institutos de educación superior, se pudo conocer el uso de los diferentes lenguajes de programación como software que imparten para el aprendizaje de para desarrollar aplicaciones. En el transcurso de las horas de clase, se pudo determinar el uso de diferentes lenguajes entre los que se puede destacar Java, Javascript y C#.

Se aplicó la técnica de la observación en cada uno de los escenarios de manera objetiva e imparcial de tal forma que no se influyó en el transcurso natural del proceso de enseñanza-aprendizaje de una determinada asignatura y sus contenidos.

Por otra parte, se contó de manera voluntaria y con la mayor predisposición de parte de

docentes responsables de impartir las cátedras de contenido teórico-práctico para el desarrollo de software. Cada proceso fue autorizado, guiado y supervisado por los respectivos coordinadores de las carreras de desarrollo de software y sus respectivos docentes de cada asignatura.

3.1.2. Revisión de documentación.

Como parte de este proceso se revisó las mallas curriculares de la carrera de desarrollo de software para entender las asignaturas que estudian el proceso de enseñanza aprendizaje para el desarrollo de software. Con el fin de determinar las asignaturas que involucran el proceso de desarrollo de software.

Luego de revisar y analizar las mallas curriculares, se presentan entre ellas gran similitud en cuanto a las asignaturas que se imparten en cada ciclo entre los institutos de educación superior. Posterior a este análisis se procedió a determinar la concordancia de asignaturas y generar una tabla. Esta tabla se enfoca en determinar las asignaturas candidatas que lleven como eje central de su contenido el proceso de desarrollo de software en un ambiente de enseñanza-aprendizaje.

La tabla 8 que se muestra a continuación parte del primer análisis de las mallas curriculares.

Asignatura	Periodo Académico	Instituto
Introducción al Desarrollo de Software	Primer Período	JBV/AZ/ENA
Fundamentos de Programación	Primer Período	JBV/AZ/ENA
Programación Orientada a Objetos	Segundo Período / Tercer Período	JBV/AZ/ENA/SUDA/IN AN

Programación Visual	Tercer Período	JBV/AZ/ENA
Programación de Aplicaciones Web	Cuarto Período	JBV/AZ/ENA/SUDA
Desarrollo de Aplicaciones Móviles	Cuarto Período	JBV/AZ/ENA
Tendencias Actuales de Programación	Quinto Período	JBV/AZ/ENA
Lenguajes de Programación	Segundo Período	INAN
Introducción a la Web	Quinto Período	INAN
Sistemas de Información	Segundo Período	SUDA
Programación de Interfaces Gráficas	Segundo Período	SUDA
Programación de Aplicaciones para Dispositivos Móviles.	Quinto Período	SUDA
Metodologías para resolver problemas informáticos	Primer Período	SUDA

Tabla 8 Asignaturas candidatas

Luego de filtrar una tabla con las asignaturas que incluyen el desarrollo de software en un ambiente de enseñanza-aprendizaje. Se procede a determinar las asignaturas que serán las adecuadas para el levantamiento de datos antes de la metodología y posterior a la implementación de la misma. Quedando la estructura de las asignaturas candidatas definitivas de la siguiente manera como indica la Tabla 9.

<u>Asignatura</u>	<u>Instituto</u>	<u>Periodo</u>
Fundamentos de Programación	Azuay (AZ)	Primer Período
	Juan Bautista Vásquez (JBV)	
	Enrique Noboa Arizaga (ENA)	
Programación Orientada a Objetos	Sudamericano (SUDA)	Tercer Período
	Integración Andina (INAN)	

Tabla 9 Asignaturas candidatas finales para aplicar al estudio.

3.1.2.1. Revisión de contenidos mínimos por asignatura.

Luego de haber realizado el análisis y filtrar las asignaturas candidatas para el estudio previo y posterior a implementar la metodología, la propuesta ahora es centrarse en los contenidos mínimos de las asignaturas en cuestión. Con este documento es esencial analizar los lineamientos que engloban los temas esenciales a tomar en cuenta para el estudio. Los contenidos mínimos de las asignaturas tomados del microcurrículo se presentan en la siguientes Tabla 10 y Tabla 11.

Nombre de la asignatura:	FUNDAMENTOS DE PROGRAMACIÓN
Campo de formación:	Adaptación e Innovación Tecnológica
Unidad de organización curricular:	Formación Básica
Número de período académico	1
Número total de horas de la asignatura:	186

Organización de aprendizajes por modalidad, número de horas destinadas a cada componente (Art. 15 y 47 del RRA)	Componente Docencia:	72
	Componente de Prácticas de Aprendizaje:	54
	Componente de Aprendizaje Autónomo:	60

b) OBJETIVO DE LA ASIGNATURA:

Objetivo general:

Aplicar de forma autónoma la lógica de programación para el desarrollo de algoritmos sencillos utilizando una sintaxis adecuada.

Objetivos específicos:

- 1. Analizar casos sencillos y plasmarlos en pseudocódigo de forma argumentada**
- 2. Desarrollar algoritmos de nivel bajo de complejidad aplicando lenguaje de programación**
- 3. Especificar instrucciones y sentencias que serán utilizadas en la programación de un problema específico de forma argumentada.**
- 4. Comprobar la validez de los resultados arrojados por los algoritmos escritos mediante pruebas de escritorio.**
- 5. Explicar la estructura de programación que se requiere para la construcción de aplicaciones a través del uso de la lógica de desarrollo, de forma correcta.**

c)	<p>RESULTADOS DE APRENDIZAJE</p> <ul style="list-style-type: none"> - Analiza los problemas planteados y los soluciona a través de pseudocódigos y diagramas de flujo. - Compara los diagramas de flujos con la resolución de problemas planteados. - Realiza sentencias e instrucciones de programación usando un lenguaje orientado a objetos. - Realiza pruebas de la funcionalidad de los algoritmos desarrollados. <p>Describe cómo se aplican las sentencias e instrucciones de programación en la resolución de problemas planteados.</p>
d)	<p>CONTENIDOS MÍNIMOS DE LA ASIGNATURA:</p> <ol style="list-style-type: none"> 1. Introducción a la programación por computadora. <ol style="list-style-type: none"> 1.1. Introducción a los Algoritmos. 1.2. Diagramas de flujo. 1.3. Pseudocódigo 1.4. Solución algorítmica y DFD para problemas con condiciones e iteraciones. 2. El lenguaje y su entorno integrado de desarrollo. <ol style="list-style-type: none"> 2.1. Introducción al lenguaje y a su entorno de desarrollo. 2.2. Estructura de un programa 2.3. Variables y constantes 2.4. Tipos de datos 2.5. Operadores aritméticos, lógicos y relacionales 2.6. Estructuras selectivas (uso y aplicación). 2.7. Bucles repetitivos (uso y aplicación). 3. Programación modular <ol style="list-style-type: none"> 3.1. Declaración de funciones 3.2. Simples

	<p>3.3. Con parámetros</p> <p>3.4. Uso de bibliotecas de funciones</p> <p>3.5. Entrada/Salida de Datos (C++, Java)</p> <p>3.6. Archivos</p> <p>3.7. Cadenas</p> <p>3.8. Vectores</p> <p>Matrices</p>
e)	<p>ESTRATEGIAS METODOLÓGICAS Y RECURSOS DIDÁCTICOS</p> <p>Estrategias metodológicas:</p> <p>Participación inductiva y Deductiva Técnicas:</p> <ul style="list-style-type: none"> - Estudio de casos - Abstracción - Observación - Investigación - Demostración <p>Recursos didácticos:</p> <ul style="list-style-type: none"> - Laboratorio - Software de diagramación POO - Proyector

Tabla 10 Contenidos mínimos Fundamentos de Programación

Fuente: Comunidad Académica Nacional

Nombre de la asignatura:	PROGRAMACIÓN ORIENTADA A OBJETOS

Campo de formación:	Adaptación e innovación tecnológica	
Unidad de organización curricular:	Formación Profesional	
Número de período académico	2	
Número total de horas de la asignatura:	196	
Organización de aprendizajes por modalidad, número de horas destinadas a cada componente (Art. 15 y 47 del RRA)	Componente Docencia:	72
	Componente de Prácticas de Aprendizaje:	54
	Componente de Aprendizaje Autónomo:	70
b)	<p>OBJETIVO DE LA ASIGNATURA:</p> <p>Objetivo general:</p> <p>Desarrollar aplicaciones de mediana complejidad aplicando el entorno de programación orientado a objetos. La codificación de los conceptos de la programación orientada a objetos se la realizará en el lenguaje Java.</p> <p>Objetivos específicos:</p> <ol style="list-style-type: none"> 1. Dominar los conceptos de la programación orientada a objetos, comportamiento, relaciones y operaciones, para la práctica. 2. Comprender la reutilización de código en la programación orientada a objetos. 	

	<ol style="list-style-type: none"> 3. Aplicar el lenguaje de programación Java en un grado de profundidad que puede realizar programaciones de complejidad mediana. 4. Realizar algoritmos en un lenguaje de programación orientado a objetos y producir programas ejecutables. 5. Emplear diferentes estrategias de solución, comparación y evaluación de los resultados
c)	<p>RESULTADOS DE APRENDIZAJE</p> <ul style="list-style-type: none"> - Aplica el paradigma de la programación orientada a objetos y utilizar los aspectos más relevantes en el diseño y el desarrollo de software. - Conoce la sintaxis del lenguaje Java. - Aborda la solución de problemas informáticos aplicando el paradigma de la programación orientada a objetos
d)	<p>CONTENIDOS MÍNIMOS DE LA ASIGNATURA:</p> <ol style="list-style-type: none"> 1. Introducción 2. Clases y objetos <ol style="list-style-type: none"> a. Tipos de datos b. Definición de una clase: atributos y métodos c. Modificadores de acceso d. Alcance de las variables e. Constructores f. Sobrecarga de método

- g. **Librerías**
 - h. **Encapsulamiento: Métodos getters y setters**
- 3. Herencia**
 - a. **Relación de herencia**
 - b. **Sobre-escritura**
 - c. **Sobrecarga**
- 4. Clases abstractas e interfaces**
- 5. Polimorfismo**
 - a. **Definición**
 - b. **Upcasting y downcasting**
- 6. Variables y métodos estáticos**
 - a. **Variables estáticas**
 - b. **Métodos estáticos**
 - c. **Constantes**
 - d. **Clases**
- 7. Manejo de errores y excepciones**
 - a. **Definición de una excepción**
 - b. **Tipos de excepciones**
 - c. **Manejo de excepciones**
- 8. Colecciones**
 - a. **Definición**
 - b. **Uso de colecciones**
- 9. Interface Gráfica**
 - a. **Componentes GUI (button, checkbox, list etc.)**
 - b. **Creación de la interfaz**

c. Manejo de eventos	
Conexión a base de datos	
e)	<p style="text-align: center;">ESTRATEGIAS METODOLÓGICAS Y RECURSOS DIDÁCTICOS</p> <p>Estrategias metodológicas:</p> <ul style="list-style-type: none"> • Metodología: Exposición magistral, demostraciones prácticas <p>Métodos: Inductivo, deductivo y experimental, etc.</p> <ul style="list-style-type: none"> • Técnicas: observación, experimental, mapas mentales, demostración, etc. <p>Recursos didácticos:</p> <ul style="list-style-type: none"> - Lenguaje de programación Java - Computadora - Pizarra - Proyector - Marcadores - Borrador

Tabla 11 Contenidos mínimos Programación Orientada a Objetos
Comunidad Académica Nacional

Fuente:

Los documentos de contenidos mínimos por asignatura detallan aspectos tales como: Periodo académico, número de horas de la asignatura, organización de aprendizajes en base a docencia, prácticas y autónomo. A esto se suma además el contar con el objetivo general y los específicos de cada asignatura. Los resultados de aprendizaje esperados se muestran en las tablas complementado con las estrategias metodológicas y recursos didácticos.

Todos los elementos de las tablas son criterios relevantes a la hora de realizar el análisis de la carrera de desarrollo de software y sobre todo las herramientas de desarrollo a la hora de poner en práctica en un ambiente de enseñanza aprendizaje.

3.1.3. Análisis de herramientas por asignatura

Luego de revisar todos los elementos dentro de los documentos de mallas curriculares y contenidos mínimos por asignaturas es importante resaltar que ninguno de estos presenta de forma estricta o formal el uso de una herramienta en específico. La variedad de herramientas de desarrollo de software, específicamente hablando de los lenguajes de programación orientados a objetos no se limita a un documento como es la malla curricular o los contenidos mínimos. De esta manera permite el definir criterios de selección a la hora de implementar el aprendizaje de un lenguaje de programación idóneo.

3.2. Entrevistas

Con la finalidad de recoger la mayor cantidad de información para el presente estudio, se aplicaron entrevistas para entender ejes temáticos del ámbito académico. En este punto se procedió a aplicar entrevistas a los coordinadores de la carrera de desarrollo de software de cada uno de los Institutos Superiores respectivamente. Es importante el entender la jerarquía del ámbito académico de las instituciones mostrado en la Ilustración 18.

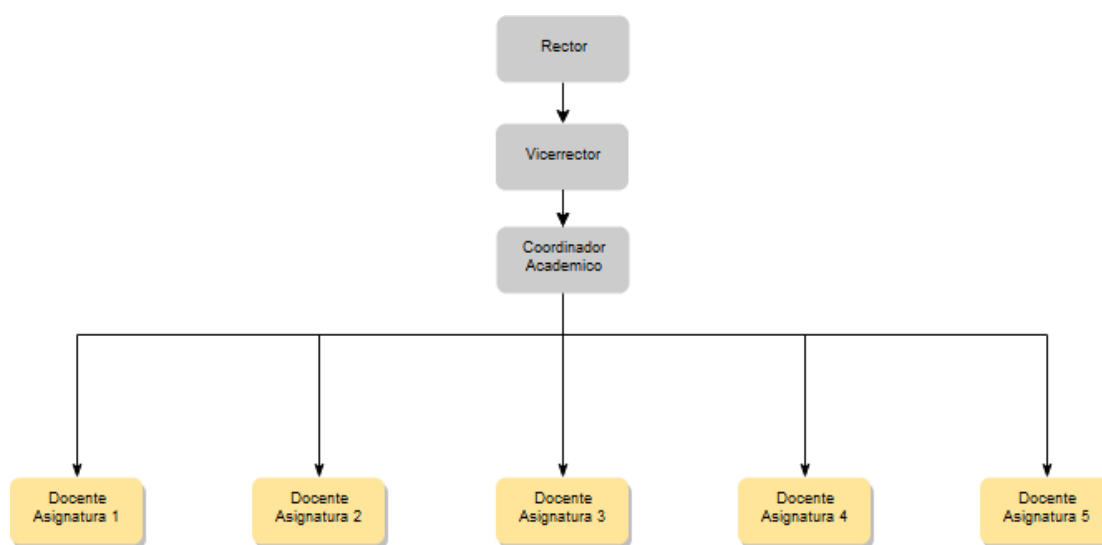


Ilustración 18 Jerarquía de roles IES

La figura de jerarquía de roles permite entender el énfasis que tienen las entrevistas a la hora de aplicarlas para la recolección de información esencial. Para el estudio se estableció que los entrevistados fundamentales del proceso son los coordinadores académicos de la carrera de desarrollo de software. En este contexto los coordinadores a ser entrevistados son cinco, uno por cada instituto tecnológico respectivamente. Cada coordinador académico es encargado de una carrera determinada y se dedica a gestionar el ámbito académico afianzados en un trabajo en conjunto

con vicerrectorado. A esto se también se añade que los coordinadores de carrera son profesionales Ingenieros en sistemas o afines con experiencia en el ámbito de desarrollo de software y gestión académica.

Se procedió a realizar las respectivas entrevistas basadas en parámetros de interés al estudio, que se encargan de abordar puntos para entender de manera más específica la carrera ofertada, software usado, proceso de enseñanza-aprendizaje, entre otros; y que se muestran a continuación en la Tabla 12.

Pregunta	Parámetro	Pregunta	Parámetro
1	Vigencia de la carrera	9	Inicio de carrera, metodología
2	Objetivo de la carrera	10	Dificultades al programar
3	Software libre o propietario	11	Dificultades comunes
4	Lenguaje de programación	12	Implementar una metodología
5	Que determina el software libre/propietario	13	Evaluación de Lenguajes de programación
6	Lineamientos técnicos	14	Paradigma de los lenguajes
7	Mejora y cambio de lenguaje	15	Proceso sistemático
8	Tendencias de programación	16	Documentación

Tabla 12 Elementos considerados en las entrevistas a los docentes.

Síntesis de entrevistas realizadas.

A continuación, en la Tabla 13, se muestra las respuestas más relevantes obtenidas de las entrevistas realizadas:

<i>Cargo</i>	<i>Coordinador Académico (AZU)</i>	<i>Coordinador Académico (JBV)</i>	<i>Coordinador Académico (ENA)</i>	<i>Coordinador Académico (SUDA)</i>	<i>Coordinador Académico (INAN)</i>
--------------	------------------------------------	------------------------------------	------------------------------------	-------------------------------------	-------------------------------------

<i>Núm.</i>	Instituto Tecnológico Superior Azuay	Juan Bautista Vásquez	Enrique Noboa Arízaga	Sudamericano	Integración Andina
	Preguntas				
	Síntesis de respuestas				
1	¿Cuántos años tiene de existencia la carrera de desarrollo de software?	2 años de aproximadamente	2 años y medio aproximadamente	2 años aproximadamente	2 años y medio aproximadamente
2	¿Cuál es el objetivo fundamental de la carrera?	El desarrollar aplicaciones web y móviles	El desarrollar software	El desarrollo de software	El análisis y desarrollo de software
3	¿Cuáles su enfoque en cuanto al uso de software libre o propietario?	Software libre y propietario	Software Libre	Software Libre	Software libre y propietario

4	¿Qué lenguaje o lenguajes de programación enseñan a lo largo de la carrera?	Java, javascript	javascript	javascript	Java, Javascript, Visual Basic, c#	JavaScript
5	¿Qué o Quién determina el enfoque de aprendizaje de software libre o propietario?	Una comisión	Rectorado y Vicerrectorado	Coordinador Académico	Coordinador Académico	Una comisión especializada
6	¿El impartir un lenguaje de programación determinado, previamente viene acompañado de lineamientos técnicos para la selección?	Criterios de experiencia docentes	y Criterio técnico de vicerrectorado	Criterios de docentes con experiencia	Análisis por parte de una comisión especializada	Análisis de mercado y criterios de docentes
7	¿La estructura de impartir cualquier lenguaje de programación puede ser mejorada?	Si, bajo criterios técnicos	Depende de la apertura de vicerrectorado	S, dependiendo del impacto a futuro	Siempre, basado en un análisis y diseño técnico	Si, dependiendo de un estudio previo

8	¿La carrera está a la vanguardia con las tendencias actuales de programación?	En la medida de lo posible	No siempre, dependemos de altos mandos	Tratamos de mantenernos actualizados	Siempre, mantenernos actualizados es prioridad	En la medida de lo posible
9	¿Al iniciar la carrera determinan el lenguaje de programación en base a alguna metodología?	No, únicamente de experiencias de coordinador y docentes	Se puede decir que se mantiene por tradición un lenguaje de programación	Nos basamos en base a experiencias de nuestros docentes	Se hace un estudio y análisis, mas no disponemos de una metodología	En base criterios de docentes con experiencia en el área de desarrollo
10	¿Los estudiantes presentan dificultades a la hora de aprender a programar?	Si, al inicio	Es bastante común	Si varios problemas de aprendizaje	sí, pero es normal y parte del proceso	Sucede bastante a menudo
11	¿Cuál es la dificultad más común que presentan los estudiantes?	La sintaxis de un lenguaje programación	Las palabras reservadas y funcionalidades	Funcionalidad y palabras reservadas	Sintaxis y semántica propia del lenguaje	Palabras reservadas, propiedades, métodos

12	¿Existen documentos o lineamientos formales que determinen el uso de un lenguaje de programación en el proceso de enseñanza-aprendizaje?	Únicamente los contenidos mínimos, pero no está ligado a un lenguaje específico	En base a criterios de autoridades es el sustento	No, únicamente criterios de selección a nivel de experiencias de docentes	Procuramos estar actualizados con las herramientas de aprendizaje, sin lineamiento o documento formal	Únicamente criterios técnicos de los docentes, mas no documentos formales
13	¿Evalúan el proceso de enseñanza-aprendizaje de los lenguajes de programación?	En cada parcia	AL final de cada parcial	En cada parcial de ciclo	Constantemente, para corregir errores	En los parciales
14	¿Qué tipo de paradigma manejan los lenguajes de programación que enseñan?	Orientado a Objetos	Orientación a objetos	Orientado a objetos	Con Orientación a objetos	Orientado a objetos
15	¿Considera que la selección de un lenguaje de programación	SI en gran parte para su análisis	Nos basamos más en experiencias	Tomando en cuenta debería	Si, siempre que este fundamentado	Nos basamos en experiencias profesionales

	obedece a criterios técnicos y un proceso sistemático?				considerar varios aspectos.			
16	¿Consideraría implementar una metodología de selección de lenguajes de programación?	Si, siempre que no afecte contenidos de	Puede ser, Si, que aporte de manera positiva en la carrera	dependemos de autoridades superiores	Si, que aporte de manera positiva en la carrera	Es bastante positivo y una excelente alternativa	Si aporta sin afectar procesos de enseñanza aprendizaje, bienvenida la metodología	

Tabla 13 Síntesis de entrevistas.

Se puede evidenciar en la tabla de síntesis de las entrevistas, que existen diferentes lineamientos en cada uno de los institutos superiores tecnológicos. A esto sumar que hay una tendencia del uso de software libre en institutos particulares en comparación con los institutos del estado, los cuales optan por combinar el aprendizaje con software libre y software pago.

Para destacar, se representa de manera gráfica estos parámetros.

Pregunta 1. La Ilustración 19 muestra, ¿Cuántos años tiene de existencia la carrera de desarrollo de software?



Ilustración 19 Años de existencia de la carrera de desarrollo de software.

En la figura se puede determinar que la carrera es relativamente nueva con apenas 2 años que se oferta. Recalcando que en los casos de los institutos que presentan 2,5 años ya tienen graduada su primera promoción. A esto hay que agregar que previo a esta carrera se ofertaba la de Análisis de Sistemas, la cual actualmente ya no está vigente y se encuentra en proceso de cierre con los últimos ciclos de la misma.

Pregunta 3. La Ilustración 20 muestra, ¿Cuál es su enfoque en cuanto al uso de software libre o propietario?

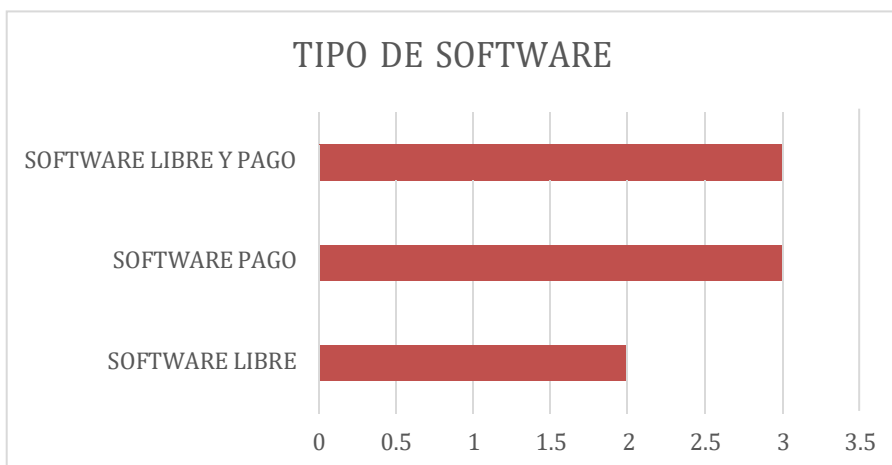


Ilustración 20 Tipo de software empleado en el proceso de enseñanza-aprendizaje

Esta figura permite entender, que existen cinco institutos, de los cuales 3 de ellos implementan en su proceso la combinación de software libre y pago. Los dos institutos restantes únicamente enfocan la enseñanza de software libre.

Pregunta 4. La Ilustración 21 muestra, ¿Qué lenguaje o lenguajes de programación enseñan a lo largo de la carrera?

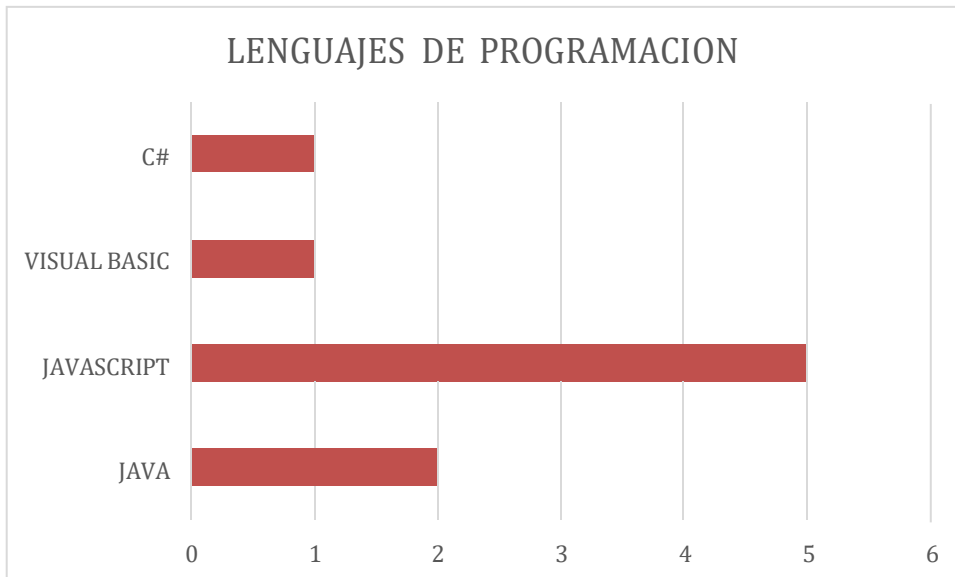


Ilustración 21 Lenguajes de programación enseñados.

En la figura se puede observar la tendencia mayoritaria hacia el uso de javascript como lenguaje de programación. Por otra parte, la incidencia de C#, Visual Basic y Java de forma minoritaria en el proceso de enseñanza aprendizaje, determinan que no son considerados otros lenguajes de programación existentes en el mercado por todos los institutos.

Pregunta 12. La Ilustración 22 muestra, ¿Existen documentos o lineamientos formales que determinen el uso de un lenguaje de programación en particular en el proceso de enseñanza-aprendizaje?

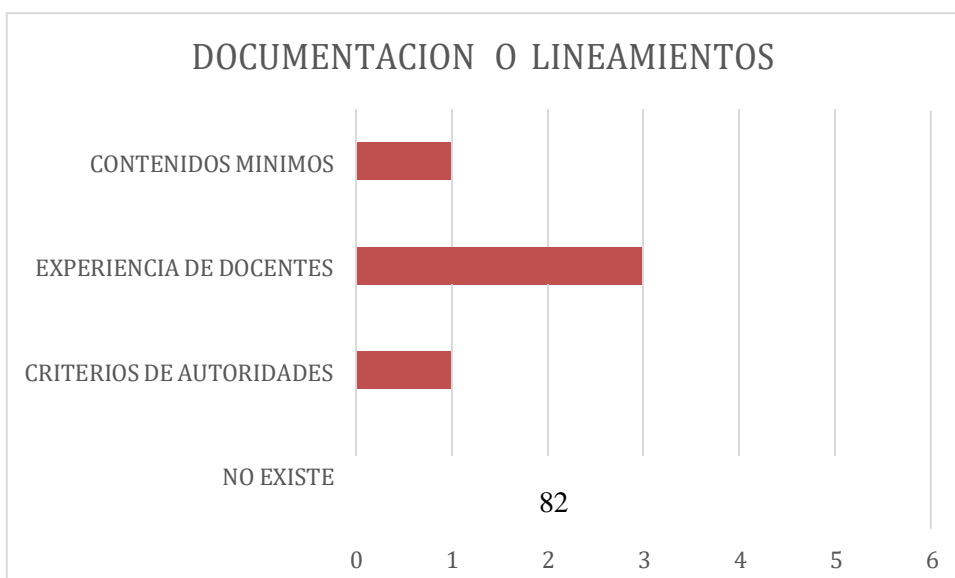




Ilustración 22 Documentos o lineamientos formales de selección de lenguajes de programación.

Este grafico se puede evidenciar que por parte de los coordinadores de carrera manifiestan que no existe un documento formal o lineamiento de selección para un lenguaje de programación. Es notorio el destacar además que la selección de los lenguajes de programación para el proceso de enseñanza-aprendizaje están llevados por otros fundamentos tales como: contenidos mínimos de la asignatura, experiencia de docentes, criterio de autoridades.

Pregunta 16. La Ilustración 23 muestra, ¿Consideraría implementar una metodología de selección de lenguajes de programación?

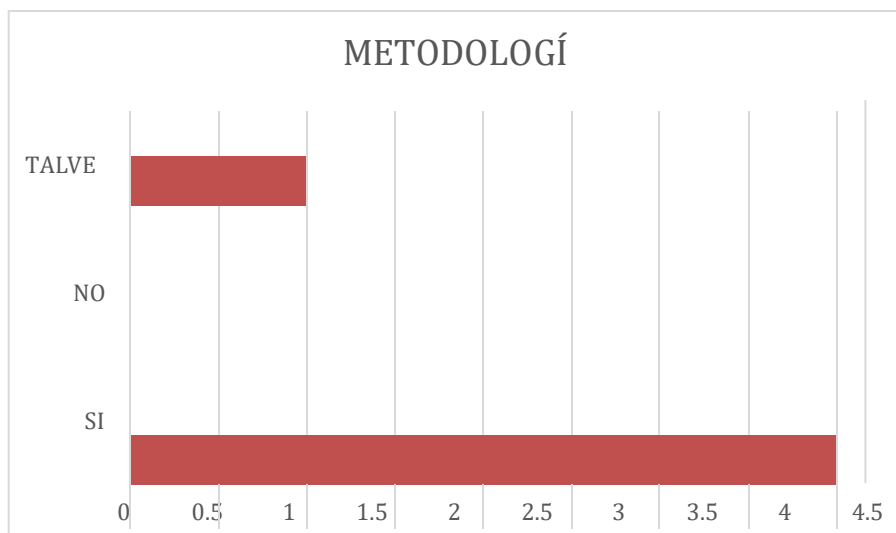


Ilustración 23 Considerar uso de una metodología.

En este grafico se puede destacar la aceptación mayoritaria positiva para poner en marcha la implementación de una metodología de selección de lenguajes de programación. Es importante el dejar en claro además que 1 instituto tomaría la propuesta de manera imparcial.

Todos los gráficos extraídos de los datos de las entrevistas realizadas a los coordinadores de carrera permiten evidenciar varios parámetros fundamentales para el estudio. Con las entrevistas se buscó recopilar información del ámbito académico para entender mejor el proceso. Todo esto aporta significativamente a la hora de desarrollar la estructura de la metodología de selección de lenguajes de programación.

3.3. Encuestas

En este apartado del estudio se procedió a dividir en 2 escenarios, los cuales complementa la perspectiva de análisis y se detallan en la Tabla 14

Escenario 1	Escenario 2
Antes de implementar la metodología	Después de implementar la metodología
<ol style="list-style-type: none">1. Describir la estructura formal del proyecto.2. Definir competencias3. Evaluar aprendizaje4. Analizar resultados	<ol style="list-style-type: none">1. Describir la estructura formal del proyecto.2. Implementar la metodología3. Definir competencias4. Evaluar aprendizaje5. Analizar resultados6. Comparativa de resultados obtenidos entre los dos escenarios.

Tabla 14 Escenarios de estudio.

Estos escenarios plantean dos puntos de análisis, en primer lugar, sin el uso de una metodología de selección de lenguajes de programación. Es decir, partiendo del estado actual y las realidades de los institutos superiores. En el segundo escenario permitirá evidenciar una perspectiva con la implementación de la metodología de selección de lenguajes. Todo esto con el objetivo de comparar los resultados y evidenciar si apporto o no la implementación de a metodología en el proceso de enseñanza aprendizaje.

Desde el punto de vista general se planteó un escenario basado en los siguientes elementos los cuales forman parte del proceso de enseñanza-aprendizaje que normalmente se lleva a cabo en los institutos de educación superior que son objeto d estudio. Las encuestas en primera instancia permiten el analizar el primer escenario del proceso de enseñanza aprendizaje sin una metodología.

3.3.1. Escenario 1

La encuesta aplicada se enfoca en base a cuatro secciones las cuales se mencionan a continuación:

1. Experiencia en el ámbito de programación
2. Proceso de enseñanza aprendizaje

3. Evaluar el lenguaje de programación impartido en la asignatura.
4. Parámetros específicos de la programación Orientada a objetos

Se muestran en la Ilustración 24 el escenario sin implementación de la metodología.

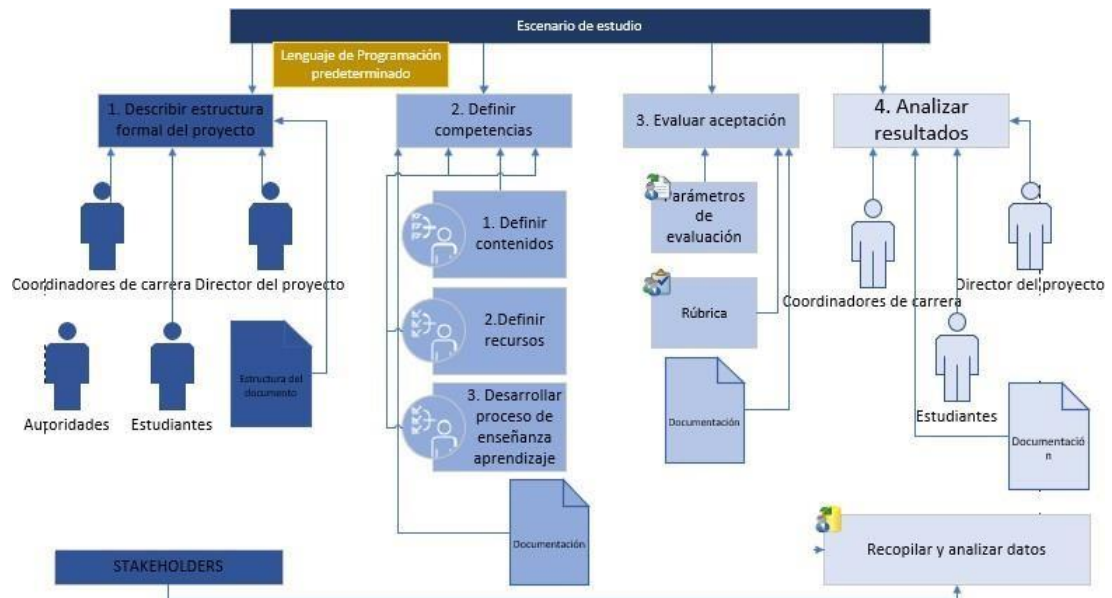


Ilustración 24 Escenario 1, sin implementación de metodología.

La estructura de secciones de la encuesta presenta el siguiente banco de preguntas mostrados en la Tabla 15:

1. <i>Experiencia en el ámbito de programación</i>	2. <i>Proceso de enseñanzaaprendizaje</i>
<ul style="list-style-type: none"> ✓ ¿Tiene Ud. algún tipo de experiencia en desarrollo de software? ✓ ¿Conoce Ud. algún lenguaje de programación? Mencione alguno. ✓ ¿Ha escuchado hablar de las tendencias actuales de programación? ✓ ¿Sabe Ud. usar algún lenguaje de programación? Menciónelo ✓ ¿Qué paradigma de programación aprenden en la asignatura? ✓ ¿Qué lenguaje de programación aprenden en la asignatura? ✓ ¿Considera que el lenguaje de programación usada aporta en su proceso de aprendizaje? 	<ul style="list-style-type: none"> ✓ ¿El Docente da a conocer los contenidos de la asignatura al inicio del ciclo? ✓ ¿El Docente da a conocer las herramientas de desarrollo de software para el ciclo? ✓ ¿El Docente da a conocer los parámetros de evaluación o rubrica? ✓ ¿el Docente considera sugerencias de herramientas de desarrollo de software?
<p>3. <i>Evaluar el lenguaje de programación impartido en la asignatura.</i></p>	<p>4. <i>Parámetros específicos de la programación Orientada a objetos</i></p>

- ✓ El lenguaje de programación actual estudiado en la asignatura es fácil de comprender.
- ✓ Las palabras reservadas del lenguaje son fáciles de recordar
- ✓ La estructura de los bucles de repetición es fácil de implementar
- ✓ La sintaxis del lenguaje de programación es fácil en intuitiva

- ✓ Estructura básica del programa "Hola Mundo"
- ✓ Uso de variables y constantes
- ✓ Implementación de un bucle
- ✓ Uso de librerías
- ✓ Creación de funciones
- ✓ Uso de Clases, atributos y métodos
- ✓
- ✓ Encapsulamiento
- ✓ Herencia
- ✓ Polimorfismo
- ✓ Abstracción

Tabla 15 Preguntas encuesta

En base a la encuesta aplicada a los estudiantes, se procede a mostrar gráficamente los resultados obtenidos de las preguntas más relevantes para el estudio. A continuación, la Ilustración 25 muestra, ¿Qué lenguaje de programación aprenden en la asignatura?

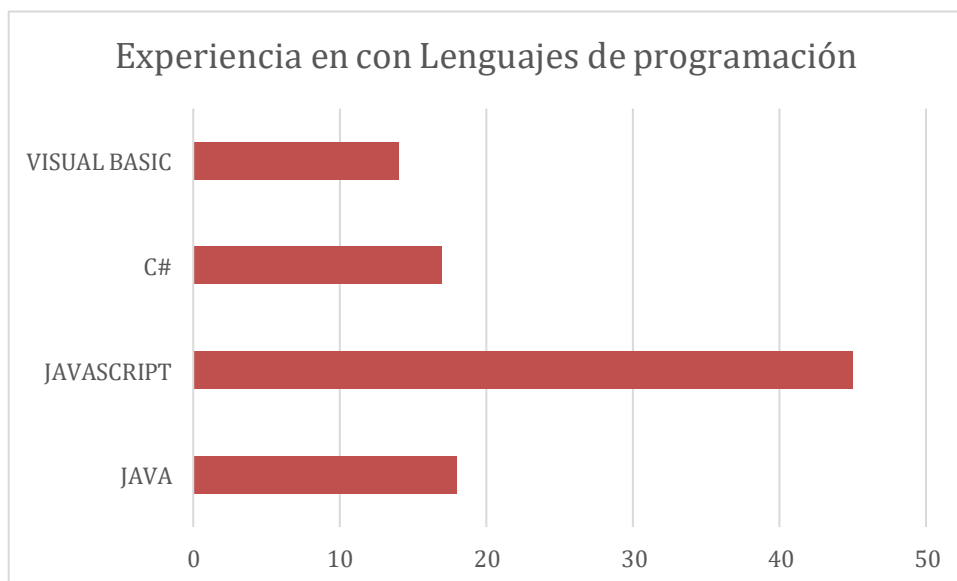


Ilustración 25 Experiencia con algún lenguaje de programación.

Este grafico permite interpretar que mayoritariamente los estudiantes programan con el lenguaje Javascript y en segundo plano destacan Java con C# y finalmente visual Basic por una mínima diferencia de frecuencia en el gráfico. A continuación, la Ilustración 26 muestra, ¿Ha escuchado hablar de las tendencias actuales de programación?

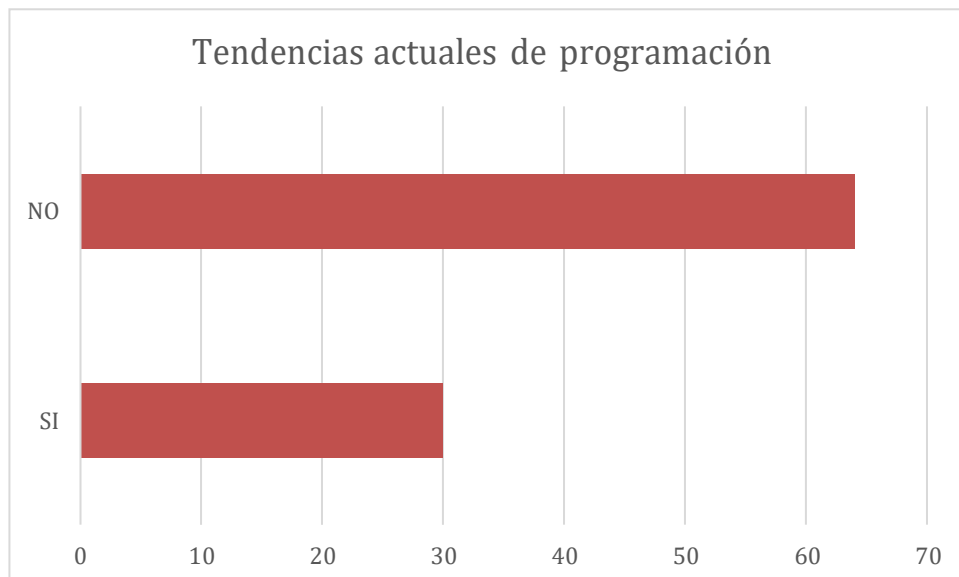


Ilustración 26 Conocimiento de las tendencias actuales de programación.

El grafico muestra que una cantidad, menos de la mitad de la muestra conoce las tendencias actuales de programación. El resto de encuestados desconoce la temática. A continuación, la Ilustración 27 muestra, ¿El Docente da a conocer las herramientas de desarrollo de software para el ciclo?

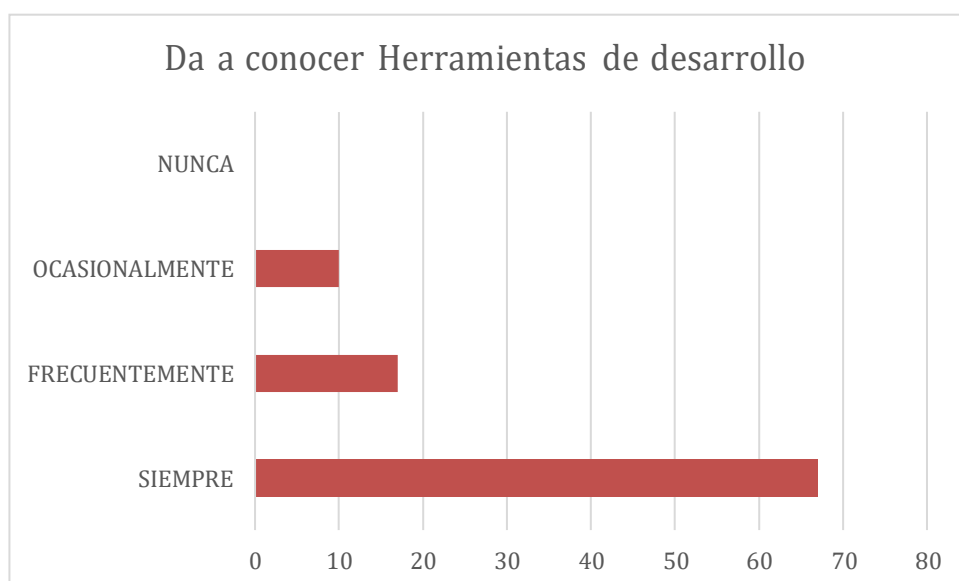


Ilustración 27 Dar a conocer herramientas usadas en el proceso

En este gráfico, cabe recalcar que mayoritariamente con un 68% los estudiantes indican que le docente da a conocer las herramientas de desarrollo a ser usadas en el proceso de enseñanza aprendizaje. Por otra parte, un 18% indica que frecuentemente da a conocer el docente y finalmente 10% manifiesta que ocasionalmente lo realiza el docente. A continuación, la Ilustración 28 muestra, ¿El Docente da a conocer los parámetros de evaluación o rubrica?

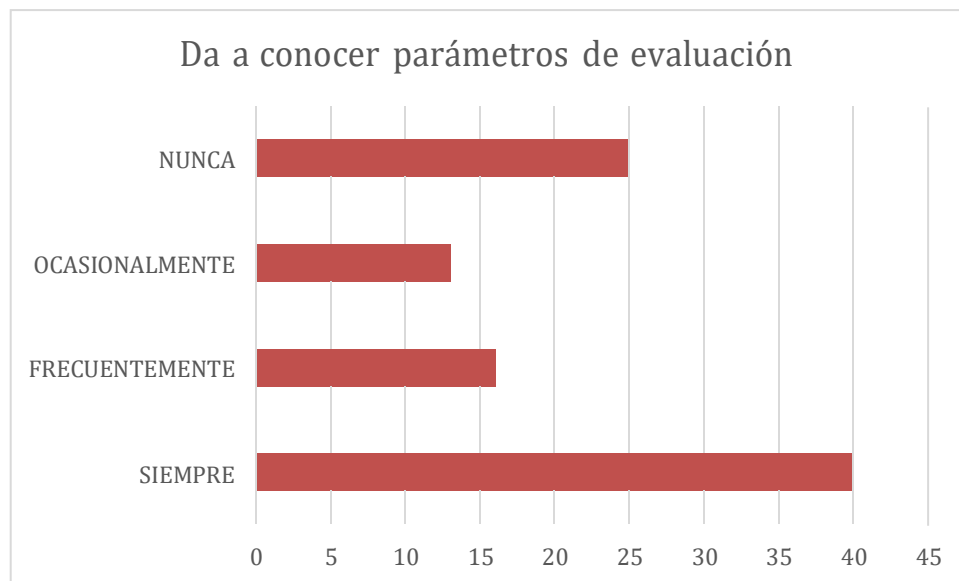


Ilustración 28 Dar a conocer parámetros de evaluación

Esta grafica es fundamental a la hora de determinar la acogida por parte de los estudiantes a la hora de medir la aceptación de la metodología. Este apartado permite el establecer un modelo de evaluación para los parámetros de programación practico. Todo esto se enfoca en la implementación de una rúbrica.

A continuación, se realiza la valoración con una escala de uno a diez, siendo 10 la máxima nota y 1 la mínima, de los lenguajes de programación que actualmente forman parte del proceso de enseñanza-aprendizaje de las respectivas asignaturas.

Hay que destacar que el objeto de estudio es el desarrollar una metodología y establecer los niveles de aceptación, mas no el enfocarse en el rendimiento de los estudiantes. De esta forma se busca analizar puntos claves y cuantitativos a través de la experiencia de uso de los estudiantes acorde a la medición de los siguientes parámetros mostrados en la Tabla 16.

<i>Parámetro / Medición</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>Total</i>
<i>Lenguaje actual</i>				21	45	20	6	2			94
<i>Palabras reservadas</i>				35	40	10	2	7			94
<i>Sintaxis</i>				25	33	25	6	5			94
<i>Estructura de bucles</i>			20	25	30	10	8	1			94

Tabla 16 Evaluación del lenguaje de programación usado

La tabla permite evidenciar que los elementos considerados para esta medición a través de la encuesta dejan ver que tiende a un término medio, centralizando la aceptación. Esto indica que, con una metodología, se pudiese mejorar este análisis.

3.4. Metodología de Selección.

La especificación de metodología para la selección de un lenguaje de programación, se fundamentó en los las temáticas de estudio, lineamientos técnicos y parámetros de evaluación. A lo largo del estudio se encontraron estos lineamientos de análisis para la construcción de una metodología. Dichos lineamientos, orientan el planteamiento de actividades y tareas que conlleva realizar la metodología.

Le metodología define un conjunto de elementos de una estructura funcional, a partir de los cuales fue como se posibilito el definir un grupo de actividades necesarias para organizar y orientar el diseño de las fases de la metodología. Esta se presenta como una alternativa al momento de realizar la selección de un lenguaje de programación orientado a objetos enfocado a un ambiente de enseñanza aprendizaje.

- 3.4.1. Fases de la metodología de selección de lenguajes de programación orientada a objetos.
- 3.4.2. El método científico, para realizar una investigación pasa por diversas fases. Son varios autores que las interpretan y se construirá en base a las cinco fases propuestas por Kerlinger y Lee.

La metodología se estructuró en cinco fases: diseño, tendencias, implementación, seguimiento y cierre. Cada una de ellas se encuentra diseñada para entrada de datos, procesamiento y salida de información. Se orienta a la realización de actividades en cada una de sus fases. La Ilustración 31 muestra cada una de las fases:

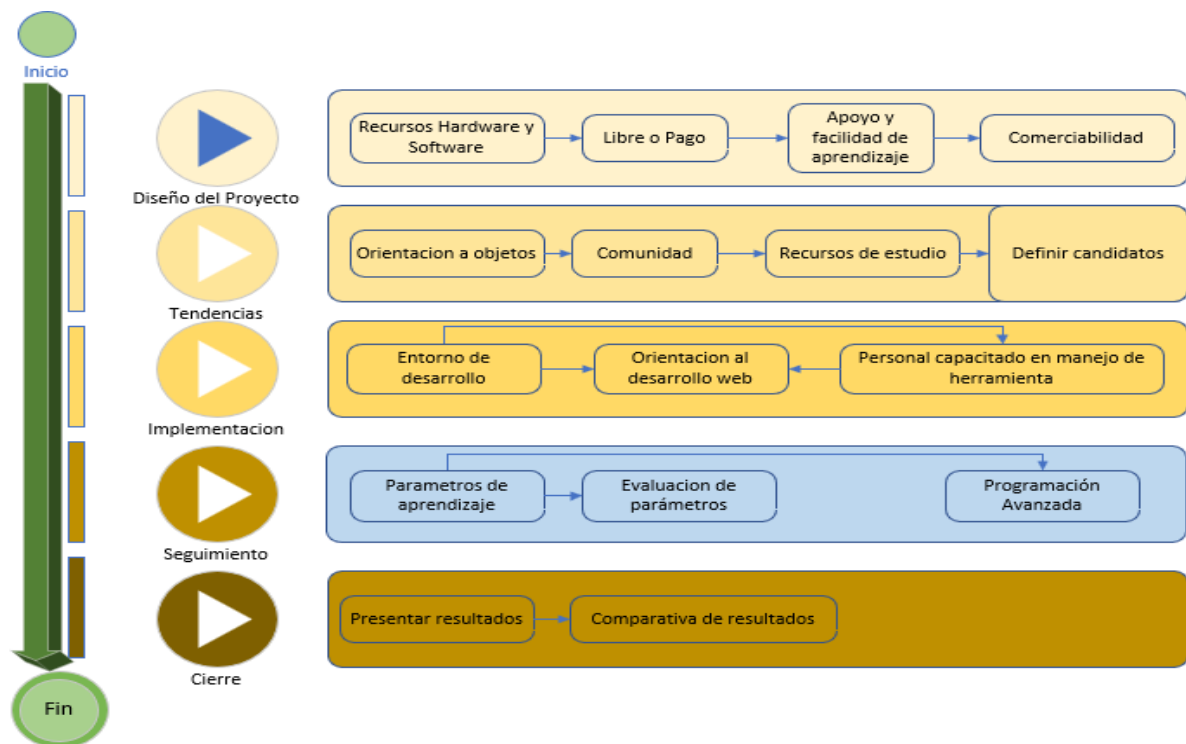


Ilustración 31 Metodología de selección de lenguajes de programación orientados a objetos

3.4.2.1. Fase de diseño del proyecto

En esta fase se propone actividades tales como:

1. Recursos de Hardware y Software.
 - a. Definir las características físicas de los computadores con los que se cuenta.
 - b. Definir el software o programas con los que interactúa el hardware.
2. Establecer el uso de Software libre o Pago
 - a. Definir el uso de usar software libre
 - b. Definir el uso de usar software pago
 - c. Definir el uso de los dos tipos de software
3. Apoyo y facilidad de aprendizaje
 - a. Establecer la curva de aprendizaje del lenguaje
 - b. Establecer curva de crecimiento
 - c. Factibilidad de dar conocer conceptos fundamentales de POO de manera clara y concreta.
4. Definir la comerciabilidad del lenguaje en el medio. Tal como se muestra en la Ilustración 32.
 - a. Establecer el uso y explotación del lenguaje en empresas a nivel local
 - a. Establecer el uso y explotación del lenguaje a nivel de empresas internacionales.

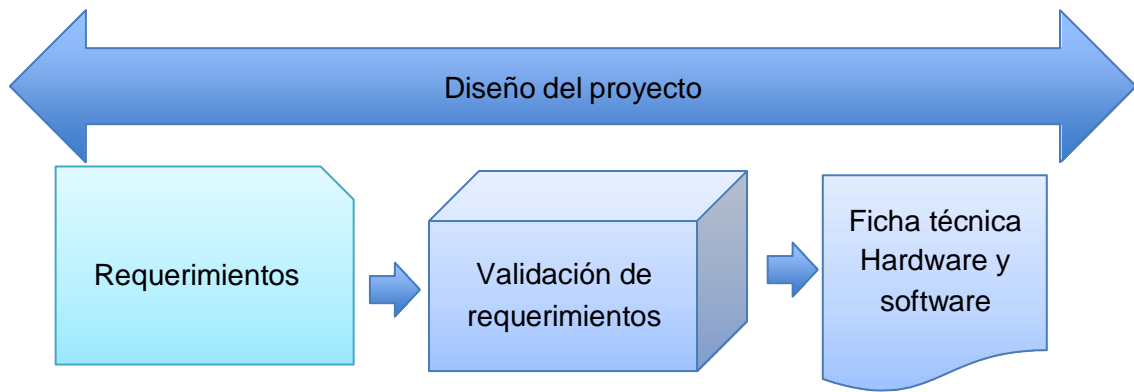


Ilustración 32 Fase de diseño del proyecto

En esta fase se establece los requerimientos técnicos con los que se cuenta a nivel de hardware y software. Además, se analiza el enfoque de trabajo de la herramienta, curva de aprendizaje para posteriormente validar esta información y finalmente obtener una ficha técnica de requerimientos iniciales con los que se cuenta o se debería contar.

3.4.2.2. Fase de tendencias de programación

En esta fase se destaca por actividades basadas en las tendencias actuales de programación. Previo al desarrollo de la misma es fundamental definir las comunidades de desarrollo tomadas en cuenta para tomar las tendencias de programación actuales, entre las consideradas para el estudio se encuentran:

TIOBE: <https://www.tiobe.com/tiobe-index/>. Como se muestra a continuación en la Ilustración 33.

Nov 2020	Nov 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.21%	+0.17%
2	3	▲	Python	12.12%	+2.27%
3	1	▼	Java	11.68%	-4.57%
4	4		C++	7.60%	+1.99%
5	5		C#	4.67%	+0.36%
6	6		Visual Basic	4.01%	-0.22%
7	7		JavaScript	2.03%	+0.10%
8	8		PHP	1.79%	+0.07%
9	16	▲	R	1.64%	+0.66%
10	9	▼	SQL	1.54%	-0.15%

Ilustración 33 Índice Tiobe, tendencias de programación.

PYPL: <https://pypl.github.io/PYPL.html>. Como se muestra a continuación en la Ilustración 34.

Worldwide, Nov 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	30.8 %	+1.8 %
2		Java	16.79 %	-2.3 %
3		JavaScript	8.37 %	+0.3 %
4		C#	6.42 %	-0.9 %
5		PHP	5.92 %	-0.2 %
6		C/C++	5.78 %	-0.2 %
7		R	4.16 %	+0.4 %
8		Objective-C	3.57 %	+1.0 %
9		Swift	2.29 %	-0.2 %
10		TypeScript	1.84 %	-0.0 %

Ilustración 34 Índice PYPL, tendencias de programación.

GITHUB: <https://octoverse.github.com/>. Como se muestra a continuación en la Ilustración 35.

# Ranking	Programming Language	Percentage (Change)
1	JavaScript	20.147% (-0.161%)
2	Python	15.873% (-1.195%)
3	Java	11.404% (+1.038%)
4	Go	8.814% (+0.005%)
5	TypeScript	7.488% (+1.479%)
6	C++	6.934% (-0.117%)
7	Ruby	6.185% (+0.316%)
8	PHP	5.030% (-0.034%)
9	C#	3.716% (-0.092%)
10	C	2.890% (-0.305%)

Ilustración 35 Índice GITHUB, tendencias de programación.

Al contar con estos recursos de tendencias de programación que se actualizan mensual o trimestralmente, se puede estructurar esta fase de manera efectiva. A continuación, se define las actividades que contemplan esta fase:

1. Orientación a objetos
 - a. Establecer los lenguajes que tienen el paradigma de orientación a objetos
2. Comunidad
 - a. Establecer comunidades de desarrolladores solidas que den continuidad al proceso de formación
3. Recursos de estudio
 - a. Respalidar el proceso formativo con documentos físicos tales como: libros, manuales, artículos, entre otros.
 - b. Respalidar el proceso formativo con documentos digitales tales como: videos, libros, manuales, artículos, entre otros.
4. Definir candidatos
 - a. En base a las actividades anteriores se pueden definir candidatos unidireccionales o bidireccionales.
 - i. Unidireccionales: Software Libre
 - ii. Bidireccionales: Software libre y pago

A continuación, el grafico del proceso se muestra a continuación en la Ilustración 36.

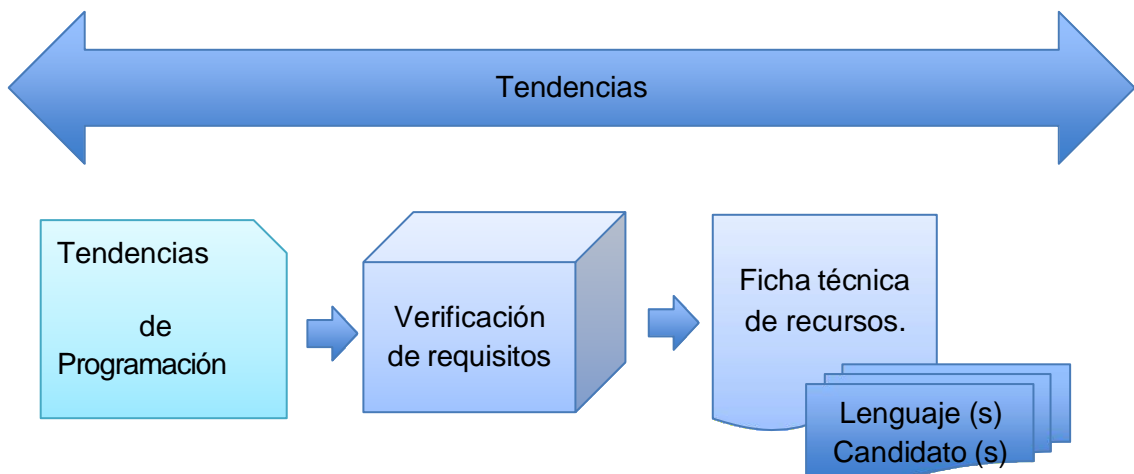


Ilustración 36 Fase de tendencias

3.4.2.3. Fase de implementación

1. Entorno de desarrollo

- a. En primera Instancia ya con el o los lenguajes de programación candidatos, es fundamental definir el entorno de desarrollo idóneo (IDE)
- b. Verificar la compatibilidad y funcionalidad del IDE con el lenguaje candidato.

2. Orientación al desarrollo web

- a. Verificar los tipos de aplicaciones que permite desarrollar el lenguaje.
- b. En este caso en particular enfocados en el desarrollo web.

3. Personal capacitado en manejo del lenguaje

- a. Verificar la disponibilidad de docentes con conocimiento del lenguaje.
- b. Capacitar a los docentes en cuanto al manejo de la herramienta de manera general.

Considerar: Un docente desarrollador con experiencia en aplicaciones web en cualquier lenguaje de programación, conoce y aplica los fundamentos de la programación orientada a objetos. De aquí la importancia de mencionar que un lenguaje OO varía únicamente en cuanto al uso de palabras reservadas y estructuras. Pero cualquier lenguaje de programación OO se basa en los pilares fundamentales de la POO:

A continuación, el gráfico del proceso se muestra a continuación en la Ilustración 37.

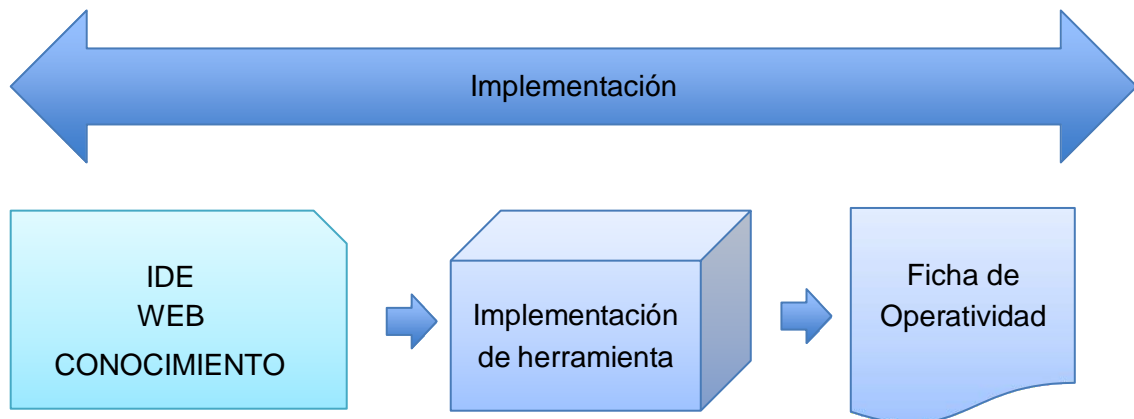


Ilustración 37 Fase de implementación

3.4.1.4. Fase de seguimiento

En esta fase se destaca por realizar tres actividades a continuación detalladas:

1. Parámetros de Aprendizaje
 - a. Considerar los pilares fundamentales de la POO:
Polimorfismo, Encapsulamiento, Abstracción y Herencia.
 - b. Tomar en cuenta el uso de clases, Atributos y Métodos.
 - c. Manejo de bucles.
 - d. Acceso y compatibilidad con bases de datos
2. Evaluación de Parámetros
 - a. En este punto es esencial el tener definido parámetros de evaluación por medio de una técnica en específico tales como:
 - i. Una rúbrica
 - ii. Lista de Cotejo
 - iii. Otras herramientas de valoración.
3. Programación Avanzada: Adicional al desarrollo de aplicaciones Web.
 - a. Considerar el desarrollo en otros ámbitos tales como:
 - i. Inteligencia de negocios
 - ii. Machine learning
 - iii. Inteligencia Artificial
 - iv. Otras disciplinas.

A continuación, el gráfico del proceso se muestra a continuación en la Ilustración 38.

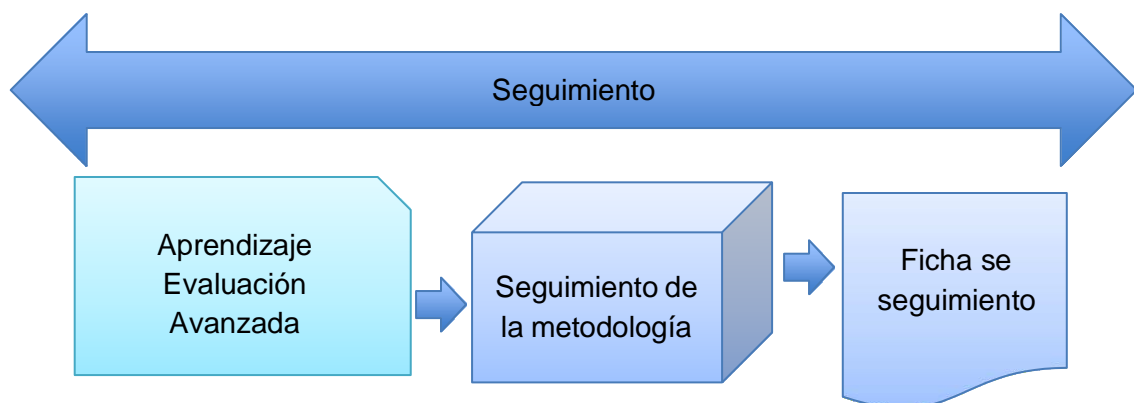


Ilustración 38 Fase de seguimiento

3.4.1.5. Fase de cierre

Finalmente, la fase de cierre se caracteriza por la realización de dos actividades descritas de la siguiente manera:

1. Presentar resultados
 - a. Recolectar la documentación de las fases anteriores:
 - i. Ficha técnica Hardware y software
 - ii. Ficha técnica de recursos.
 - iii. Ficha de Operatividad
 - iv. Ficha de seguimiento
2. Comparativa de resultados

Esta actividad permite dar seguimiento y retroalimentando la metodología.

- a. Realizar un contraste entre lenguajes candidatos en el caso de usar software gratuito y de pago.
 - b. Al finalizar la implementación de la metodología se puede evaluar los resultados obtenidos para compararlos diferentes resultados obtenidos.
3. A continuación, el gráfico del proceso se muestra a continuación en la Ilustración 39.

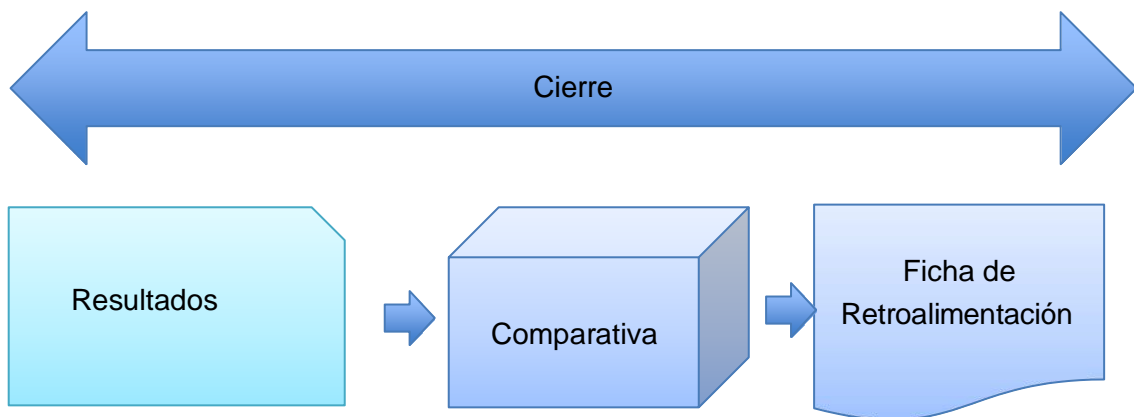


Ilustración 39 fase de cierre

Cabe indicar que cada una de las cinco fases de la metodología, están determinadas por un proceso de entrada, procesamiento de información y salida. La salida en este caso se

presenta como artefactos entregables para documentar y retroalimentar la metodología de selección de lenguajes de programación como se muestra el formato en la Tabla 19.

3.4.3. Formato de Ficha de fases

Formato de Ficha			
Nombre de la fase			
Actividades			
Actividad 1			
Actividad 2			
Actividad 3			
Actividad 4			
Recursos			
Gráfico de Procesos de la fase			
Observaciones			
Fecha:			Responsable

Tabla 19 Formato de ficha de fases de metodología

3.5. Aplicación de metodología

En este apartado se aplica el escenario que conlleva la evaluación de la metodología con la implementación de la misma en el estudio.

3.5.1. Escenario 2

Posterior al desarrollo de metodología de selección se procedió a realizar el segundo escenario del estudio en el cual se contempla la implementación de la misma. Tomando en cuenta los mismos lineamientos y parámetros analizados en el escenario uno se procede a realizar el estudio como se muestra en la Ilustración 40.

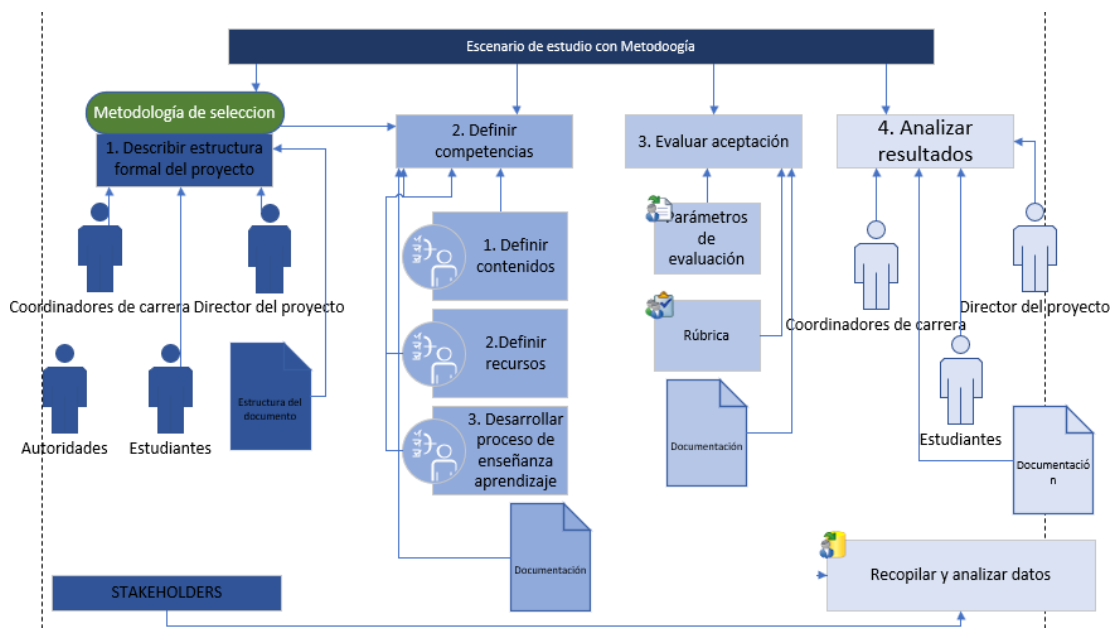


Ilustración 40 Escenario 2, con la implementación de metodología.

Aplicando la metodología de selección de lenguajes de programación orientados a objetos para la implementación en un ambiente de enseñanza-aprendizaje se muestra el proceso en la Tabla 20.

Fase	Actividad	Entrada	Proceso	Salida
Diseño	1. Recursos	Core i7, 8 Ram, 500Gb Disco Duro	Validación Verificación de información	Diseño acorde a necesidades

- 2. Tipo Software Libre
Crecimiento
- 3. Apoyo Comerciability

Tendencias	1. Objetos	Aplica Orientación a Objetos	Criterios basados tendencias actuales	Lenguajes candidatos en orden de prioridad:
	2. Comunidad	Comunidades grandes de desarrolladores ingles / latinas		1. Python 2. Java 3. JavaScript
	3. Recursos	Físicos / digitales PYTHON -		
	4. Candidatos	JAVA - JAVASCRIPT		
Implementación	1. IDE	VSCode - Eclipse	Implementación de entornos funcionalidad.	IDE y Lenguaje de programación operativos
	2. WEB Experiencia	Desarrollo Web Pilares de la POO		
Seguimiento	1. Aprendizaje	Pilares de la POO Clases, Atributos, métodos Rúbrica	Proceso de aprendizaje y evaluación de fundamentos de POO.	Aplicación de rubrica de evaluación
	2. Evaluación		Lenguaje de programación	

	3. Avanzado	Estadística / con mayor Inteligencia de alcance y negocios prestaciones		
Cierre	1. Resultados	Fichas de fases	Comparativa de	Retroalimentación
	2. Comparativa	anteriores	lenguajes candidatos y parámetros	de la metodología

Tabla 20 Aplicación de metodología de selección

Basados en el análisis y las actividades descritas en la metodología, y tomando en cuenta las tendencias actuales de programación, actualmente Python se presenta con mayores atenuantes a favor en comparación con Java y JavaScript.

Finalmente se procede a realizar la evaluación de la implementación de la metodología bajo parámetros y lineamientos establecidos en el escenario uno.

Capítulo 4

4. Generalidades.

En este capítulo se muestra los resultados obtenidos y se establecen las comparaciones sin la metodología y con la implementación de la misma. De igual forma se evalúa la metodología con criterios de expertos el ámbito.

4.1 Metodología de selección

Como primer resultado obtenido se presenta la metodología de selección de los lenguajes de programación para desarrollo de software, aplicado a un ambiente de enseñanza-aprendizaje presencial para los institutos de educación superior de las provincias de Azuay y Cañar. Se muestra a continuación en la ilustración 41.

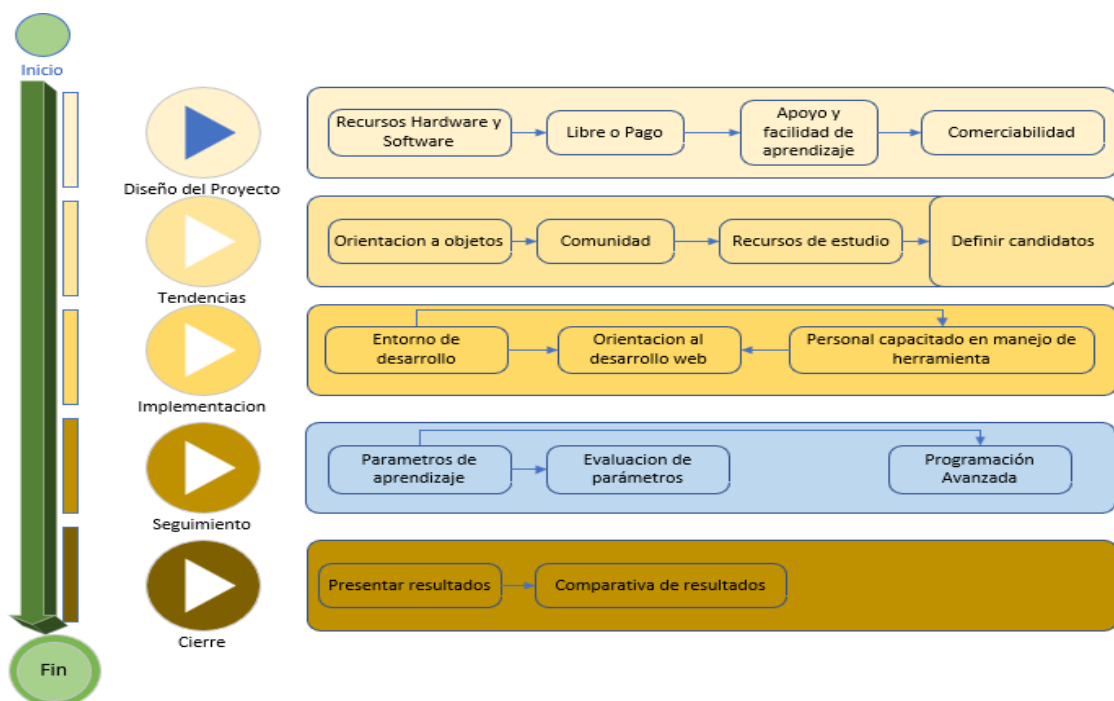


Ilustración 41 Objetivo general. Metodología de selección

Cabe indicar que esta metodología hace referencia al objetivo general establecido en el presente estudio, el desarrollo y explicación de la misma se encuentra en el capítulo tres.

4.2. Tendencias actuales de programación

En esta sección se muestra las tendencias actuales de programación, las cuales se actualizan cada mes. Esta información está basada en comunidades de programadores a nivel global. Esta información aporta significativamente el estudio a la hora de construir e implementar una metodología de selección. Esencialmente el estudio está basado en 3 índices de valoración de lenguajes de programación que se detalla a continuación:

- ✓ Índice Tiobe tomado del sitio: <https://www.tiobe.com/tiobe-index/>
- ✓ Índice PYPL tomado del sitio: <https://pypl.github.io/PYPL.html>
- ✓ Índice GitHub tomado del sitio: <https://octoverse.github.com/>

4.3. Recolectar y tabular información Escenario 1

En el escenario uno, con el panorama de no tener una metodología de selección de lenguajes de programación orientados a objetos se obtuvieron los siguientes datos estadísticos presentados en la Ilustración 42.

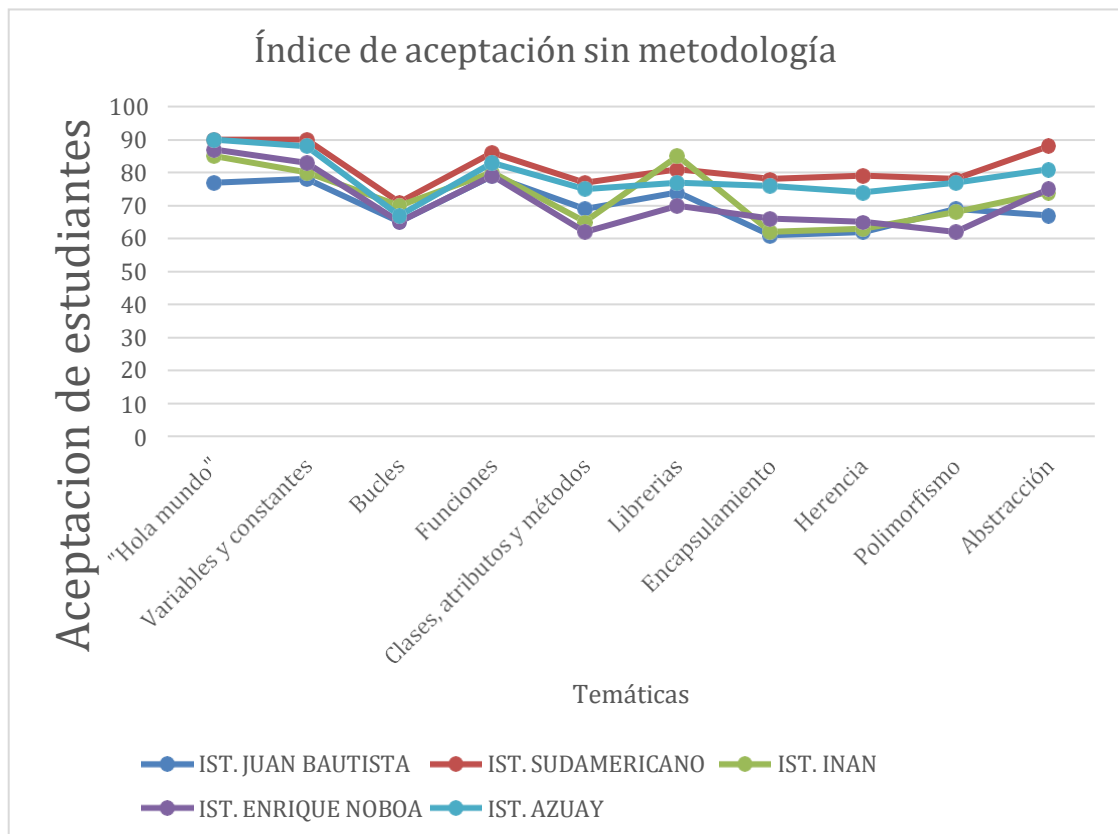


Ilustración 42 Aceptación sin una implementación de Metodología

Esta ilustración permite evidenciar que las 10 temáticas de análisis, se mantienen con un valor alto en 90 y un valor bajo en 60. Lo cual permite determinar claramente que

existe una variación significativa entre valores y más aún cuando se compara entre institutos.

Escenario 2

Escenario dos, con la implementación de la metodología de selección bajo los mismos parámetros y lineamientos expuestos en el escenario uno, se evidencia en la ilustración 43.

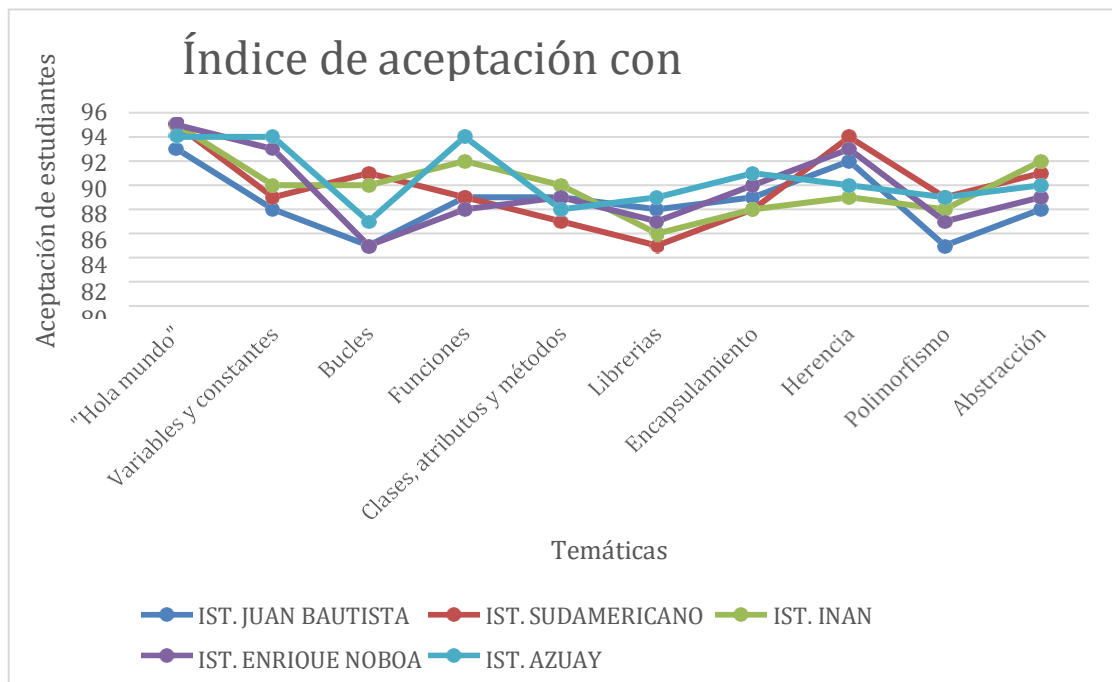


Ilustración 43 Aceptación con la implementación de Metodología

En esta ilustración es evidente el rango de frecuencias que se encuentra en valores mucho más elevados a comparación de la ilustración anterior. Este rango de valores fluctúa entre 80 y 96, con algunas variantes si se pone en punto de comparación entre los distintos Institutos de Educación Tecnológica Superior.

Cabe destacar que la implementación de la metodología, obtuvo como lenguaje de programación destacado para implementación a Python. Esta perspectiva está dada tomando en cuenta que se enfocaría en el uso de software libre.

4.4. Evaluar la metodología

Basado en criterio de expertos, a través de la aplicación del método Delphi, se realiza la evaluación de la metodología en base a parámetros mostrados en la tabla 21. Esto permite determinar la eficacia de la implementación de la metodología. Además, brinda una retroalimentación de mejora para aplicar a la misma de manera que pueda mejorar aspectos relevantes.

Método Delphi

<i>Parámetro</i>	Descripción
<i>Definir el tema</i>	Metodología de selección de lenguajes de Programación Orientados a Objetos
<i>Hacer el cuestionario</i>	Banco de Preguntas (10)
<i>Definir expertos</i>	Cinco expertos con experiencia mínima de 5 años en desarrollo de aplicaciones de escritorio, web y móviles.
<i>Informar a expertos su papel</i>	Evaluar la metodología
<i>Distribuir cuestionario</i>	Expertos independientes entre si
<i>Tabular respuestas y analizar</i>	Analizar resultados obtenidos.
<i>Iniciar la siguiente vuelta</i>	Aplica solo, de considerarse necesario

Tabla 21 Pasos del método Delphi

A continuación, la Tabla 22 define los criterios de evaluación de expertos.

Banco de Preguntas.

Ítem	Parámetros	Descripción del parámetro	1	2	3	4	5
1	Funcionalidad	Capacidad de la metodología para brindar funciones que cumplan necesidades específicas con ciertos requerimientos					
2	Fiabilidad	Capacidad de definir lenguajes de programación bajo criterios de trabajo y requerimientos específicos					
3	Usabilidad	Capacidad de aceptación del producto por parte de los institutos de educación tecnológica superior					

4	Eficiencia	Capacidad de efectividad al momento de establecer un lenguaje de programación idóneo
5	Mantenibilidad	Capacidad de la metodología de adaptación a cambios o mejoras
6	Portabilidad	Capacidad de la metodología de ser implementada en otro tipo de instituciones educativas
7	Aspectos Técnicos	Características a nivel hardware y software que provean un entorno de trabajo eficiente al momento de la implementación
8	Aspectos pedagógicos	Características de curva de crecimiento de aprendizaje al momento de aplicar conceptos básicos y fundamentales de la POO. Evaluar los conocimientos a través de técnicas medibles.
9	Aspectos comparativos	Capacidad de realizar analogías técnicas y medibles en cuanto a comparativa entre lenguajes de programación
10	Aspectos Administrativos	Capacidad de mejoras o cambios con respecto al tiempo y costes.

Tabla 22 Definición de criterios de evaluación de expertos

La siguiente Tabla 23, muestra la escala de valoración utilizada.

Escala de valoración

- 1 No adecuado
- 2 Poco Adecuado
- 3 Adecuado

- 4 Bastante adecuado
- 5 Muy adecuado

Tabla 23 Escala de valoración

Luego de realizar la lista de parámetros a ser considerados por parte de los expertos, se describe cada uno de ellos para una mejor explicación y comprensión de los expertos. Se procede a enviar el diseño de la metodología y los resultados junto con los cuestionarios a los expertos para que realizar la evaluación de la misma. Posterior a este proceso se procede a tabular la información y exponerla a través de Ilustración 44 para una mejor explicación.

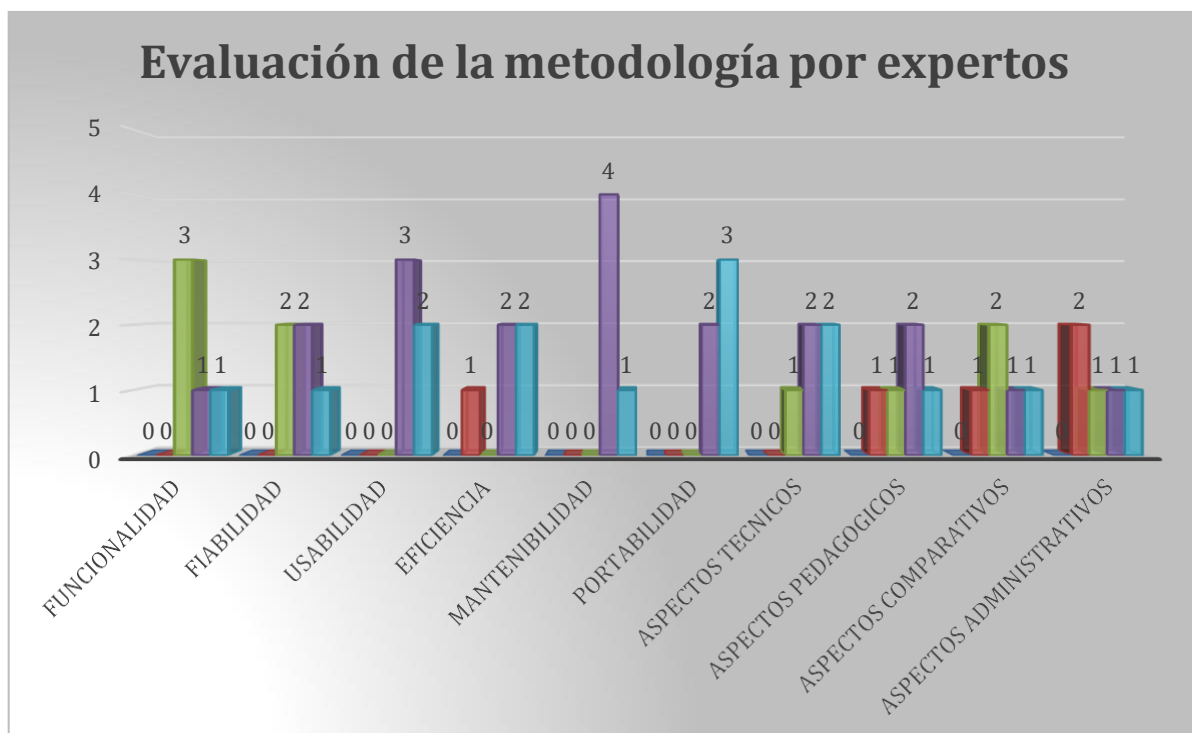


Ilustración 44 Evaluación de la metodología por parte de expertos

Luego de la aplicación del cuestionario de evaluación por parte de expertos en el tema, se puede evidenciar en el gráfico una acogida bastante positiva de la evaluación de la metodología. Esta gráfica permite destacar la aceptación en parámetros como funcionalidad, usabilidad, mantenibilidad, portabilidad.

El estudio presenta un punto de discusión con la muestra de resultados con la que se puede contrastar los datos sin uso de la metodología y una mejora considerable posterior a la implementación de una metodología de selección. De igual forma a esto se le suma el análisis técnico por parte de expertos en el ámbito de desarrollo, los cuales evalúan y determinan un rango aceptable de aceptación de la metodología.

Conclusiones

- ✓ Con el estudio se logró establecer una metodología de selección de lenguajes de programación orientados a objetos para desarrollar software en un ambiente de enseñanza aprendizaje presencial. Basado en tendencias actuales de programación y evaluando la implementación de la misma.
- ✓ La metodología de selección de lenguajes de programación es una propuesta importante a la hora de iniciar un proceso de enseñanza-aprendizaje de manera efectiva, ya que permite analizar parámetros para una selección idónea de la herramienta.
- ✓ Seguir un proceso sistemático y metodológico a la hora de realizar una investigación permite fundamentar adecuadamente el planteamiento de un problema y encaminar la solución de manera eficiente.
- ✓ El definir herramientas de recolección de datos adecuado, garantiza un proceso de tabulación y análisis de datos efectivo a la hora de presentar gráficos estadísticos.
- ✓ Es interesante desatacar que más allá de la metodología, hoy en día Python se presenta como un lenguaje fácil, claro y eficiente a la hora de programar. Sumado a esto según tendencias de programación, Python ha tenido una curva de crecimiento y aceptación más grande en comparación con otras y continua en constante crecimiento.

Recomendaciones

- ✓ El estudio aplicado al establecer y evaluar la metodología, determina procesos de análisis previo y post implementación de la misma, para de esta manera el enfocar el cumplimiento de objetivos y determinar la eficiencia del proceso.
- ✓ Es esencial a la hora de entrar en el mundo de la programación el tener una visión clara del enfoque que se busca explotar, en este sentido el desarrollo puede ser de escritorio, web, móvil, híbrido y en la nube.
- ✓ Es interesante el tomar artículos científicos actualizados que se enfoquen en el proceso de revisión sistemática de la literatura, ya que presenta de forma clara y detallada los pasos a seguir para un mejor resultado.
- ✓ El contexto de la investigación define las herramientas idóneas al momento de recolectar información, esto permitirá que el tratamiento de la información sea más minucioso y efectivo a la hora de la implementación de una solución.
- ✓ A la hora de seleccionar un lenguaje como Python es fundamental conocer más allá de las prestaciones básicas como lenguaje y profundizar en cuanto a la realización de tareas mucho más elaboradas y que aporten un ámbito de estudio de manera significativa como estadística, inteligencia, entre otras.

Bibliografía:

- [1] Información Estadística sobre Educación Superior, Ciencia, Tecnología e Innovación – Senescyt – Secretaría de Educación Superior, Ciencia, Tecnología e Innovación. <https://www.educacionsuperior.gob.ec/informacion-estadistica-sobre-educacion-superior-ciencia-tecnologia-e-innovacion/> (accedido jul. 10, 2020).
- [2] Senescyt – Secretaría de Educación Superior, Ciencia, Tecnología e Innovación. <https://www.educacionsuperior.gob.ec/propuesta-reforma-loes-entregada-comision-educacion-asamblea/> (accedido jul. 10, 2020).
- [3] Cifras a nivel nacional y provincial de la oferta académica, acceso y permanencia en el sistema de educación superior – Senescyt – Secretaría de Educación Superior, Ciencia, Tecnología e Innovación. <https://www.educacionsuperior.gob.ec/cifras-a-nivel-nacional-y-provincial-de-la-oferta-academica-acceso-y-permanencia-en-el-sistema-de-educacion-superior/> (accedido jul. 14, 2020).
- [4] R. Pressman, El software y la ingeniería de software, *RS Press. Ing. Softw. Un Enfoque Prácticopág 12 N. Y. McGraw-Hill*, 2010.
- [5] J. McAlister, Avances en pro de la profesionalización de la Ingeniería de Software., *Rev. Antioqueña Las Cienc. Comput.*, vol. 4, n.º 2, 2014.
- [6] Y. S. Pellicer, M. C. Zea, R. A. Pérez, Y. C. Blanco, y M. G. L. Brito, visualización dinámica, una opción para la enseñanza-aprendizaje de la programación de computadoras, *holos*, vol. 2, pp. 1-20, 2020.
- [7] M. Y. L. Vazquez, Estudio y propuesta metodológica para la enseñanza-aprendizaje de la programación informática en la educación superior, *Rev. Dilemas Contemp. Educ. Política Valores*, n.º 8, 2019.
- [8] AESOFT presentó el estudio de Hardware y Software en el Ecuador, *Ekos Negocios*. <https://www.ekosnegocios.com/articulo/aesoft-presento-el-estudio-de-hardware-y-software-en-el-ecuador> (accedido jul. 10, 2020).
- [9] J. C. G. Monsálvez, Python como primer lenguaje de programación textual en la Enseñanza Secundaria, *Educ. Knowl. Soc.*, vol. 18, n.º 2, pp. 147-162, 2017.

- [10]J. Gmys, T. Carneiro, N. Melab, E.-G. Talbi, y D. Tuyttens, A comparative study of high-productivity high-performance programming languages for parallel metaheuristics, *Swarm Evol. Comput.*, p. 100720, 2020.
- [11]R. Peña Ros, Python como primera aproximación a la programación, *ReVisión*, vol. 8, n.º 2, 2015.
- [12]I. Boada, J. Soler, F. Prados, y J. Poch, A teaching/learning support tool for introductory programming courses, en *Information Technology Based Proceedings of the Fifth International Conference on Higher Education and Training, 2004. ITHET 2004.*, 2004, pp. 604-609.
- [13]A. F. Szpiniak y G. A. Rojo, Enseñanza de la programación, *Rev. Iberoam. Tecnol. En Educ. Educ. En Tecnol.*, n.º 1, pp. 8-p, 2006.
- [14]M. McCracken *et al.*, A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, en *Working group reports from ITiCSE on Innovation and technology in computer science education*, 2001, pp. 125-180.
- [15]J. G. Ruiz, M. H. López, y J. A. L. Brito, Pensamiento sistémico y desarrollo de competencias, en el aprendizaje de los lenguajes de programación, *ANFEI Digit.*, n.º 2, 2015.
- [16]P. Ferrara, A generic framework for heap and value analyses of object-oriented programming languages, *Theor. Comput. Sci.*, vol. 631, pp. 43-72, 2016.
- [17]O. Solarte y L. E. M. Villegas, Fortaleciendo la motivación y mejorando el rendimiento de estudiantes de un curso introductorio de programación: Un enfoque de enseñanza integrado, *Rev. EIA*, vol. 16, n.º 31, pp. 65-76, 2019.
- [18]R. Pressman, *Ingeniería del Software Un Enfoque Práctico. 7ma ed. University of Connecticut*. SA, 2010.
- [19]T. Henwood, S. Channon, H. Penny, M. Robling, y C. S. Waters, Do Home Visiting Programmes improve children's language development? A systematic review, *Int. J. Nurs. Stud.*, p. 103610, 2020.
- [20] K. J. Goldman, An interactive environment for beginning Java programmers, *Sci. Comput. Program.*, vol. 53, n.º 1, pp. 3-24, 2004.

- [21]R. Z. Cabada, M. L. B. Estrada, F. G. Hernández, R. O. Bustillos, y C. A. Reyes-García, An affective and Web 3.0-based learning environment for a programming language, *Telemat. Inform.*, vol. 35, n.º 3, pp. 611-628, 2018.
- [22]M. del Pilar Salas-Zárate, G. Alor-Hernández, R. Valencia-García, L. Rodríguez-Mazahua, A. Rodríguez-González, y J. L. L. Cuadrado, Analyzing best practices on Web development frameworks: The lift approach, *Sci. Comput. Program.*, vol. 102, pp. 1-19, 2015.
- [23]Y. Ocaña-Fernández, L. A. Valenzuela-Fernández, y L. L. Garro-Aburto, Inteligencia artificial y sus implicaciones en la educación superior, *Propósitos Represent.*, vol. 7, n.º 2, pp. 536-568, 2019.
- [24]S. F. Morales, J. L. H. Hernández, y R. E. C. Valencia, Interoperabilidad entre lenguajes de programación, *Rev. Vínculos*, vol. 8, n.º 1, pp. 184-193, 2011.
- [25] A. E. C. González, *Programación orientada a objetos*. Recuperado el, 2018.
- [26]Y. P. Pérez y L. M. López, Multiparadigma en la enseñanza de la programación, 2007.
- [27]B. A. Myers, Taxonomies of visual programming and program visualization, *J. Vis. Lang. Comput.*, vol. 1, n.º 1, pp. 97-123, 1990.
- [28]A. Leitão, L. Santos, y J. Lopes, Programming languages for generative design: a comparative study, *Int. J. Archit. Comput.*, vol. 10, n.º 1, pp. 139-162, 2012.
- [29]F. I. Anfurrutia, A. Álvarez, M. Larrañaga, y J.-M. López-Gil, Entornos de Programación Visual para Programación Orientada a Objetos: Aceptación y Efectos en la Motivación de los Estudiantes, *Rev Iberoam Tecnol Aprendiz*, vol. 5, n.º 1, pp. 11-18, 2017.
- [30]L. L. ROMÁN, Metodología para el Desarrollo de la Lógica de la Programación Orientada a Objetos.
- [31]A. O. Ramirez, Python como primer lenguaje de programación, *Publ. Interna Tecnológico Monterrey Campus Estado México*, 2010.
- [32]S. H. Edwards, D. S. Tilden, y A. Allevato, Pythy: improving the introductory python programming experience, en *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 641-646.

- [33]Z. Chen, L. Chen, W. Ma, X. Zhou, Y. Zhou, y B. Xu, Understanding metric- based detectable smells in Python software: A comparative study, *Inf. Softw. Technol.*, vol. 94, pp. 14-29, 2018.
- [34]M. G. Benedetto *et al.*, Selección de lenguajes orientados a objetos para un estudio comparativo y análisis de rendimiento, 2015.
- [35]A. A. B. Abdelnabi, An Analytical Hierarchical Process Model to Select Programming Language for Novice Programmers for Data Analytics Applications, en *2019 International Arab Conference on Information Technology (ACIT)*, 2019, pp. 128-132.
- [36]S. Nanz y C. A. Furia, A comparative study of programming languages in rosetta code, en *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, vol. 1, pp. 778-788.
- [37]M. Y. Mahmoud y A. P. Felty, Formal meta-level analysis framework for quantum programming languages, *Electron. Notes Theor. Comput. Sci.*, vol. 338, pp. 185-201, 2018.
- [38]I. Challenger-Pérez, Y. Díaz-Ricardo, y R. A. Becerra-García, El lenguaje de programación Python, *Cienc. Holguín*, vol. 20, n.º 2, pp. 1-13, 2014.
- [39]K. R. Parker, T. A. Ottaway, y J. T. Chao, Criteria for the selection of a programming language for introductory courses, *Int. J. Knowl. Learn.*, vol. 2, n.º 1-2, pp. 119-139, 2006.
- [40]J. O. Ogala y D. V. Ojie, comparative analysis of c, c++, c# and java programming languages, *GSJ*, vol. 8, n.º 5, 2020.
- [41]F. L. Khaleel, N. S. Ashaari, T. S. M. T. Wook, y A. Ismail, Methodology for developing gamification-based learning programming language framework, en *2017 6th international conference on electrical engineering and informatics (iceei)*, 2017, pp. 1-6.
- [42]A. Rojas-López y F. J. García-Peñalvo, Evaluación del pensamiento computacional para el aprendizaje de programación de computadoras en educación superior, *Rev. Educ. Distancia*, vol. 20, n.º 63, 2020.

- [43]K. R. Parker, T. A. Ottaway, y J. T. Chao, Criteria for the selection of a programming language for introductory courses, *Int. J. Knowl. Learn.*, vol. 2, n.º 1-2, pp. 119-139, 2006.
- [44]D. Weintrop y U. Wilensky, Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms, *Comput. Educ.*, vol. 142, p. 103646, 2019.
- [45]N. Monjelat y G. Rodríguez, Repensando la programación como formación práctica en Ingeniería: Un estudio de caso en primer año, *Ingeniare Rev. Chil. Ing.*, vol. 26, n.º 1, pp. 172-183, 2018.
- [46]R. Lister *et al.*, A multi-national study of reading and tracing skills in novice programmers, *ACM SIGCSE Bull.*, vol. 36, n.º 4, pp. 119-150, 2004.
- [47]K. I. Díaz Tejera, E. Fierro Martín, y M. A. Muñoz Pentón, La enseñanza de la programación: una experiencia en la formación de profesores de informática, *Educación*, vol. 27, n.º 53, pp. 73-91, 2018.
- [48]M. N. Solís Sierra y P. L. Duarte Morante, La educación superior tecnológica y la empleabilidad, *Rev. Univ. Soc.*, vol. 10, n.º 3, pp. 21-33, 2018.
- [49]E. López-Gómez, El método Delphi en la investigación actual en educación: una revisión teórica y metodológica, *Educ. XXI*, vol. 21, n.º 1, pp. 17-40, 2018.

Anexos Mallas Curriculares

MALLA CURRICULAR - TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE																																										
Semestre Académico	Nivel de Formación Profesional	Semestre	FUNDAMENTOS TEÓRICOS					Semestre	ASIGNACIONES E INNOVACIÓN TECNOLÓGICA					COMUNICACIÓN Y LENGUAJES					Semestre	INTEGRACIÓN DE SABERES, CONTENIDOS Y CULTURAS					Prácticas Pre-Profesionales				TOTAL													
			Asignatura	Teoría	Prácticas	Laboratorio	Total		Asignatura	Teoría	Prácticas	Laboratorio	Total	Asignatura	Teoría	Prácticas	Laboratorio	Total		Asignatura	Teoría	Prácticas	Laboratorio	Total	Vinculadas con la actividad	Prácticas Pre-Prof.	Total	Total componente docente	Total componente Prácticas (Prácticas Pre-profesionales)	Total Componente Académico	Total período académico											
1	BÁSICA	1	Matemática Discreta	54	38	50	122	2	Fundamentos de Programación	72	54	60	186	1	Inglés A1(Básico)	54	36	30	120																							
		3	Desarrollo del Pensamiento	54	0	40	94	5	Análisis y Diseño de Sistemas	72	36	50	158																													
		4	Introducción al Desarrollo de Software	54	36	40	130																																			
2	PROFESIONAL	7	Álgebra y Trigonometría	54	38	50	122	8	Base de Datos	72	54	70	196	2	Lenguaje y Comunicación	54	0	30	84																							
								9	Programación Orientada a Objetos	72	54	70	196																													
								10	Metodologías de Desarrollo de Software	54	18	20	92																													
3	PROFESIONAL	13	Cálculo Diferencial e Integral	54	38	50	122	14	Programación Visual	72	54	70	196	3	Inglés B1.1(Technical)	54	36	30	120																							
			Fundamentos de Administración	54	0	30	84	15	Base de Datos Avanzada	72	36	60	168																													
								17	Diseño de Interfaz	54	36	30	120																													
4	PROFESIONAL	19	Estadística Descriptiva	54	38	30	102	20	Programación de Aplicaciones Web	72	72	50	194	4	Inglés B1.2(Specific purpose)	54	36	30	120	22	Diversidad y Cultura	36	0	30	66																	
			Legislación Informática	72	0	20	92	21	Desarrollo de Aplicaciones Móviles	72	54	50	176																													
5	TITULACIÓN	28	Emprendimientos	54	38	28	100	25	Proyecto de Titulación	36	0	204	240																													
								26	Tendencias Actuales de Programación	90	72	40	202																													
								27	Calidad del Software	72	36	40	148																													
								30	Fundamentos de Redes y Conectividad	72	54	40	166																													
TOTAL HORAS CURRÍCULO																																										
			504	126	338	968			954	630	854	2.438		270	144	150	564			72	0	58	130	160	240	400	1.800	1.500	1.400	4.500												

Ilustración 14 Malla curricular 1. Juan Bautista Vásquez / Azuay (JBV/AZ)

PERIODO ACADÉMICO	UNIDADES DE ORGANIZACIÓN CURRICULAR	FUNDAMENTOS TÉCNICOS				ADAPTACIÓN E INNOVACIÓN TECNOLÓGICA				COMUNICACIÓN Y LENGUAJES				INTEGRACIÓN DE SABERES, CONTEXTOS Y CULTURAS				Trabajo de titulación	Prácticas Pre-Profesionales		TOTAL																			
		ASIGNATURA	Docencia	Práctica	Autónomo	Total	ASIGNATURA	Docencia	Práctica	Autónomo	Total	ASIGNATURA	Docencia	Práctica	Autónomo	Total	ASIGNATURA		Docencia	Práctica	Autónomo	Total	Vinculación con la sociedad	Prácticas Pre-Prof.	Total componentes docentes	Total componente práctico Instituto	Total Componente Autónomo	Total período académico												
1	BÁSICA	1 Matemática Discreta	54	18	50	122	3	Fundamentos de Programación	72	54	60	186	5	Desarrollo del Pensamiento	54	0	40	94																						
		2 Introducción al Desarrollo de Software	54	36	40	130	4	Análisis y Diseño de Sistemas	72	36	50	158	6	Inglés A1(Acceso)	54	36	30	120																						
2	PROFESIONAL	7 Álgebra y Trigonometría	54	18	50	122	8	Base de Datos	72	54	70	196	11	Lenguaje y Comunicación	54	0	30	84																						
							9	Programación Orientada a Objetos	72	54	70	196	12	Inglés A2(plataforma)	54	36	30	120																						
							10	Metodologías de Desarrollo de Software	54	18	20	92																												
3	PROFESIONAL	13 Cálculo Diferencial e Integral	54	18	50	122	15	Programación Visual	72	54	70	196	18	Inglés B1.1(Intermedio)	54	36	30	120																						
		14 Fundamentos de Administración	54	0	30	84	16	Base de Datos Avanzada	72	36	60	168																												
4	PROFESIONAL						17	Diseño Multimedia	54	36	30	120											0	120	360	180	270	890												
		19 Estadística Descriptiva	54	18	30	102	21	Programación de Aplicaciones Web	72	72	50	194	23	Inglés B1.2(Intermedio alto)	54	36	30	120	24	Diversidad y Cultura	36	0	30	66																
5	TITULACIÓN	20 Legislación Informática	72	0	20	92	22	Desarrollo de Aplicaciones Móviles	72	54	50	176											0	120	360	180	210	870												
		25 Calidad del Software	72	36	40	148	27	Fundamentos de Redes y Conectividad	72	54	40	166						29	Ética profesional	36	0	28	64																	
		26 Emprendimientos	54	18	28	100	28	Tendencias Actuales de Programación	90	72	40	202																												
							Proyecto de Titulación					240							240	0	0	324	180	176	920															
TOTAL HORAS CURRÍCULO						522	322	338	1,022					846	594	610	2,290					324	144	190	658					72	0	58	130	240	160	240	1764	900	1196	4500

Ilustración 15 Malla curricular 2. Enrique Noboa Arizaga (ENA)

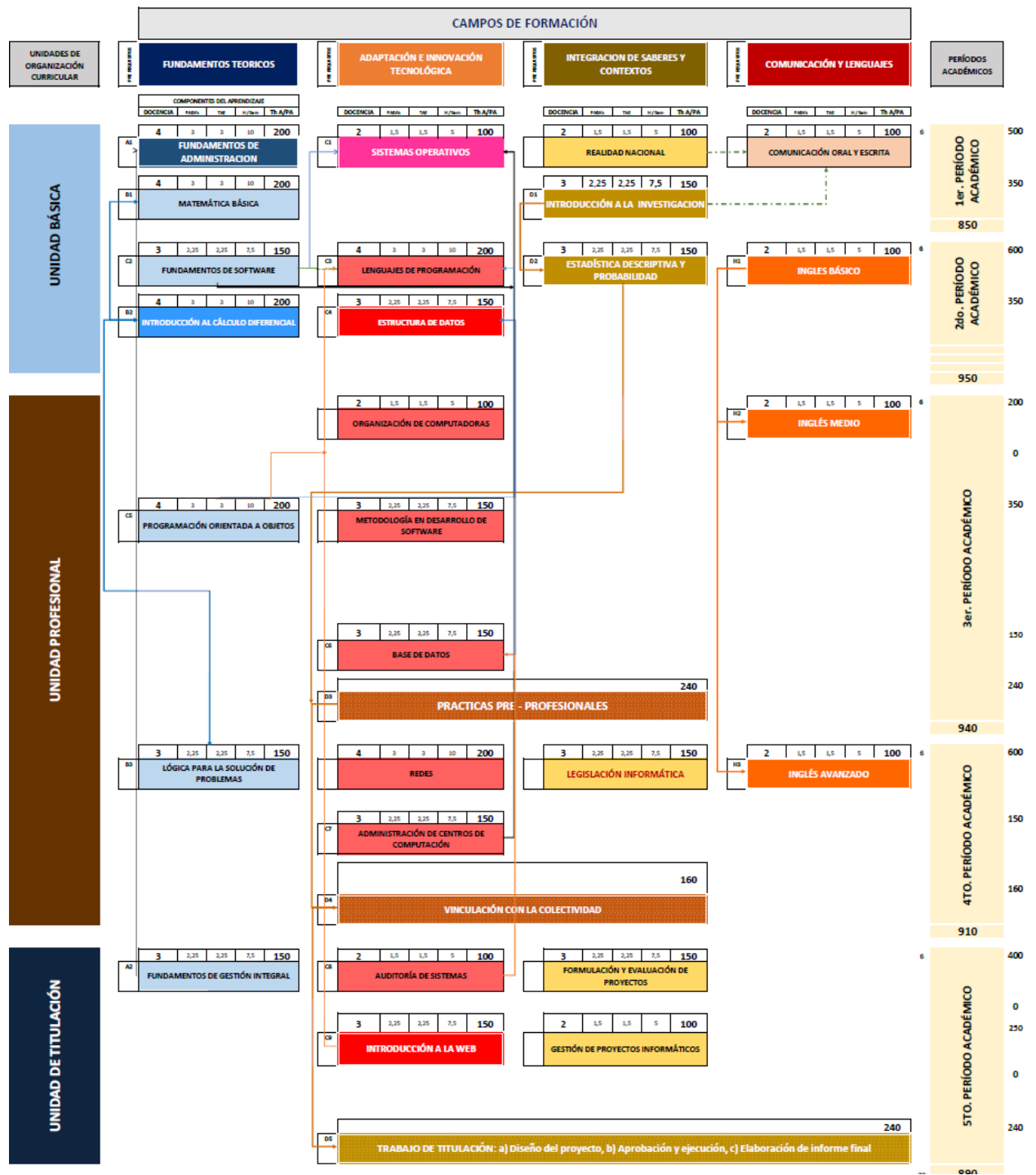


Ilustración 16 Malla curricular 3. Integración Andina (INAN)

NIVEL	ASIGNATURA	EJE FORMATIVO	CÓDIGO	HORAS
1ER NIVEL	PROYECTO INTEGRADOR DE SABERES: CREACIÓN DE APLICACIONES EN CONSOLA	BASICA	TG-DS-BISCC-01-5	36
1ER NIVEL	METODOLOGÍAS PARA RESOLVER PROBLEMAS INFORMÁTICOS	BASICA	TG-DS-BAIT-01-4	72
1ER NIVEL	LAS TICs Y SOPORTE EN HARDWARE	BASICA	TG-DS-BAIT-01-2	72
1ER NIVEL	MATEMÁTICA COMPUTACIONAL APLICADA	BASICA	TG-DS-BFT-01-1	72
1ER NIVEL	SISTEMAS OPERATIVOS	BASICA	TG-DS-BAIT-01-3	72
1ER NIVEL	COMUNICACIÓN ORAL Y ESCRITA	BASICA	TG-DS-BCL-01-6	36
				360
2DO NIVEL	ECOLOGIA	PROFESIONAL	TG-DS-PCL-02-6	36
2DO NIVEL	PROCESOS CONTABLES	PROFESIONAL	TG-DS-PFT-02-1	72
2DO NIVEL	PROGRAMACIÓN DE INTERFASES GRÁFICAS Y ESTRUCTURA DE DATOS	PROFESIONAL	TG-DS-PAIT-02-3	72
2DO NIVEL	PROYECTO INTEGRADOR DE SABERES: CREACIÓN DE APLICACIONES DE ESCRITORIO	PROFESIONAL	TG-DS-PISCC-02-5	36
2DO NIVEL	COMUNICACIONES Y REDES DE DATOS	PROFESIONAL	TG-DS-PAIT-02-2	72
2DO NIVEL	SISTEMAS DE INFORMACIÓN	PROFESIONAL	TG-DS-PAIT-02-4	72
				360
3ER NIVEL	PROBABILIDADES Y PROCESOS ESTOCÁSTICOS	PROFESIONAL	TG-DS-PFT-03-1	72
3ER NIVEL	BASE DE DATOS RELACIONALES	PROFESIONAL	TG-DS-PAIT-03-2	72
3ER NIVEL	HERRAMIENTAS INFORMÁTICAS PARA EL DESPLIEGUE DE DIAGRAMAS	PROFESIONAL	TG-DS-PAIT-03-4	72
3ER NIVEL	PROYECTO INTEGRADOR DE SABERES: CREACIÓN DE APLICACIONES DE ESCRITORIO CON BASE DE DATOS	PROFESIONAL	TG-DS-PISCC-03-6	36
3ER NIVEL	SOFTWARE APLICATIVO	PROFESIONAL	TG-DS-PAIT-03-5	36
3ER NIVEL	PROGRAMACIÓN ORIENTADA A OBJETOS	PROFESIONAL	TG-DS-PAIT-03-3	72
				360
4TO NIVEL	SISTEMAS AGILES	PROFESIONAL	TG-DS-PAIT-04-5	72
4TO NIVEL	PROYECTO INTEGRADOR DE SABERES: CREACIÓN DE APLICACIONES WEB CON BASE A LA ARQUITECTURA CLIENTE SERVIDOR	PROFESIONAL	TG-DS-PISCC-04-6	36
4TO NIVEL	SISTEMAS DIGITALES PROGRAMABLES	PROFESIONAL	TG-DS-PAIT-04-4	72
4TO NIVEL	PROGRAMACION DE APLICACIONES WEB	PROFESIONAL	TG-DS-PAIT-04-3	72
4TO NIVEL	GESTION DE BASE DE DATOS	PROFESIONAL	TG-DS-PAIT-04-2	72
4TO NIVEL	DERECHO Y SEGURIDAD INFORMATICA	PROFESIONAL	TG-DS-PFT-04-1	36
				360
5TO NIVEL	PROYECTO DE TITULACION	TITULACION	TG-DS-PT-05-6	0
5TO NIVEL	APLICACIONES TECNOLÓGICAS AUTÓNOMAS	TITULACION	TG-DS-TAIT-05-1	72
5TO NIVEL	HERRAMIENTAS CASE	TITULACION	TG-DS-TCL-05-6	72
5TO NIVEL	ÉTICA PROFESIONAL	TITULACION	TG-DS-TAIT-05-3	72
5TO NIVEL	PROYECTO DE DESARROLLO EMPRESARIAL	TITULACION	TG-DS-TAIT-05-2	72
5TO NIVEL	PROGRAMACION DE APLICACIONES PARA DISPOSITIVOS MOVILES	TITULACION		72
				360

Ilustración 17 Malla curricular 4. Sudamericano (SUDA)