ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Diseño de un modelo de trabajo basado en las metodologías tradicionales y no tradicionales implementando las mejores prácticas de la industria de desarrollo de software en la empresa de ACR DIGITAL S.A.S.

Llamuca Manzano, Andrea Lizbeth y Pérez Guanopatin, Lorena Mishel

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Software

Trabajo de titulación, previo a la obtención del título de Ingeniero en Software

Ing. Espinosa Gallardo, Edison Gonzalo

AGOSTO, 2023

Latacunga

Copyleaks

Plagiarism report

Llamuca_Perez_Tesis_final_f.docx

Scan details

Scan time: February 15th, 2024 at 14:32 UTC

Total Pages: 58

Total Words: 14473

Plagiarism Detection

Types of plagiarism		Words
Identical	1.2%	167
Minor Changes	0%	0
Paraphrased	0%	0
Omitted Words	4.7%	681

AI Content Detection

Text coverage
- AI text
- Human text

Plagiarism Results: (23)

Control Estadístico de Calidad y Seis Sigma 0.7%

https://www.uv.mx/personal/ermeneses/files/2018/05/6-control-estadistico-de-la-calidad-y-seis-sigma-gutierr...

Humberto Gutiérrez Pulido

CONTROL ESTADÍSTICO DE CALIDAD Y SEIS SIGMA CONTROL ESTADÍSTICO DE CALIDAD Y SEIS SIGMA Segunda edición Humberto Gutiérrez Pulido Centr...

SORYP.pdf 0.4%

https://oa.upm.es/36552/1/soryp.pdf

Elementos de sistemas operativos, de representación de la información y de procesadores hardware y software 1001 110010111 1001101100011...

T00313.pdf 0.2%

https://repository.icesi.edu.co/biblioteca_digital/bitstream/10906/77747/1/t00313.pdf

Hugo Arboleda

Proceso de Pruebas Unitarias Bajo Entornos de Desarrollo Ágiles: Un Estudio Sistemático TRABAJO DE GRADO José Andrés Moncada Quintero D...

31072019-Informe-Final-Conicyt.pdf 0.2%

https://www.conicyt.cl/wp-content/uploads/2014/07/31072019-informe-final-conicyt.pdf

Ricardo Ruiz

Informe Final SEGUNDA ENCUESTA DE PERCEPCIÓN Y APROPIACIÓN SOCIAL DE LA CIENCIA Y LA TECNOLOGÍA EN CHILE Julio 2019 1 Contenido ANTECED...

Ing. Espinosa Gallardo, Edison Gonzalo
C.C.: 0501578910

Certified by Copyleaks

About this report
help.copyleaks.com

copyleaks.com

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Software

Certificación

Certifico que el trabajo de titulación: **"Diseño de un modelo de trabajo basado en las metodologías tradicionales y no tradicionales implementando las mejores prácticas de la industria de desarrollo de software en la empresa de ACR DIGITALS S.A.S."** fue realizado por la señora **Llamuca Manzano, Andrea Lizbeth** y la señorita **Pérez Guanopatin, Lorena Mishel;** el mismo que cumple con los requisitos legales, teóricos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Latacunga, 15 de febrero de 2024

Ing. Espinosa Gallardo, Edison Gonzalo

C.C.: 0501578910

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Software

Autorización de publicación

Nosotras, **Llamuca Manzano, Andrea Lizbeth,** con cédula de ciudadanía No. **1803596525** y **Pérez Guanopatin, Lorena Mishel,** con cédula de ciudadanía No. **1724042468,** autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en trabajo de titulación, **"Diseño de un modelo de trabajo basado en las metodologías tradicionales y no tradicionales implementando las mejores prácticas de la industria de desarrollo de software en la empresa de ACR DIGITALS S.A.S.",** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Latacunga, 15 de febrero de 2024

Llamuca Manzano, Andrea Lizbeth
C.C.: 1803596525

Pérez Guanopatin, Lorena Mishel
C.C.: 1724042468

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Software

Responsabilidad de autoría

Nosotras, **Llamuca Manzano, Andrea Lizbeth,** con cédula de ciudadanía No. **1803596525** y **Pérez Guanopatin, Lorena Mishel,** con cédula de ciudadanía No. **1724042468,** declaramos que el contenido, ideas y criterios del trabajo de titulación: , **"Diseño de un modelo de trabajo basado en las metodologías tradicionales y no tradicionales implementando las mejores prácticas de la industria de desarrollo de software en la empresa de ACR DIGITALS S.A.S.",** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, 15 de febrero de 2024

Llamuca Manzano, Andrea Lizbeth

C.C.: 1803596525

Pérez Guanopatin, Lorena Mishel

C.C.: 1724042468

Dedicatoria

Este trabajo de titulación se lo dedico a Dios y a toda mi familia. Primero a Dios que me ha permitido culminar mi carrera universitaria en conjunto con esta tesis, a mi hija Anahí, que a pesar de todos los tropiezos es quien está a mi lado siempre, a mi esposo, que ha estado conmigo apoyándome durante estos años juntos y siempre me da ánimos para seguir adelante.

A mis padres Fernando y Silvanía, quienes me han sabido apoyar en todo momento en todas las etapas de mi vida, en los malos y buenos momentos. Gracias por enseñarme a siempre salir adelante, enseñarme valores y actuar siempre con honestidad, mediante su esfuerzo y perseverancia diariamente y a mi hermano que es un pilar fundamental en mi vida.

Andrea Lizbeth Llamuca Manzano

Dedicatoria

Esta tesis se la dedico:

A Dios y a la Virgencita quienes me han dado salud, vida y me permiten haber llegado hasta este momento tan importante de mi vida.

A mis padres Alicia y Pablo quienes, con amor, paciencia, esfuerzo me han permitido llegar a cumplir hoy un sueño más, por haberme apoyado en todo momento a pesar de las circunstancias que se han presentado nunca me han dejado sola, siempre han sabido alentarme, darme fuerzas para salir adelante por ser un pilar muy importante en mi vida.

A mis hermanos Pablo e Israel por brindarme su apoyo, cariño y estar conmigo en las buenas y en las malas.

A mi esposo Edison y a mis hijas Nicol y Zoé por siempre apoyarme, por estar conmigo siempre a pesar de todos los momentos difíciles que hemos pasado, por ser el principal motor en mi vida, por ser mi alegría y porque siempre han sabido confiar en mí.

Lorena Mishel Pérez Guanopatín

.

Agradecimiento

Una vez finalizado un trabajo arduo y con muchas dificultades como es el desarrollo de la tesis, es muy importante para mí y un verdadero placer extender un sincero agradecimiento a aquellas personas y seres que con poco o mucho han sido parte de este proceso. Debo agradecer de manera sincera y especial primero a Dios que ha sido quien ha cuidado de mí en esta etapa y me ha permitido llegar a culminar este proceso. Su apoyo incondicional y la confianza a mi familia, un pilar muy importante en mí, como es mi esposo y mi nena, que han sido mi mayor motivación.

Quiero expresar también mi más sincero agradecimiento a mis padres que nunca se dieron por vencidos y siempre lucharon hasta que yo llegue a este momento, me han guiado y me dieron la oportunidad de seguirme superando en mi vida.

Al P.h.d Edison Gonzalo Espinosa Gallardo y al Ing. Javier Montaluisa por ser una guía en el proceso de tesis y como profesores, que gracias a su apoyo hoy estoy en un paso a culminar mi pregrado.

Mi agradecimiento también a la Universidad de las Fuerzas Armadas ESPE Extensión Latacunga y en especial al Departamento de Ciencias de la Computación, por permitirme continuar con mis estudios y en un futuro ser una Ingeniera.

Andrea Lizbeth Llamuca Manzano

Agradecimiento

A Dios, por la salud, energía y por darme la sabiduría para poder culminar con mi tesis.

Agradezco a mis padres y hermanos por todo el apoyo brindado en todo el transcurso de mis estudios por darme fuerza para poder seguir adelante a pesar de los tropiezos, por los valores que han sabido inculcarme y el sacrificio que han hecho en todo este tiempo para darme mis estudios

A mis suegros y a mis cuñadas/os por haberme apoyado en mis estudios a pesar de todos los malentendidos que hubo entre nosotros siempre estuvieron dispuestos a ayudarme cuando más los necesitaba, por ayudarme en el cuidado de mi nena para así poder ir a estudiar.

Agradezco a mi esposo por haberme tenido mucha paciencia, por siempre apoyarme y estar cuando más lo he necesitado y a mis hijas por ser mi motivo de esfuerzo y sacrificio por ser las personas que alegran mi vida con sus ocurrencias y porque son el pilar fundamental de mi vida

Agradezco a los ingenieros que fueron parte de mi experiencia como estudiante por brindarme su sabiduría y conocimientos necesarios para poder defenderme en la vida como profesional, en especial al Ing. Fabian Montaluisa por siempre alentarme a seguir adelante, por sus consejos y enseñanzas. Al P.h.d Edison Espinosa por su ayuda y apoyo en el proceso de la tesis.

Lorena Mishel Pérez Guanopatín

ÍNDICE DE CONTENIDOS

Carátula. 1

Reporte de verificación de contenido..... 2

Certificación 3

Autorización de publicación 4

Responsabilidad de autoría 5

Dedicatoria 6

Dedicatoria 7

Agradecimiento 8

Agradecimiento 9

Índice de contenidos 10

Índice de tablas 16

Índice de figuras 18

Resumen..... 20

Abstract 21

Capítulo I: Presentación del Problema 22

Introducción..... 22

Antecedentes..... 22

Planteamiento y Formulación del Problema..... 24

Justificación e Importancia 25

Objetivos..... 26

Objetivo General..... 26

Objetivos Específicos 26

Hipótesis **26**

Variables de la Investigación **27**

Variable Dependiente ***27***

Variable Independiente ***27***

Conceptualización de la Variable Independiente **27**

Indicadores **27**

Capítulo II: Marco Teórico **28**

Introducción **28**

Antecedentes Históricos **28**

Primera etapa cronológica (1940-1970) **28**

Segunda etapa cronológica (1970-2000) **29**

Modelo de desarrollo de software ***30***

Metodología de desarrollo de software: ***30***

Tercera etapa cronológica (2000-Presente) **32**

Antecedentes conceptuales y referenciales ***34***

Software **34**

Ingeniería de Software **34**

Desarrollo de software. **35**

Proceso de desarrollo de software. **35**

Modelos de desarrollo de software. **37**

Modelo en Cascada **37**

Prototipo **38**

Reutilización o Desarrollo basado en componentes **39**

Desarrollo en espiral **39**

Modelo RAD. …… 39

Lenguaje Unificado de Modelado UML…… 40

Metodología…… 41

Metodologías de desarrollo de software. ……... 41

Metodologías Tradicionales.…… 42

Metodologías no tradicionales. …… 43

Capítulo III: Análisis, diseño y desarrollo del sistema web 44

Introducción del Capítulo 44

Extracción de artículos científicas 45

Fase de Planeación 45

Descripción del contexto de la investigación 45

Estrategia de búsqueda bibliográfica 45

Criterios de exclusión e inclusión..... 46

Generación de cadenas de búsqueda..... 47

Generación del instrumento para la extracción de datos..... 48

Fase de Ejecución 50

Ingreso de cadenas de búsqueda en motores 50

Cadena de IEEExplore 50

Cadena de Scopus 50

Cadena de Web of Science 50

Selección de estudios y extracción de información 51

Registrar los datos en el instrumento de selección 51

Identificación de las mejores prácticas 52

Identificación de las mejores prácticas de desarrollo 53

Fase de Análisis 54

Descripción del Instrumento *55*

Fase de Diseño **57**

Fase de Codificación **59**

Fase de Pruebas **60**

Descripción del instrumento *60*

Evolución de las mejores prácticas en el desarrollo de Sistemas *63*

Descripción de criterios del instrumento para las mejores practicas *63*

Capítulo IV: Análisis de Resultados **64**

Selección de mejores prácticas de las metodologías tradicionales 2018-2023 **64**

Instrumento de selección para desarrollo de software: Fase Análisis **64**

Descripción del Instrumento *64*

Instrumento de selección para el desarrollo de software: Fase Diseño **67**

Descripción del instrumento *67*

Instrumento de selección para el desarrollo de software: Fase Codificación. . **68**

Descripción del instrumento *69*

Instrumento de selección para el desarrollo de software: Fase Pruebas **70**

Demostración del instrumento *71*

Selección de mejores prácticas de metodologías no tradicionales 2018-2023. **72**

Instrumento de selección para el desarrollo de software: Fase Análisis **73**

Instrumento de selección para el desarrollo de software: Fase de Diseño. **75**

Descripción del instrumento *75*

Instrumento de selección para el desarrollo de software: Fase Codificación .. **77**

Descripción del instrumento *77*

Instrumento de selección para el desarrollo de software: Fase Pruebas **78**

Descripción del instrumento *78*

Semaforización de la frecuencia de uso de las mejores practicas **79**

Principales hallazgos de mejores prácticas para la construcción de software **80**

Mejores prácticas en la fase de codificación **81**

Mejores prácticas en la fase de pruebas **82**

Propuestas actuales de mejores prácticas para el desarrollo de software **83**

Validación del modelo de trabajo para ayudar al desarrollo de software. **86**

Validación del Sistema **86**

Capítulo V: Ejecución del modelo de trabajo basado en las mejores prácticas de las metodologías tradicionales y no tradicionales. **99**

Introducción del Capítulo **99**

Análisis del entorno de ACR DIGITAL S.A.S. **99**

Historia *99*

Flujo de Procesos Actuales *100*

Diseño del modelo de trabajo **100**

Especificación Requisitos **101**

Propósito *101*

Descripción *101*

Mejores practicas *101*

Análisis Requisitos **101**

Propósito *101*

Descripción *102*

Mejores practicas *102*

Diseño **102**

Propósito **102**

Descripción **102**

Mejores practicas **102**

Desarrollo de software **103**

Propósito ***103***

Descripción ***103***

Mejores practicas ***103***

Pruebas **103**

Propósito ***103***

Descripción ***104***

Mejores practicas ***104***

Resultados experimentales **104**

Especificación de requisitos **106**

Análisis de requisitos y Desarrollo de software **106**

Capítulo V: Conclusiones y Recomendaciones **117**

Conclusiones **117**

Recomendaciones **119**

Bibliografía **120**

Anexos **122**

ÍNDICE DE TABLAS

Tabla 1 *Modelos y metodologías de desarrollo de software*..... 30

Tabla 2 *Diferencia entre metodologías tradicionales y agiles*..... 31

Tabla 3 *Cadenas de Búsqueda* 47

Tabla 4 *Instrumento de extracción de datos de publicaciones científicas* 48

Tabla 5 *Instrumento de extracción en uso* 49

Tabla 6 *Instrumento de análisis de artículos científicos* 52

Tabla 7 *Identificación de las mejores prácticas.* 53

Tabla 8 *Identificación de las mejores prácticas: Fase Análisis.* 56

Tabla 9 *Identificación de las mejores prácticas: Fase Diseño* 58

Tabla 10 *Identificación de las mejores prácticas en la Fase de Codificación* 59

Tabla 11 *Identificación de las mejores prácticas en la Fase Pruebas.* 62

Tabla 12 *Instrumento de selección de mejores prácticas: Fase Análisis* 65

Tabla 13 *Mejores prácticas de la Fase de Análisis 2018 -2023* 66

Tabla 14 *Mejores prácticas de la Fase de Diseño 2018-2023*..... 68

Tabla 15 *Mejores prácticas de la Fase de Codificación 2022-2023*. 69

Tabla 16 *Mejores prácticas de la Fase de Pruebas 2018-2023* 71

Tabla 17 *Semaforización de las mejores prácticas Fase Análisis* 74

Tabla 18 *Evaluación de las mejores prácticas en la Fase de Análisis*..... 76

Tabla 19 *Semaforización de mejores prácticas Fase Codificación*..... 78

Tabla 20 *Indicadores visuales de las mejores prácticas en la Fase de Pruebas* 79

Tabla 21 *Propuestas de las mejores prácticas para el desarrollo de software* 84

Tabla 22 *Pregunta 1* 86

Tabla 23 *Pregunta 2* 87

Tabla 24 *Pregunta 3* 89

Tabla 25 *Pregunta 4* 90

Tabla 26 *Pregunta 5* 91

Tabla 27 *Pregunta 6* 92

Tabla 28 *Pregunta 7* 94

Tabla 29 *Pregunta 8* 95

Tabla 30 *Pregunta 9* 96

Tabla 31 *Pregunta 10* 97

Tabla 32 *Historia de Usuario número 1* 106

Tabla 33 *Historia de Usuario número 2* 107

Tabla 34 *Historia de Usuario número 3* 108

Tabla 35 *Historia de Usuario número 4* 109

Tabla 36 *Historia de Usuario número 5* 111

Tabla 37 *Historia de Usuario número 6* 111

Tabla 38 *Modelo de trabajo* 113

ÍNDICE DE FIGURAS

Figura 1 *Estructura de cifrado AES Proceso y ciclo de vida de software* 30

Figura 2 *Modelo en cascada* 38

Figura 3 *Modelo prototipo*..... 38

Figura 4 *Modelo para el desarrollo rápido de aplicaciones* 40

Figura 5 *Porcentaje de buenas prácticas* 66

Figura 6 *Resumen de buenas prácticas* 68

Figura 7 *Porcentaje de mejores prácticas establecido*..... 70

Figura 8 *Porcentaje de mejores prácticas establecido*..... 72

Figura 9 *Fase de Análisis: Porcentaje de aplicación de las mejores prácticas*..... 75

Figura 10 *Fase Diseño: Porcentaje de aplicación de mejores prácticas.* 77

Figura 11 *Fase de Codificación: Porcentaje de Implementación* 78

Figura 12 *Evaluación de la fase de pruebas: Porcentaje de implementación.* 80

Figura 13 *Resultados clave: Fase de Diseño* 81

Figura 14 *Principales descubrimientos: Fase de Codificación* 82

Figura 15 *Principales hallazgos de las mejores prácticas: Fase de Pruebas* 83

Figura 16 *Pregunta 1*..... 87

Figura 17 *Pregunta 2*..... 88

Figura 18 *Pregunta 3*..... 89

Figura 19 *Pregunta 4*..... 90

Figura 20 *Pregunta 5*..... 91

Figura 21 *Pregunta 6*..... 93

Figura 22 *Pregunta 7*..... 94

Figura 23 *Pregunta 8*..... 95

Figura 24 *Pregunta 9*..... 96

Figura 25 *Pregunta 10*..... 97

Figura 26 *Etapas en el desarrollo de software*..... 100

Figura 27 *Flujo de comunicación*..... 105

Figura 28 *Arquitectura de la aplicación*..... 105

Figura 29 *Pantalla de Login e Ingreso al sistema.* 107

Figura 30 *Pantalla de Login e Ingreso al sistema* 108

Figura 31 *Proceso de adicionar usuarios.* 109

Figura 32 *Proceso de ingreso de órdenes.* 110

Figura 33 *Pantalla menú* 110

Figura 34 *Proceso registro de órdenes.*..... 111

Figura 35 *Proceso despachos*..... 112

Figura 36 *Tablero de Trabajo* 114

Figura 37 *Trabajo Pendiente* 114

Figura 38 *Épicas* 115

Figura 39 *Casos de Uso*..... 116

Figura 40 *Historia de Usuario*..... 116

Resumen

El presente trabajo de tesis se centra en el campo de la Ingeniería de Software y tiene como objetivo diseñar un modelo de trabajo para la empresa ACR DIGITALD S.A.S. que combine metodologías tradicionales y no tradicionales, implementando las mejores prácticas de la industria de desarrollo de software. Para lograr este objetivo, se llevó a cabo una revisión bibliográfica exhaustiva que permitió identificar publicaciones científicas relevantes que orientaron el proceso de desarrollo basado en la aplicación de metodologías tanto tradicionales como no tradicionales. Asimismo, mediante esta revisión sistemática se identificó la aplicación habitual de modelos, técnicas y herramientas en el desarrollo de sistemas en empresas, lo cual proporcionó una base sólida para adoptar estas prácticas y construir un modelo de trabajo adecuado. Como resultado de este estudio, se seleccionaron una serie de artefactos, tales como casos de uso, diagramas de clases y se generaron plantillas específicas para respaldar la especificación de requisitos, el análisis de requisitos, la documentación de código, la programación orientada a objetos y los planes de pruebas funcionales y unitarias. Estos artefactos y plantillas fueron adoptados para respaldar cada fase del proceso de desarrollo. El enfoque de este trabajo se basa en la factibilidad de aplicar metodologías y mejores prácticas de desarrollo de software en la empresa ACR DIGITALD S.A.S. El análisis exhaustivo de la literatura y la selección cuidadosa de artefactos y plantillas permitieron diseñar un modelo de trabajo eficiente y adaptado a las necesidades específicas de la empresa. La implementación de este modelo no solo mejorará la calidad del software desarrollado, sino que también optimizará el proceso de desarrollo y aumentará la productividad de los equipos de trabajo.

Palabras Clave: Modelo de trabajo, metodologías tradicionales, metodologías no tradicionales, mejores prácticas, calidad del software.

Abstract

This thesis work focuses on the field of software engineering and aims to design a working model for the company ACR DIGITALD S.A.S. that combines traditional and non-traditional methodologies, implementing the best practices of the software development industry. To achieve this objective, an exhaustive bibliographic review was carried out to identify relevant scientific publications that guided the development process based on the application of both traditional and non-traditional methodologies. Likewise, through this systematic review, the usual application of models, techniques and tools in the development of systems in companies was identified, which provided a solid basis for adopting these practices and building an adequate working model. As a result of this study, a number of artifacts were selected, such as use cases, class diagrams, and specific templates were generated to support requirements specification, requirements analysis, code documentation, object-oriented programming, and functional and unit test plans. These artifacts and templates were adopted to support each phase of the development process. The focus of this work is based on the feasibility of applying software development methodologies and best practices in the company ACR DIGITALD S.A.S. The exhaustive literature analysis and the careful selection of artifacts and templates allowed designing an efficient working model adapted to the specific needs of the company. The implementation of this model will not only improve the quality of the developed software, but will also optimize the development process and increase the productivity of the work teams.

Key words: Work model, traditional methodologies, non-traditional methodologies, best practices, software quality

Capítulo I

Presentación del problema

Introducción

En este capítulo se abordan las generalidades, incluyendo una descripción de los antecedentes históricos que han influido en la evolución de las metodologías y modelos utilizados en el proceso de desarrollo de software en el campo de la ingeniería de software. Basándose en estos antecedentes, se establece el planteamiento del problema y se detallan los objetivos que se persiguen para lograr con éxito el proyecto de investigación.

Es necesario, por tanto, exponer la justificación e importancia del tema propuesto para la investigación. Adicionalmente, se definen las Hipótesis, que son enunciados no verificados que, una vez refutados o confirmados, dejarán de ser Hipótesis y se convertirán en enunciados verificados. En el caso del proyecto de investigación se tratará una Hipótesis de trabajo.

Antecedentes

Las metodologías de desarrollo de software han ido evolucionando desde los años 40, desde el principio de las computadoras, donde no se contaba con estándares, todo era artesanal, lo que terminaba en proyectos fallidos, estos proyectos no cumplían con las expectativas del usuario y terminaron en lo que se le conoce como la “crisis de software”.

Los proyectos que se realizaban eran de manera empírica y artesanal y las entregas de estos de manera extemporáneas y con los presupuestos excedidos, por lo que se optó por modelos y metodologías donde se incrementan estándares, formalidades y controles (Maida & Pacienzia, 2015). Uno de los proyectos fracasados en la presente fueron las muertes causadas por el Therac-25 en los años de 1985 al 1987, que es una máquina de radioterapia, donde varios pacientes murieron en hospitales tanto de Estados Unidos como de Canadá, debido a la alta radiación que se aplicó sin control, atribuidas en la calidad del software médico (Leveson & Turner, 1993). Por este motivo se incrementa los modelos y

metodologías de desarrollo de software que permitieron ir poco a poco mejorando, estas fueron incorporando estándares, formalidades y controles al desarrollo de software.

Las metodologías tradicionales se incorporaron con la finalidad de dar disciplina para el proceso del desarrollo de software y hacerlo de manera eficiente. Presentan el proceso en una dirección definida y se necesita realizar al inicio del proyecto la recolección de requisitos, análisis y diseño de manera eficaz, debido a que esto se realiza una sola vez y los requisitos no deben ser modificados (Riano, 2021).

La evolución siguió en marcha, surgiendo proyectos con requerimientos cambiantes y tiempos exigentes, donde las metodologías no se adaptaron, por lo que se desarrollan nuevas metodologías, donde la interacción del equipo y del usuario son primordiales, así también, las entregas tempranas y el enfoque a cambios. Las metodologías agiles o no tradicionales son adaptables y flexibles al proceso, los cambios son eventos esperados que se acogen con normalidad (Riano, 2021).

En la actualidad las empresas exigen software de calidad, por lo que se ha decidido profundizar las metodologías y modelos donde se mejora las características y detalles del software, con el fin de identificar técnicas y métodos para adoptarlas como mejores prácticas que crea un producto de calidad (Maida & Pacienzia, 2015).

La propuesta de este proyecto está basada en las metodologías correspondientes ya sea tradicionales y no tradicionales, así también, modelos para el desarrollo de software, que tiene como inicio el estudio de las metodologías que se ocupa para el desarrollo de software identificando las mejores prácticas que definen el desarrollo de software.

Planteamiento y Formulación del Problema

El principal desafío de la ingeniería de software es que se desarrollen los productos software con calidad. Así también, busca aplicar las mejores prácticas en la industria del desarrollo de software, de tal manera que muchas de las cosas realizadas en nuestra vida cotidiana tienden a estar relacionadas con este tema.

Las mejores prácticas lo encontramos en el software que lo desarrollamos y una de la principal característica es optimizar las estructuras técnicas, organizativas y de proceso. Por lo que se requiere procedimientos apropiados y específicos para su construcción. La dificultad de la ingeniería en software es tener en ella métodos y técnicas que se apliquen como mejores prácticas para satisfacer la demanda del desarrollo de software y la calidad con que se entrega su producto.

Cabe recalcar que por la exigencia en la calidad de producto que se exige, esta puede generar pérdidas económicas e incluso terminar en un producto ineficiente, así también, el descontento del cliente que lo requiere y su objetivo a ser alcanzado. La identificación de las mejores prácticas está destinada a proporcionar una optimización en el tiempo para que los ejecutivos ocupados en la gestión de operaciones superiores cumplan con los imperativos de la organización, sus metodologías y modelos a seguir.

En la industria de software muchos de los sistemas no contienen una guía especifica que trate acerca de las mejores prácticas en el proceso que se desarrolla el software, el cual, debe ajustarse a la necesidad del software y su estructura, debido a que lo realizan de manera empírica, lo que dificulta el desenvolvimiento de los proyectos de desarrollo de software y la calidad con la que se desarrolla, es decir, las metodologías tradicionales y las metodologías no tradicionales no son utilizadas al parecer de manera correcta en este desarrollo, por lo que se busca desarrollar un modelo de desarrollo de software mediante la implementación de las mejores prácticas para soportar el desarrollo de software en la academia y la industria del software.

En base a lo expuesto se desarrolla el siguiente problema:

¿Cómo ayudar al proceso de desarrollo de software mediante la implementación de las mejores prácticas de la industria de desarrollo de software?

Justificación e Importancia

En la actualidad para realizar un software primero debemos relacionar su desarrollo con las mejores prácticas, el cual, define modelos o diseños de referencia que proporcionan un plan o protocolo para optimizar las estructuras técnicas, organizativas y de proceso. Muchas de estas estructuras no solo debemos incluirlas en el desarrollo de software, sino también, en nuestra vida cotidiana.

Las mejores prácticas están tomando en la industria de desarrollo de software un gran impacto, debido a que optimizan su desarrollo, dándole así también un valor agregado tanto en innovación como en proceso.

Por este motivo esta investigación tiene como objetivo guiar a las personas que se encuentran en el mismo campo de estudio. Así también, para todo aquel que desea implementar en su desarrollo de software como son los docentes, alumnos, investigadores y hasta la misma industria del desarrollo de software.

Adicional a esto, el resultado de la investigación nos mostrará un modelo que servirá como guía en el desarrollo de software que comúnmente se realiza, debido a que este modelo incluye un sin número de buenas prácticas y así poder adquirir el conocimiento plasmado en el mismo.

Los beneficios que se encuentra en el desarrollo de esta investigación serían: Primero, podemos contar con las mejores prácticas de la industria de desarrollo de software que permitirán desarrollar software de manera eficaz. Como segundo punto sería un modelo que permitirá una adecuada utilización de las mismas, así también prevenir errores y problemáticas que surgen en el futuro. Como último punto podremos estandarizar un modelo que nos permitirá la construcción de software junto a las mejores prácticas de la industria de software.

Objetivos

Objetivo General

Elaborar un enfoque laboral que integre tanto metodologías convencionales como alternativas con el fin de respaldar el progreso del desarrollo de software, incorporando las prácticas más destacadas de la industria del desarrollo de software en ACR DIGITALS S.A.S.

Objetivos Específicos

- Construcción del marco teórico para fundamentar las metodologías y modelos tradicionales y no tradicionales de desarrollo de software.
- Diseñar un modelo de trabajo para ayudar al proceso de desarrollo de software mediante la implementación de las mejores prácticas.
- Implementar el modelo de trabajo que ayude y beneficie el desarrollo de software mediante la implementación de las mejores prácticas de la industria de desarrollo de software.
- Validar el modelo de trabajo para ayudar al proceso de desarrollo de software mediante la implementación de las mejores prácticas.

Hipótesis

Si se elabora un modelo de trabajo basado en las metodologías tradicionales y no tradicionales entonces se dispone de un modelo para soporte de desarrollo de software mediante la implementación de las mejores prácticas de la industria de desarrollo de software en la empresa ACR DIGITALS S.A.S.

Variables de la Investigación

Variable Dependiente

Se mejora el proceso de desarrollo de software mediante la implementación de las mejores prácticas de la industria de desarrollo de software.

Variable Independiente

Modelo de trabajo basado en las metodologías tradicionales y no tradicionales.

Conceptualización de la Variable Independiente. El modelo de trabajo basado en las metodologías tradicionales y no tradicionales es una margen de trabajo que obtendrá un conjunto de buenas prácticas, modelos, herramientas, métodos y técnicas que permiten el desarrollo y la implementación mediante una ayuda al proceso de desarrollo de software

Indicadores

- Tiempo del desarrollo de software.
- Ayuda sobre el proceso de desarrollo de software.
- Modelo de trabajo para desarrollar software.
- Reutilización de recursos.
- Evaluación de desarrolladores.
- Generar documentación.

Capítulo II

Marco teórico

Introducción

La siguiente parte comienza con la configuración del Marco Teórico, delineando eventos históricos que permiten visualizar la evolución de las metodologías y modelos, tanto tradicionales como no convencionales, en el ámbito del desarrollo de software dentro de la ingeniería en software.

El análisis histórico se lo efectuará de manera cronológica, además de describir los antecedentes conceptuales, conceptualizando las metodologías y modelos del proceso de desarrollo de software y como última etapa, tenemos los antecedentes contextuales en el cual nos permite justificar la procedencia del problema que se plantea en la investigación.

Antecedentes Históricos

En los siguientes pasos, describiremos la evolución de las metodologías y modelos, tanto convencionales como alternativos, basándonos en las prácticas más destacadas de la industria del desarrollo de software.

Primera etapa cronológica (1940-1970)

A partir de la década de los 40 nos encontramos en el inicio de la generación de las computadoras, junto a esto podemos ver la creación de programas y sistemas que necesitaban para su funcionamiento, estas prácticas no obedecen una metodología lo realizaban de manera empírica (Rivas, Corona, Gutiérrez, & Hernández, 2015).

Desde finales de la década de 1950, el desarrollo de software comenzó a llevarse a cabo de manera artesanal, lo que resultó en la aparición de una serie de problemas. Para esto se inicia una utilización de hardware y la complejidad de sus tareas (Garcés & Egas, 2013).

En la década de 1960, surge el modelo "Code and Fix" (codificar y corregir), que refleja la anarquía de los primeros años y marca el comienzo de la búsqueda de una base más sólida para la fabricación de software. Hacia la mitad de este período, se inicia la llamada "Crisis del Software", marcada por problemas recurrentes como costos elevados, falta de confiabilidad, entregas tardías e insatisfacción, que son característicos de los "síntomas de la crisis del software". Los proyectos más comunes que fracasaban dando perdidas hasta millonarias eran los sistemas de aeropuertos, sistemas de medicina y entre otros (Cadavid, Martínez, & Vélez, 2013).

En 1968, en Alemania, se acuña el término "Ingeniería de Software", definido como un "enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software" (Cadavid, Martínez, & Vélez, 2013) como respuesta a la “Crisis del Software”.

Segunda etapa cronológica (1970-2000)

A partir de los años 70 comienzan a resolver los problemas complejos, lo que permite crear etapas para el desarrollo de software, en 1972 Edsger Dijkstra, en su trabajo “The Humble Programmer” sienta las primeras bases para la creación de metodologías tradicionales que son utilizadas hasta la actualidad.

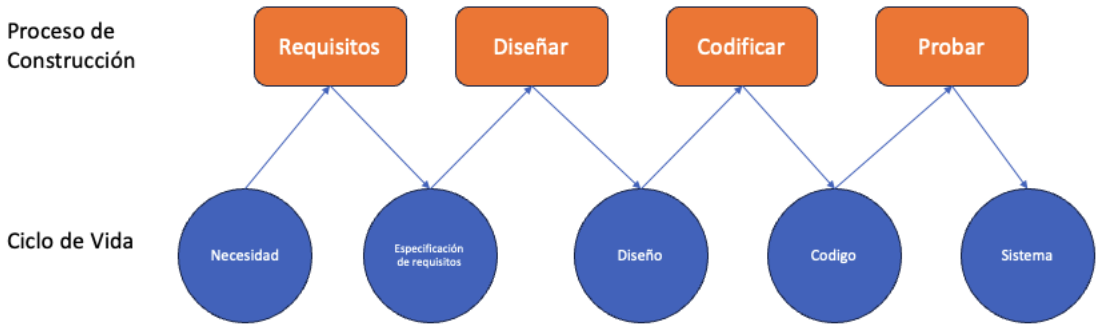
Luego, se introduce la famosa declaración "Go to statement considered harmful" junto con el libro "Discipline of Programming". Estas obras establecieron parámetros fundamentales que continúan siendo referentes en la actualidad para el desarrollo de software (Tinoco Gómez, Rosales López, & Salas Bacalla, 2014).

En la segunda mitad de la década de 1970, surge el término "Ciclo de Vida del Software" (SDLC, por sus siglas en inglés, Software Development Life Cycle). Este concepto se adoptó de manera consensuada para la construcción centralizada de software, definiendo estados a través de los cuales el producto software evoluciona desde su

concepción a partir de una necesidad hasta su conclusión (Garcés & Egas, 2013). En la figura 1 se muestra el proceso y ciclo de vida de software.

Figura 1

Estructura de cifrado AES Proceso y ciclo de vida de software



Nota. Tomado: Garcés & Egas (2013)

Entre 1970 y 1988 aparecen los “modelos tradicionales de desarrollo de software” y la diferencia entre modelos y metodologías (Cendejas Valdéz, Vega Lebrún, Careta Isordia, Gutiérrez Sánchez, & Ferreira Medina, 2014).

Modelo de desarrollo de software

Es la representación simplificada del proceso de desarrollo de software.

Metodología de desarrollo de software:

Se refiere a un enfoque estructurado que engloba modelos para la creación de software. En la Tabla 1, se destacan las principales metodologías y modelos tradicionales de desarrollo de software desde 1970 hasta 1990.

Tabla 1

Modelos y metodologías de desarrollo de software

Modelos de DS	Metodologías Tradicionales de DS
• **Modelo de Cascada** • **Modelo de Cascada en “V”** • Modelo de Desarrollo Evolutivo (Espiral)	• RUP (Rational Unified Process) • RAD (Rapis Application Development)

Modelos de DS	Metodologías Tradicionales de DS
• Modelo de Desarrollo Evolutivo por Prototipos • Desarrollo Evolutivo por etapas o Incremental • Desarrollo Evolutivo Iterativo • Modelo Basado en Componentes	• MSF (Microsoft Solution Framework) • Win- Win Spiral Model • Iconix • Desarrollo de sistemas de Jackson (JSD) • Ingeniería de la información • Structured System Analysis and Desing Method (SSADM)

Nota. Metodologías y modelos de desarrollo de software. Tomado: (Zumba Gamboa & León Arreaga, 2018).

Desde la década de 1990, la proliferación del internet y el surgimiento de una cantidad innumerable de requisitos han generado la necesidad de acelerar el desarrollo de software. Por lo que, a pesar de haber implementado Metodologías clásicas, en los proyectos de gran magnitud y con requisitos cambiantes se vieron fracasados, debido a que el tiempo se pasaba más en diseño. Los cambios en muchas especificaciones no eran de modalidad cambiantes y no eran compatibles para realizar una documentación adecuada y realizar el desarrollo de software ineficiente (Pérez, 2011).

Como respuesta a este desafío, surgen las metodologías ágiles, las cuales se implementan en el desarrollo de software con un enfoque más centrado en la creación del software que en la arquitectura o documentación. Estas metodologías acogen de manera positiva los cambios en los requisitos y promueven entregas tempranas durante el proceso de desarrollo. (Zumba Gamboa & León Arreaga, 2018).

Según Zumba y León, en la Tabla 2 se establece una distinción entre metodologías tradicionales y metodologías ágiles.

Tabla 2

Diferencia entre metodologías tradicionales y agiles

Metodologías ágiles	Metodologías Tradicionales
Se basan en heurísticas provenientes de prácticas de producción de código Énfasis en los aspectos humanos, el individuo y el trabajo en equipo	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo Énfasis en la definición del proceso: roles, actividades y artefactos

Metodologías ágiles	Metodologías Tradicionales
Preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente por el equipo	Impuestas externamente
Proceso menos controlado con pocos principios	Proceso muy controlado, numerosas normas
Contrato flexible e incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	Cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10) con pocos roles, más genéricos y flexibles	Grupos grandes con más roles y más específicos
Orientada a proyectos pequeños, y en el mismo lugar	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente
Pocos artefactos, el modelo es prescindible, modelos desechables	efectivas usadas en proyectos grandes
Menor énfasis en la arquitectura del software se va definiendo y mejorando a lo largo del proyecto	Mas artefactos, el modelo es esencial, mantenimiento en modelos
	La arquitectura del software es esencial, se defina tempranamente en el proyecto

Nota. Metodologías tradicionales y ágiles. Tomado: (Zumba Gamboa & León Arreaga, 2018).

Tercera etapa cronológica (2000-Presente)

Con la creciente demanda del software, lleva al crecimiento a métodos más simples y rápidos. A partir del año 2001 miembros de la comunidad de desarrollo de software en Utah, USA las proponen por 17 representantes de la industria del software como metodologías ágiles y así promueven el desarrollo ágil de aplicaciones (Amaro Calderón & Valverde Rebaza, 2007).

El Manifiesto Ágil marcó el inicio, siendo un documento que resume la filosofía "ágil", enfocándose en los siguientes valores:

- Priorizar al individuo y las interacciones dentro del equipo de desarrollo sobre el proceso y las herramientas.
- Valorar el desarrollo de software funcional por encima de la obtención de una extensa documentación.
- Favorecer la colaboración con el cliente en lugar de centrarse en la negociación de un contrato.

- Estar dispuesto a adaptarse a los cambios en lugar de seguir rigurosamente un plan preestablecido.
- (Cadavid, Martínez, & Vélez, 2013).

Encontramos métodos similares al ágil,

Entre las metodologías ágiles más destacadas se encuentran: Scrum, Crystal Clear (cristal transparente), programación extrema (en inglés eXtreme Programming o XP), desarrollo de software adaptativo (ASD), feature driven development (FDD), Método de desarrollo de sistemas dinámicos (Garcés & Egas, 2013).

Existen dos de mayor presencia internacional, mayor divulgación en textos y en la web, certificaciones, training, comunidades activas y participación en eventos:

- XP (Extreme Programming), tienen un enfoque más simple y rápido (Letelier & Penadés, 2006).
- Los Equipos Scrum son autogestionados, multifuncionales y trabajan en iteraciones.

Extreme Programming (XP), propuesto por Kent Beck, tiene como objetivo guiar a equipos de desarrollo de software pequeños, entre dos y diez desarrolladores, en entornos con requisitos imprecisos o cambiantes. Esta metodología ágil se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores y fomenta un buen clima laboral. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, así como en la comunicación fluida entre todos los participantes. (Canós , Letelier, & Penadés, 2003).

SCRUM, desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, establece un marco para la gestión de proyectos que ha demostrado ser exitoso en la última década. Esta metodología es especialmente adecuada para proyectos con cambios rápidos de requisitos. Sus principales características pueden resumirse en dos puntos clave. Primero, el desarrollo de software se realiza mediante iteraciones, conocidas como sprints, con una duración de 30 días, donde el resultado de cada sprint es un incremento ejecutable que se

presenta al cliente. Segundo, se enfatizan las reuniones a lo largo del proyecto, destacando la reunión diaria de 15 minutos del equipo de desarrollo para la coordinación e integración (Mariño & Alfonzo, 2014).

Las metodologías ágiles ofrecen una perspectiva diferente sobre cómo desarrollar sistemas, proponiendo un enfoque más rápido, adaptable y sin comprometer la rigurosidad impuesta por las metodologías clásicas.

Antecedentes conceptuales y referenciales

Software. Pressman menciona en su libro que son programas de computadora que al ejecutarse cumplen una función o un requerimiento deseado por un usuario. Son estructuras de datos que permite a los programas manipular información (Pressman, 2010).

El software se refiere a un programa o conjunto de programas que contienen las instrucciones con las cuales opera una computadora. Constituye el conjunto de directrices que las computadoras utilizan para manipular datos. Sin el software, la computadora sería simplemente un conjunto de recursos sin utilizar. Al cargar programas en una computadora, esta adquiere conocimiento instantáneo, actuando como si hubiera recibido una educación repentina; de repente, "sabe" cómo pensar y operar (Villacis, 2013).

Como se puede inferir a partir de la información recopilada, el software consiste en un programa o conjunto de programas que capacita a una computadora para ejecutar instrucciones. De este modo, posibilita la manipulación de información esencial para el usuario.

Ingeniería de Software. La ingeniería de software es una disciplina para poder llegar a el desarrollo de sistemas software, no existen limitaciones físicas para esto debido a que no está gobernado por leyes físicas o procesos de manufactura (Sommerville, 2011).

Según Pressman, la ingeniería de software se desarrolla mediante un proceso que proporciona enfoques robustos con el objetivo de aumentar las probabilidades de cumplir con los objetivos de negocio en términos de tiempo, calidad y funcionalidad. Esto es crucial

para las empresas que actualmente se enfrentan al desafío de llevar a cabo sus actividades de manera productiva, garantizando calidad y una producción estratégica. Para esto el uso de un enfoque adecuado para el desarrollo de software como es la obtención de requisitos, estimación, desarrollo y control son los pasos esenciales para una empresa (Pressman, 2010).

A partir de estos conceptos, se puede concluir que la ingeniería de software es una disciplina que facilita el desarrollo del software al abordar la obtención de requisitos, estimación, desarrollo y control. Su objetivo es llevar a cabo actividades de manera productiva, garantizando calidad y una producción estratégica.

Desarrollo de software. El desarrollo de software es una estructura utilizada para el desarrollo de un producto de software, entre sus sinónimos encontramos el “ciclo de vida del software”. Se encuentra diferentes modelos para el desarrollo de software y cada uno de estos contiene diferentes enfoques, con una variedad de tareas y actividades a ser ejecutadas (Gómez, López & Bacalla, 2010).

Proceso de desarrollo de software. Encontramos diferentes pasos para la ejecución del proceso

- Investigar requisitos del usuario: Se lleva a cabo en la fase del análisis. Consiste en ser un especialista al dominio del usuario para guiarlo en la especificación de sus requisitos. Para esto el desarrollador debe:
    - Escuchar claramente y descubrir el máximo de la información.
    - Interrogar y aclarar dicha información
    - Comprobar información y sugerir soluciones
    - Escribir el documento de especificación de requerimientos
- Definir claramente las características necesarias para el sistema: Se lleva a cabo en la fase de especificación. El objetivo es realizar una especificación de

requisitos que, en forma clara, comunique al proyectista las características del proyecto software.

- Crear o adaptar una solución adecuada al problema: Se lleva a cabo en la creación del proyecto. Se busca desarrollar una solución a los requisitos previamente recopilados, generando varias soluciones posibles. El resultado es un documento del proyecto que muestra claramente el correcto funcionamiento del proyecto para los que lo van a implementar.
- Desarrollar la solución propuesta: Esta etapa se ejecuta durante la fase de implementación. Aquí, se realiza el desarrollo de la aplicación, se escribe el código, se documenta y se abordan posibles errores. El código se prepara para la prueba, y se envían informes al proyectista, analista y probador o integrador.
- Garantizar que la solución responda al problema propuesto y funcione correctamente en el contexto previsto: Esta tarea se lleva a cabo en la fase de pruebas. Se verifica si la implementación es precisa y cumple con los requisitos establecidos.
- Modificar las soluciones cuando se presentan nuevos requisitos: Esta actividad se realiza en la fase de mantenimiento. Dado que las necesidades del usuario evolucionan con el tiempo, los cambios en cada requisito generan nuevas implementaciones y pruebas adicionales, lo que puede implicar trabajo adicional en el proyecto o incluso análisis adicional (Martínez, 2015).

Modelos de desarrollo de software. Los modelos han sido creados con el fin de organizar el proceso de desarrollo de software, es una representación abstracta de este proceso.

Un modelo de desarrollo de software determina el orden en el que se llevan a cabo las actividades del proceso de desarrollo de software (Gómez & Fuentes, 2012).

A continuación, se proporcionará los principales modelos utilizados:

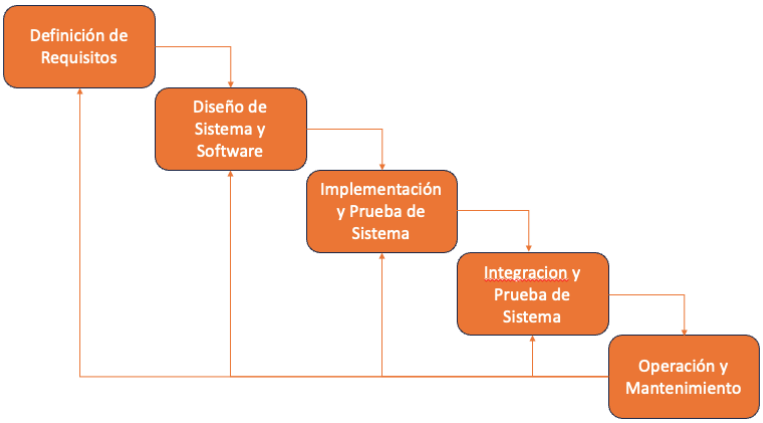
Modelo en Cascada. Promovido por Winston Royce en 1970, este enfoque sugiere una metodología sistemática y secuencial, disciplinada y basada en análisis, diseño, pruebas y mantenimiento. tal como se muestra en la Figura 2.

Las principales fases de este enfoque son:

- Definición de los requisitos: Se establecen los servicios, restricciones y objetivos en detalle con la participación de los usuarios del sistema.
- Diseño de software: Se divide en sistemas de software o hardware y se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.
- Implementación y pruebas unitarias: Implica la construcción de los módulos y unidades de software, seguido de pruebas individuales para cada unidad.
- Integración y pruebas del sistema: Todas las unidades se integran y se realizan pruebas conjuntas. El conjunto probado se entrega al cliente.
- Operación y mantenimiento: Esta fase es generalmente la más extensa. El sistema se pone en marcha, se corrigen errores descubiertos, se realizan mejoras de implementación y se identifican nuevos requisitos.

Figura 2

Modelo en cascada

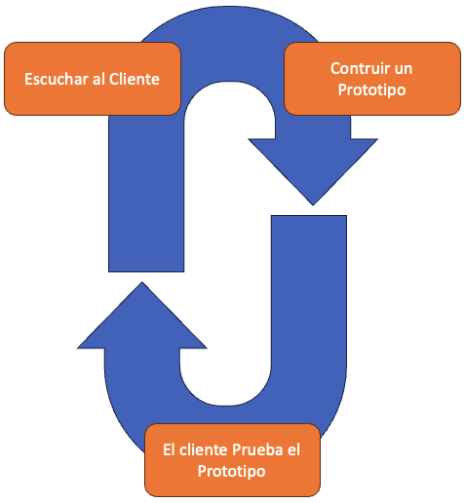


Nota. Modelo en cascada. Tomado: (Delgado Olivera & Díaz Alonso, 2021)

Prototipo. Este enfoque opera con el propósito de comprender los requisitos del usuario y trabajar en su mejora continua. Se comienzan identificando los requisitos conocidos, y a partir de estos, se desarrolla rápidamente un prototipo o maqueta. Este prototipo es luego evaluado por el cliente, quien lo utiliza y brinda retroalimentación. Este proceso contribuye a refinar nuevamente los requisitos del software a desarrollar, como se ilustra en la Figura 3.

Figura 3

Modelo prototipo



Nota. Modelo prototipo. Fuente: (Delgado Olivera & Díaz Alonso, 2021)

Reutilización o Desarrollo basado en componentes. Los compromisos en los requisitos son inevitables, lo que puede resultar en que el software no satisfaga completamente las expectativas del cliente. Además, estos compromisos pueden generar altos costos, especialmente si la adaptación de nuevos paquetes es costosa. Además, pueden surgir dificultades para asegurar el versionamiento si los proveedores de componentes no mejoran constantemente sus productos. Este panorama subraya la importancia de gestionar cuidadosamente los compromisos y considerar las implicaciones a largo plazo en el desarrollo de software. (Ojeda & Fuentes, 2012).

Desarrollo en espiral. Lo que describes es el Modelo Espiral, propuesto por Barry Boehm en 1986. Este modelo es actualmente uno de los más reconocidos y se diferencia al representar el ciclo de desarrollo como una espiral en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra. En este enfoque, las actividades de especificación, desarrollo y validación se entrelazan. El proceso comienza con un sistema inicial que se desarrolla rápidamente a partir de especificaciones abstractas, siguiendo una trayectoria en espiral para refinar continuamente el producto a lo largo del tiempo.

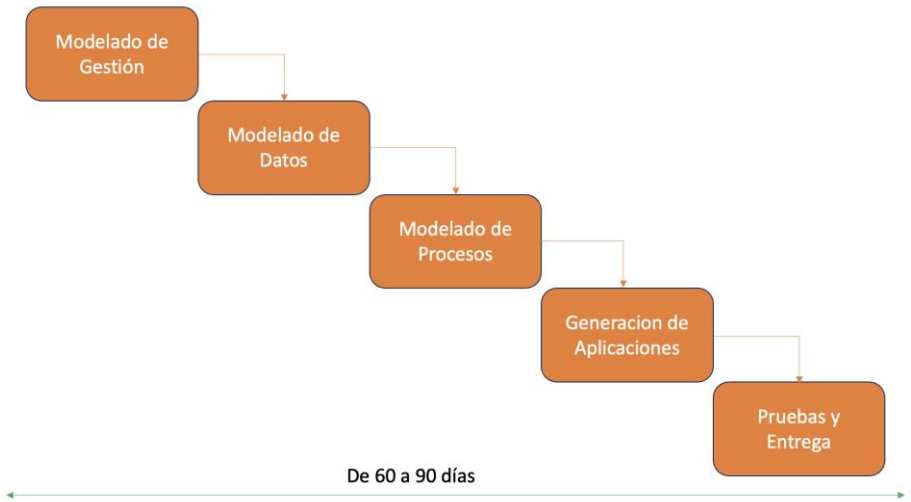
Modelo RAD. El Desarrollo Rápido de Aplicaciones (RAD, por sus siglas en inglés) es un modelo de proceso de desarrollo de software de duración relativamente corta, entre 60 y 90 días. Este modelo comprende las siguientes etapas, como se muestra en la Figura 4:

- Modelado de gestión: Identifica la dirección de la información y quién la procesa.
- Modelado de datos: Define los almacenes de datos y cómo se relacionan entre sí.
- Modelado del proceso: Utilizado para realizar operaciones en un objeto de datos, como añadir, modificar, suprimir o recuperar.
- Generación de aplicaciones: Utiliza herramientas de cuarta generación para crear el software y facilitar la construcción del programa.

- Pruebas y entrega: Se llevan a cabo pruebas de calidad del software diseñado con la herramienta RAD, seguidas por la implementación de la aplicación.

Figura 4

Modelo para el desarrollo rápido de aplicaciones



Nota. Modelo para el desarrollo rápido de aplicaciones. Tomado: (Delgado Olivera & Díaz Alonso, 2021).

Lenguaje Unificado de Modelado UML. UML (Unified Modeling Language) es un conjunto de diagramas que representan un software desde distintos puntos de vista, se ha transformado en un estándar de facto para lo que a representaciones de software orientado a objetos se refiere.

Provee elementos para representar visualmente los sistemas de software, pero no provee ninguna metodología o proceso para guiar el desarrollo de los mismos (Bustos, 2002).

Metodología. Sebas Zerón en su artículo científico define la metodología como El estudio del método o los métodos y abarca el análisis de sus características, cualidades y debilidades (Zerón Acastenco, 2014).

Lo que podemos definir como un conjunto de procedimientos racionales que proviene de una investigación científica que requiere de habilidades para abarcar análisis características y debilidades.

Metodologías de desarrollo de software. Correcto, una metodología de desarrollo de software se define como un marco de trabajo utilizado para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. A lo largo del tiempo, ha evolucionado una amplia variedad de estos marcos de trabajo, cada uno con sus propias fortalezas y debilidades. Es importante destacar que una metodología de desarrollo de sistemas no tiene la obligación de ser adecuada para todos los proyectos; su elección depende del contexto y las características específicas de cada proyecto (Maida & Pacienzia, 2015).

Lo que describes es la definición general de una metodología de desarrollo de software. Es un conjunto de procedimientos, técnicas, herramientas y documentación de soporte que asiste a los desarrolladores en la creación de nuevo software. Por lo general, se compone de fases o etapas desglosadas en subfases, módulos, etapas, pasos, etc. Esta descomposición ayuda a los desarrolladores a seleccionar las técnicas apropiadas en cada estado del proyecto, facilitando así la planificación, gestión, control y evaluación del desarrollo de software (Cataldi, 2000).

Podemos concluir mediante los conceptos obtenidos que las metodologías de desarrollo de software son un marco de trabajo que nos permite estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Contiene fases que se descompondrán en subfases, módulos, etapas, entre otros.

Encontramos una clasificación entre metodologías, son las metodologías tradicionales y no tradicionales. Las metodologías tradicionales son metodologías que

buscan desarrollar un software mediante el orden y la documentación, a diferencia de las metodologías no tradicionales buscan desarrollar un software utilizando la comunicación directa entre las personas involucradas en el proceso de desarrollo.

Metodologías Tradicionales. Las metodologías que describiste, a veces llamadas de forma despectiva como "metodologías pesadas", se caracterizan por centrarse en la elaboración exhaustiva de documentación para todo el proyecto, la planificación y control detallados, especificaciones precisas de requisitos y modelado. Estas metodologías tradicionales establecen un énfasis particular en la fase inicial del desarrollo del proyecto, imponiendo una disciplina rigurosa sobre el proceso de desarrollo de software. El objetivo es lograr un software más eficiente mediante un enfoque estructurado y disciplinado (Letelier & Penadés, 2006).

Las metodologías tradicionales buscan introducir disciplina en el proceso de desarrollo de software para hacerlo predecible y eficiente. Para lograrlo, se apoyan en un proceso detallado con un enfoque en la planificación, similar al utilizado en otras disciplinas de ingeniería. Sin embargo, la principal problemática de esta filosofía es que la cantidad de actividades que deben llevarse a cabo para seguir la metodología puede generar retrasos en la etapa de desarrollo. Además, estas metodologías no suelen adaptarse fácilmente a los cambios, lo que puede resultar limitante en entornos donde la flexibilidad y la capacidad de respuesta a modificaciones son esenciales. (García Rodríguez, 2015).

Las metodologías tradicionales contienen una guía de trabajo que nos permiten imponer disciplina para el proceso de desarrollo de software, contienen una documentación exhaustiva del proyecto detallando la planificación propia. Imponen un riguroso trabajo con el fin de conseguir un software de mejor calidad.

Metodologías no tradicionales. Las metodologías no convencionales son caracterizadas por su capacidad adaptativa, lo cual es crucial ya que difiere de la predictibilidad que buscan las metodologías tradicionales. En el contexto de las metodologías ágiles, los cambios son considerados eventos esperados que aportan valor al cliente. Este enfoque subraya la flexibilidad y la disposición a ajustarse a las necesidades cambiantes, ofreciendo una perspectiva dinámica y centrada en proporcionar beneficios al cliente. (Figueroa, Solís, & Cabrera, 2008).

El propósito de esta metodología fue establecer los valores y principios que posibilitarían a los equipos desarrollar software de manera rápida y adaptarse a los cambios que puedan surgir durante el proyecto. La intención era proporcionar una alternativa a los procesos de desarrollo de software tradicionales, conocidos por su rigidez y por estar dirigidos por la documentación generada en cada una de las actividades realizadas. (Montero, Cevallos, & Cuesta , 2018).

Las metodologías ágiles o no tradicionales nos permiten desarrollar software de manera rápida, respondiendo a los cambios que el usuario nos dé, debido a que sus requisitos son cambiantes. Estos cambios son lo que generan valor para el cliente.

Capítulo III

Análisis, diseño y desarrollo del sistema web

Introducción del Capítulo

La relevancia del diseño del modelo de trabajo en el desarrollo de software radica en su contribución significativa al proceso de construcción de sistemas. Al emplear las mejores prácticas de metodologías, ya sean tradicionales o no tradicionales, o al proponer un nuevo modelo basado en esas prácticas, el objetivo es mejorar diversos aspectos del proceso de desarrollo de software. Esto incluye la definición de requisitos, la determinación de entregables, la documentación, las pruebas, entre otros. En la empresa de desarrollo de software ACR-DIGITALS S.A.S. en el proyecto “El sabor de la Tía”.

En este capítulo, se desarrollará un modelo de trabajo que aplicará las mejores prácticas de desarrollo de Ingeniería de Software. Para iniciar, se realizará una revisión exhaustiva de la literatura académica relacionada con el proyecto de estudio, centrándose especialmente en las metodologías, tanto tradicionales como no tradicionales. Además, se llevará a cabo un segundo estudio de referencia con el objetivo de identificar las mejores prácticas seleccionadas para adoptarlas y utilizarlas como fundamentos en el diseño del modelo de trabajo propuesto. Una vez establecido el modelo de trabajo para el desarrollo de software, se avanzará con la implementación en el proyecto propuesto. A continuación, en las siguientes secciones se detalla el proceso de la revisión sistemática. Dando inicio con la extracción de publicaciones científicas, etapa que inicia con la planeación del contenido a extraer, la creación de una estrategia y la creación o generación de artefactos para documentar, continua con la ejecución, donde se ingresan las cadenas de búsqueda creadas y la identificación de artículos relevantes al caso para pasas a la fase de análisis y diseño terminando en la fase de codificación y validando con la etapa de pruebas.

Extracción de artículos científicas

En esta etapa para el desarrollo del proyecto propuesto, se dará uso al proceso metodológico de revisión científico-bibliográfica, el mismo consta de cuatro fases: Planeación, Selección, Extracción y Ejecución, enfocando el proceso en las búsquedas de las mejores prácticas de desarrollo de software. A continuación, se detalla cada una de las etapas:

Fase de Planeación

Durante esta fase, se delimita el ámbito de estudio, permitiendo así la definición de una estrategia efectiva para la búsqueda de contenidos. Se describe a continuación el contexto de la investigación y la estrategia de búsqueda bibliográfica.

Descripción del contexto de la investigación

El propósito de este estudio es llevar a cabo una revisión completa de la literatura sobre las mejores prácticas de las metodologías tradicionales y no tradicionales aplicadas en el ámbito del software o sistemas en distintos sectores, incluyendo la industria, la investigación, la agricultura, la educación, entre otros. Por lo tanto, el primer paso para la realización de la investigación consiste en la elaboración de una estrategia, la cual se detalla a continuación.

Estrategia de búsqueda bibliográfica

En esta fase, se concentra en la búsqueda de artículos y la revisión de artículos seminales para la extracción de palabras clave. Esto implica identificar los motores de búsqueda y establecer criterios de inclusión y exclusión. La extracción bibliográfica se llevará a cabo mediante la búsqueda de publicaciones indexadas en bases de datos científicas, utilizando IEEExplore, Scopus y Web of Science. Estos resultados servirán como guía para identificar las mejores prácticas en la industria del desarrollo de software. Con las

bases de datos seleccionadas y los artículos semilla, se procederá a la selección de palabras clave y descripciones, utilizando las primeras para la búsqueda literaria.

A continuación, se expondrán los criterios de inclusión y exclusión, que se tomaron en cuenta para la búsqueda filtrada.

Criterios de exclusión e inclusión

Para identificar los artículos obtenidos en las diversas bases de datos científicas, se aplicaron los siguientes criterios de inclusión y exclusión.

Se seleccionarán los artículos científicos que guarden relación con las mejores prácticas, el desarrollo de software, así como las metodologías ágiles y tradicionales. Para lo cual:

- Se incluirán estudios:
    - En los idiomas inglés.
    - Publicaciones científicas indexadas en IEEExplore, Scopus y Web Of Sciece
    - Artículos científicos
    - Tesis
    - Casos de estudio.
- Se excluirán estudios:
    - Que no estén en el idioma inglés.
    - Publicaciones científicas que no estén bien definidos en contenido y estructura.

Una vez se tomaron en cuenta las validaciones para la extracción de información el siguiente paso es la generación de cadenas de búsqueda.

Generación de cadenas de búsqueda

La generación de cadenas de búsqueda tiene como objetivo facilitar y presentar de manera eficiente información relevante sobre el tema de estudio. Con este propósito, se ha desarrollado una tabla que posibilita la identificación de palabras clave junto con sus sinónimos, como se muestra en la Tabla 3, con el fin de construir la cadena de búsqueda:

Tabla 3

Cadenas de Búsqueda

Palabras claves	Sinónimos
Programas	Software
Metodología	Procedimientos
Tradicional o No Tradicional	Tradicional o Ágil
Buenas	Mejores
Prácticas	Métodos Técnicas

Nota: En esta tabla se presenta de manera detallada cada palabra clave junto con sus respectivos sinónimos, los cuales fueron empleados en la construcción de la cadena de búsqueda.

A continuación, se procede a consolidar los resultados mediante la utilización de los operadores lógicos "and" y "or" y la traducción al idioma inglés de cada uno. Se establece que para la concatenación de términos se empleará el salto de línea con el operador "and", mientras que para la inclusión de sinónimos se utilizará el operador "or". El resultado de este proceso nos proporciona la siguiente cadena de búsqueda:

“(software) AND (Methodologies OR Procedures) AND (traditional OR Agil) AND (Best) AND (Practice OR Methods OR Techniques)”

La cadena de búsqueda desarrollada presentó un total 568 publicaciones en el rango comprendido entre los años 2018 hasta el 2023, Ajustando la misma de acuerdo con las necesidades del proyecto, obteniendo finalmente 68 artículos.

Una vez establecida la estrategia de búsqueda bibliográfica, bases de datos científicas seleccionadas, identificadas las palabras clave y las cadenas de búsqueda se procederá a crear un instrumento de extracción de información relevante de publicaciones científicas.

Generación del instrumento para la extracción de datos

Una vez definidas las publicaciones filtradas, se procede a generar el instrumento de extracción de datos de publicaciones científicas, mediante una tabla informática que cuente con la información necesaria para la evaluación de las mejores prácticas, tal como se puede observar en la Tabla 4. Se procede a describir cada uno de los campos que forman parte del instrumento de extracción de información.

Tabla 4

Instrumento de extracción de datos de publicaciones científicas

ID	Autores	Año	Fuente	Editorial	Tipo de publicación	Titulo

Nota: En la tabla se observa el diseño del instrumento de recolección de información

- ID: representa al identificador de cada publicación extraída en la base de datos.
- Autores: se refiere a los autores que realizaron la publicación.
- Año: muestra el año en el que se realizó la publicación.
- Fuente: muestra donde fue presentada la publicación.
- Editorial: Indica donde se expuso el articulo científico.
- Tipo de publicación: describe el tipo de publicación, puede ser, tesis, caso de estudio o articulo científico.
- Título: indica el nombre de la publicación seleccionada.

Con el instrumento correctamente generado se procederá a realizar la ejecución planificada a fin de encontrar las mejores prácticas en el desarrollo de software.

Con el instrumento diseñado se procederá a ingresar la información recopilada como se observa en la Tabla 5.

Tabla 5

Instrumento de extracción en uso

ID	Autores	Año	Fuente	Editorial	Tipo de publicación	Titulo
P01	Leo R. Vijayasarathy; Charles W. Butler	2018	IEEE Software	Scopus	Articulo	Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?
P02	Abhiup Sinha; Pallabi Das	2021	2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)	Scopus	Articulo	Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry
P03	An experience in blending the traditional and Agile methodologies to assist in a small software development project	2018	2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)	Scopus	Articulo	An experience in blending the traditional and Agile methodologies to assist in a small software development project

Nota: En la tabla se observa el diseño del instrumento de recolección de información para el análisis de artículos científicos. (Ver instrumento completo: https://docs.google.com/spreadsheets/d/1TzHtc8oZP8HKe9RQtcAWn3jyUgRNEwWlqGRnJ2KlvIA/edit?usp=sharing).

Fase de Ejecución

En la etapa de ejecución se procederá a ingresar las cadenas de búsqueda, a fin de extraer las mejores prácticas del proceso de desarrollo de software, para dicho propósito se tomaron en cuenta los parámetros anteriormente establecidos, siendo el primer paso el ingreso de las cadenas de búsqueda en los diferentes motores de bibliografía científica.

Ingreso de cadenas de búsqueda en motores

Se ingresan las respectivas cadenas de búsqueda en IEEExplore, Scopus, Web Of Sciece, presentando las mismas a continuación.

Cadena de IEEExplore

El rango de búsqueda para los filtros fueron desde el año 2018 al 2023, obteniendo como resultados 140 publicaciones tanto de conferencias, cursos, revistes y libros.

(software) AND (Methodologies OR Procedures) AND (traditional OR Agil) AND (Best) AND (Practice OR Methods OR Techniques).

Cadena de Scopus

El rango de búsqueda para los filtros fueron desde el año 2018 al 2023, obteniendo como resultados 250 publicaciones tanto de conferencias, cursos, revistes y libros.

(software) AND (Methodologies OR Procedures) AND (traditional OR Agil) AND (Best) AND (Practice OR Methods OR Techniques).

Cadena de Web of Science

El rango de búsqueda para los filtros fueron desde el año 2018 al 2023, obteniendo como resultados 365 publicaciones tanto de conferencias, cursos, revistes y libros.

(software) AND (Methodologies OR Procedures) AND (traditional OR Agil) AND (Best) AND (Practice OR Methods OR Techniques).

La selección de estudios, acompañada de la extracción de información, se presenta, asegurando que cumplan con los criterios de inclusión y exclusión validados mediante las reglas mencionadas anteriormente.

Selección de estudios y extracción de información

Para iniciar la investigación, se lleva a cabo la gestión y depuración de los resultados. Este proceso implica filtrar, seleccionar y clasificar los estudios previamente identificados en las etapas anteriores. Además, se evaluará la viabilidad de utilizar cada una de las publicaciones en medidas de su disponibilidad para la extracción desde la base científica que lo aloja.

La selección de cada artículo se realizó a través de dos filtros de revisión, el primero enfocado en determinar si el artículo se relaciona con el tema mediante el análisis del título, resumen o abstract y las palabras clave y el segundo filtro se basó en el contenido de cada uno de los artículos y trabajos relacionados en la investigación.

Una vez concluida esta etapa mediante el uso de un instrumento de selección se procederá a registrar todos los datos necesarios.

Registrar los datos en el instrumento de selección

En esta fase, a través de la creación de un instrumento, se procederá a clasificar las publicaciones para documentar aquellas que cumplen con los criterios de selección. Posteriormente, se presenta un análisis de los artículos que serán discutidos según los criterios definidos en el esquema de caracterización. Se observa la cantidad de estudios primarios utilizados en el proyecto de investigación, considerando los siguientes parámetros.

Durante la revisión de la literatura, se realiza una lectura cuidadosa de las publicaciones seleccionadas, analizando el título, el resumen y las palabras clave de los estudios primarios identificados. Estos campos proporcionan información coherente y relevante sobre el tema de investigación, enfocándose en la identificación de las mejores

prácticas para el proceso de desarrollo de software, tanto en metodologías tradicionales como no tradicionales.

Utilizando las cadenas de búsqueda generadas y la selección de publicaciones, se identificaron alrededor de 68 estudios en diversas bases bibliográficas. Con los nuevos campos establecidos en el instrumento de extracción de datos y de acuerdo a los criterios establecidos en secciones anteriores, se obtuvieron las publicaciones primarias que serán utilizadas para el estudio, como se observa en la Tabla 6.

Tabla 6

Instrumento de análisis de artículos científicos.

SID	Autores	Año	Tipo de publicación	Titulo
S01	Nathalie Skotnik; Andreas Frey	2022	Articulo	An Approach towards Reusability in Hybrid Avionic Software Development by Using a Unified Graph Representation of the Software System
S02	Alia Fátima; Tayyaba Rasool; Usman Qamar	2018	Articulo	GDGSE: Game Development with Global Software Engineering
S03	Ahmed M. Khan; Timothy D. Blackburn	2020	Articulo	AUTILE Framework: An AUTOSAR Driven Agile Development Methodology to Reduce Automotive Software Defects
S04	Pooja Sharma; Nitasha Hasteer	2016	Articulo	Analysis of linear sequential and extreme programming development methodology for a gaming application

Nota: En la tabla se observa el diseño del instrumento de recolección de información para el análisis de artículos científicos (Ver instrumento completo: https://docs.google.com/spreadsheets/d/1TzHtc8oZP8HKe9RQtcAWn3jyUgRNEwWlqGRnJ2KlvIA/edit?usp=sharing).

Identificación de las mejores prácticas

Se han identificado las mejores prácticas del proceso de desarrollo de software de metodologías tradicionales y no tradicionales aplicadas en la construcción de sistemas, a partir de las publicaciones científicas presentadas en los apartados anteriores. La Tabla 7 tiene como objetivo comprender el alcance de la aplicación de estas mejores prácticas en el

desarrollo de sistemas, así como identificar el campo o dominio en el que se utiliza. Los criterios utilizados se basan en la revisión de la literatura.

Tabla 7

Identificación de las mejores prácticas.

SID	Autores	Año	Titulo	Dominio	Tipo de contribución
S01	Nathalie Skotnik; Andreas Frey	2022	An Approach towards Reusability in Hybrid Avionic Software Development by Using a Unified Graph Representation of the Software System	Requisitos funcionales, Requisitos no funcionales, ciclo de vida	Aviación
S02	Alia Fátima; Tayyaba Rasool; Usman Qamar	2018	GDGSE: Game Development with Global Software Engineering	Modelos en Cascada, Espiral y Ágil Ciclo de vida, Requisitos funcionales y no funcionales, Planificación,	Industria de software
S03	Ahmed M. Khan; Timothy D. Blackburn	2020	AUTILE Framework: An AUTOSAR Driven Agile Development Methodology to Reduce Automotive Software Defects	Fase de diseño, Ciclo de vida, Calidad de software, Requisitos funcionales y no funcionales, Desarrollo ágil, Arquitectura	Automotriz
S04	Pooja Sharma; Nitasha Hasteer	2018	Analysis of linear sequential and extreme programming development methodology for a gaming application	Modelo XP, Modelo en cascada, Requisitos funcionales y no funcionales,	Industria de software

Nota: En la tabla se observa la identificación de las mejores prácticas para ser implementadas en el proceso de desarrollo de software (Ver instrumento completo: https://docs.google.com/spreadsheets/d/1TzHtc8oZP8HKe9RQtcAWn3jyUgRNEwWlqGRnJ2KlvIA/edit?usp=sharing).

Identificación de las mejores prácticas de desarrollo

En esta sección, se diseñará un instrumento para identificar y clasificar las mejores prácticas utilizadas en el proceso de desarrollo de software. Tras analizar las publicaciones

científicas, como resultado de esta investigación, se han identificado métodos y técnicas consideradas como mejores prácticas en el desarrollo de sistemas. Sin embargo, es importante tener en cuenta que estas mejores prácticas pueden variar dependiendo del campo o dominio en el que se apliquen. Estas variaciones pueden encontrarse entre diferentes compañías, institutos de investigación, e incluso dentro de las mismas empresas y universidades. Esta diversidad es un elemento crucial para mejorar el proceso de desarrollo del producto final y fomentar la innovación en el campo del desarrollo de software.

En el proceso de construcción de un sistema, es esencial diseñar las funcionalidades del software. Con el propósito de identificar las mejores prácticas que faciliten esta actividad, se extrajeron de las 68 publicaciones científicas investigadas las mejores prácticas. Para simplificar la identificación de estas mejores prácticas, se estableció un sistema de valores en el cual se asigna "0" si no se identifica una mejor práctica y "1" si se reconoce una mejor práctica en un aspecto específico del diseño de las funcionalidades del software. Este enfoque permite evaluar y reconocer las prácticas más efectivas para lograr un diseño óptimo de las funcionalidades de los sistemas, determinado dichos factores el siguiente paso será la fase de análisis.

Fase de Análisis

Después de llevar a cabo el análisis de requisitos en colaboración con el cliente, se generan diversos artefactos, los cuales varían según la metodología empleada en el proceso de desarrollo. Estos artefactos pueden incluir la Especificación de Requerimientos, Diagramas UML, Product Backlog, Sprints e Historias de Usuario, los cuales son considerados como mejores prácticas de desarrollo.

La generación de estos artefactos es resultado de un análisis de requisitos adecuado y se realizará tanto en metodologías tradicionales como en metodologías no tradicionales de desarrollo de software. Estos documentos y técnicas son herramientas

valiosas para capturar los requerimientos del cliente y establecer una base sólida para el desarrollo del software, de tal manera que procedemos a detallar el instrumento.

Descripción del Instrumento

En resumen, se han identificado las mejores prácticas de Ingeniería de Software aplicadas en el proceso de desarrollo de software. La Tabla 8 resume estas mejores prácticas identificadas en la fase de análisis. A continuación, se describen los nuevos campos agregados al instrumento:

Metodología: Este criterio indica si la publicación científica seleccionada corresponde a una metodología tradicional o no tradicional (ágil).

Especificación de requerimientos: Este campo define el comportamiento del sistema que se va a desarrollar. Dentro de la especificación de requerimientos se han identificado elementos como roles, requisitos funcionales, requisitos no funcionales, diagramas de casos de uso, diagramas de actividades y diagramas de flujo.

Product Backlog: Se refiere a una enumeración de todas las tareas que deben llevarse a cabo en un proyecto de software. Este registro abarca de manera integral las funcionalidades, mejoras y correcciones planificadas.

Sprints: Hace referencia a una lista de tareas más específicas y detalladas que deben completarse en un periodo de tiempo más corto y definido, usualmente dentro de un ciclo de desarrollo ágil. Estos sprints representan unidades de trabajo más manejables y focalizadas en comparación con el conjunto completo de tareas del proyecto.

Historias de usuario: Este campo permite conocer las historias de usuario, que son descripciones cortas y simples de las funcionalidades o características que se espera que el sistema tenga.

Tabla 8

Identificación de las mejores prácticas: Fase Análisis.

SID	Autores	Año	Titulo	Metodología	Análisis									
					Especificación de requerimientos			Diagramas UML			Product Backlog	Sprint	Historias de usuario	
					Roles	Requisitos funcionales	Requisitos no funcionales	Diagramas de caso de uso	Diagrama de actividades	Diagrama de flujo				
S01	Nathalie Skotnik; Andreas Frey	2022	An Approach towards Reusability in Hybrid Avionic Software Development by Using a Unified Graph Representation of the Software System	1	1	1	0	0	1	1	1	0	0	
S02	Alia Fátima; Tayyaba Rasool; Usman Qamar	2018	GDGSE: Game Development with Global Software Engineering	1	0	1	0	1	0	1	0	0	0	
S03	Ahmed M. Khan; Timothy D. Blackburn	2020	AUTILE Framework: An AUTOSAR Driven Agile Development Methodology to Reduce Automotive Software Defects	1	0	1	1	0	0	0	0	0	0	
S04	Pooja Sharma; Nitasha Hasteer	2018	Analysis of linear sequential and extreme programming development methodology for a gaming application	1	0	1	1	0	0	0	0	0	0	

Nota: (Ver instrumento completo:

https://docs.google.com/spreadsheets/d/1TzHtc8oZP8HKe9RQtcAWn3jyUgRNEwWlqGRnJ2KlvIA/edit?usp=sharing).

Fase de Diseño

Esta fase se centra en establecer la infraestructura que respaldará el producto de software. Durante esta etapa, se preparan todas las funcionalidades requeridas en base a los requisitos identificados en la fase de análisis. Aquí se definen el modelo de datos, los diagramas UML y los prototipos. La generación de estos artefactos sigue una metodología, ya sea tradicional o no tradicional, como se muestra en la Tabla 9. A continuación, se describe los nuevos campos agregados al instrumento:

Modelo de datos: Este aspecto determina los modelos de datos que se emplearán en el diseño del sistema para visualizar el proceso de desarrollo de software. Esto abarca elementos como el módulo físico, modelo entidad-relación, modelo conceptual, diagrama de secuencia, diagrama de estados, diagrama de componentes, diagrama de paquetes y diagrama de clases.

Prototipo: En este contexto, se define el diseño o prototipo que actuará como guía durante el proceso de desarrollo de los sistemas. Esto puede comprender tanto el diseño de software como el diseño de hardware.

Estos nuevos campos en el instrumento permiten capturar información relevante sobre las mejores prácticas utilizadas en la fase de diseño del proceso de desarrollo de software.

Tabla 9

Identificación de las mejores prácticas: Fase Diseño.

SID	Autores	Año	Titulo	Metodología	Diseño							Prototipo		
					Modelo de datos			Diagramas UML						
					Modelo físico	Modelo identidad relación	Modelo conceptual	Diagramas de secuencia	Diagrama de estado	Diagrama de componentes	Diagramas de paquetes	Diagrama de clases	Diseño de software	Diseño de hardware
S01	Nathalie Skotnik; Andreas Frey	2022	An Approach towards Reusability in Hybrid Avionic Software Development by Using a Unified Graph Representation of the Software System	1	0	1	1	0	0	1	0	0	0	1
S02	Alia Fátima; Tayyaba Rasool; Usman Qamar	2018	GDGSE: Game Development with Global Software Engineering	0	0	1	1	0	0	1	0	0	0	1
S03	Ahmed M. Khan; Timothy D. Blackburn	2020	AUTILE Framework: An AUTOSAR Driven Agile Development Methodology to Reduce Automotive Software Defects	0	0	1	1	0	0	1	0	0	0	1
S04	Pooja Sharma; Nitasha Hasteer	2018	Analysis of linear sequential and extreme programming development methodology for a gaming application	0	0	1	1	0	0	1	0	0	0	1

Nota: (Ver instrumento completo:

https://docs.google.com/spreadsheets/d/1TzHtc8oZP8HKe9RQtcAWn3jyUgRNEwWlqGRnJ2KlvIA/edit?usp=sharing).

Fase de Codificación

En esta etapa, se inicia la codificación de algoritmos y, en caso necesario, estructuras de datos que hayan sido previamente diseñadas en etapas anteriores. Convertir un diseño en código es una parte crucial de los procesos de Ingeniería de Software, ya que impacta de manera determinante en la calidad del producto final.

En algunos proyectos de software, se puede omitir ciertas etapas y avanzar directamente a la codificación. A continuación, se detallan las mejores prácticas aplicadas en la fase de codificación, como se muestra en la Tabla 10:

Documentación de código: Destaca la importancia de documentar el código como una de las mejores prácticas. La documentación de código actúa como una referencia para los desarrolladores de software, proporcionando información clave sobre el funcionamiento y la estructura del código.

Lenguaje de programación: Permite determinar qué paradigma de programación es más utilizado para la construcción de software. El lenguaje de programación elegido puede influir en la eficiencia, la portabilidad y otras características relevantes del software.

Estas mejores prácticas en la fase de codificación ayudan a garantizar la calidad y el rendimiento óptimo del software.

Tabla 10

Identificación de las mejores prácticas en la Fase de Codificación

SID	Autores	Año	Titulo	Metodología	Codificación	
					Documentación de código	P.O.O.
S01	Nathalie Skotnik; Andreas Frey	2022	An Approach towards Reusability in Hybrid Avionic Software Development by Using a Unified Graph Representation of the Software System	Tradicional	1	1
S02	Alia Fátima; Tayyaba Rasool; Usman Qamar	2018	GDGSE: Game Development with Global Software Engineering	Tradicional	0	1

SID	Autores	Año	Titulo	Metodología	Codificación	
					Documentación de código	P.O.O.
S03	Ahmed M. Khan; Timothy D. Blackburn	2020	AUTILE Framework: An AUTOSAR Driven Agile Development Methodology to Reduce Automotive Software Defects	No tradicional	1	1
S04	Pooja Sharma; Nitasha Hasteer	2018	Analysis of linear sequential and extreme programming development methodology for a gaming application	Tradicional	0	1

Nota: En la tabla se observa la identificación de las mejores prácticas para ser implementadas en el proceso de desarrollo de software en la fase de codificación (Ver instrumento completo: https://docs.google.com/spreadsheets/d/1TzHtc8oZP8HKe9RQtcAWn3jyUgRNEwWlqGRnJ2KlvIA/edit?usp=sharing).

Fase de Pruebas

El propósito de esta etapa es verificar que el software realice correctamente las actividades mencionadas en las etapas anteriores, con especial énfasis en la etapa de pruebas. Se sugiere llevar a cabo pruebas por módulos y, posteriormente, realizar pruebas integrales del software. En la Tabla 11 se detallan los componentes del instrumento que contienen las mejores prácticas para esta etapa.

Descripción del instrumento

Los artículos científicos analizados se obtuvieron mediante una revisión de literatura en bases de datos científicas, abarcando proyectos tanto de la industria como de instituciones académicas. El objetivo de esta revisión fue incorporar evidencia al proceso de desarrollo de software.

Pruebas Funcionales: Este criterio se utiliza para determinar las pruebas funcionales empleadas en las metodologías tradicionales y no tradicionales, considerando las características específicas de cada una.

Pruebas no Funcionales: Se identifican las pruebas no funcionales ofrecidas por las publicaciones científicas en la matriz de hallazgos establecida para esta investigación.

Pruebas Unitarias: A partir de la revisión literaria realizada, se han identificado las pruebas unitarias utilizadas en las metodologías tradicionales y no tradicionales.

Pruebas de Componentes: Se especifica la integración de componentes como una de las mejores prácticas establecidas por las metodologías de desarrollo de sistemas.

Pruebas de Aceptación: Estas pruebas se llevan a cabo en presencia del cliente y representan las últimas pruebas que se realizan al software.

Pruebas de Integración: Este criterio identifica las pruebas de integración para verificar la interacción entre los componentes del sistema.

Pruebas de Sistema: Este criterio identifica las pruebas de sistema como una de las mejores prácticas aplicadas al momento de probar un sistema. Estas pruebas representan la culminación del proceso de pruebas al construir un sistema.

Tabla 11

Identificación de las mejores prácticas en la Fase Pruebas.

SID	Autores	Año	Titulo	Pruebas						
				Pruebas funcionales	Pruebas no funcionales	Pruebas unitarias	Pruebas de componentes	Pruebas de aceptación	Pruebas de integración	Pruebas de sistema
S01	Nathalie Skotnik; Andreas Frey	2022	An Approach towards Reusability in Hybrid Avionic Software Development by Using a Unified Graph Representation of the Software System	1	0	1	0	0	0	1
S02	Alia Fátima; Tayyaba Rasool; Usman Qamar	2018	GDGSE: Game Development with Global Software Engineering	1	0	1	0	0	0	0
S03	Ahmed M. Khan; Timothy D. Blackburn	2020	AUTILE Framework: An AUTOSAR Driven Agile Development Methodology to Reduce Automotive Software Defects	1	1	0	0	0	0	0
S04	Pooja Sharma; Nitasha Hasteer	2018	Analysis of linear sequential and extreme programming development methodology for a gaming application	1	1	1	0	0	0	0

Nota: (Ver instrumento completo:

https://docs.google.com/spreadsheets/d/1TzHtc8oZP8HKe9RQtcAWn3jyUgRNEwWlqGRnJ2KlvIA/edit?usp=sharing).

Evolución de las mejores prácticas en el desarrollo de Sistemas

Con la selección de las mejores prácticas, para un análisis y distribución más efectivos, se ha establecido la revisión de la literatura abarcando el periodo comprendido entre los años 2018 y 2023. Los artículos científicos también se clasifican según el método de desarrollo, ya sea tradicional o no tradicional. Los ejemplos incluyen análisis de requisitos, diseño, codificación y pruebas (Gómez, 2012).

Descripción de criterios del instrumento para las mejores practicas

Tras un análisis retrospectivo de la literatura sobre desarrollo de software, hemos tomado la decisión de seleccionar y difundir las mejores prácticas. Este enfoque contribuirá nuevamente a la creación de un modelo funcional. Los estándares comunes utilizados en el instrumento (ver Tabla 9) son:

- SID: este criterio identifica y enumera publicaciones científicas seleccionadas que contienen una o mejores prácticas para los procesos de desarrollo de software.
- Año: Se considera un criterio obligatorio para posibilitar la clasificación y ordenación temporal de las publicaciones.
- Metodología: Este criterio ha sido seleccionado debido a que facilita la comprensión de la metodología adoptada en cada publicación, siendo fundamental para la clasificación de la investigación.

Al tener un conjunto de prácticas seleccionadas se puede determinar cuáles serán las mejores enfocándose en el aspecto tradicional como se detalla a continuación.

Capítulo IV

Análisis de resultados

Selección de mejores prácticas de las metodologías tradicionales 2018-2023

En la presente sección se proporciona un instrumento que cumplen con ciertos estándares, donde se puede seleccionar y difundir las mejores prácticas revisadas de 68 publicaciones científicas. Donde, las mejores opciones y tareas prácticas relacionadas con el alcance utilizadas en la construcción software, cada año desde 2018 hasta 2023 permitieron seleccionar las tendencias en mejores prácticas metodológicas tradicionales

Instrumento de selección para desarrollo de software: Fase Análisis

Descripción del Instrumento

Los criterios que se detallan a continuación representan los resultados del análisis llevado a cabo en etapas anteriores. Las mejores prácticas se estructuran de la siguiente manera:

- *Requisitos Funcionales y No Funcionales:* Este formulario estándar forma parte del documento de especificación de requisitos y su inclusión en el proceso de desarrollo es destacable.
- *Diagrama de Casos de Uso:* Permite una representación más precisa del desarrollo y proceso del software.

La Tabla 12 exhibe el análisis, selección y clasificación de las mejores prácticas en el proceso de desarrollo de software. Muchas de estas buenas prácticas surgen del proceso de ingeniería de software. Esta herramienta visualiza la selección y asignación de buenas prácticas de desarrollo a la creación de cada ciclo de vida del software.

Existe un estándar para el diseño del instrumento. Por ejemplo, SID, año, metodología, requisitos funcionales, requisitos no funcionales y diagramas de casos de uso. Como puede ver después de seleccionar la publicación, las mejores prácticas

seleccionadas incluyen requisitos funcionales, requisitos no funcionales y diagramas de casos de uso, los mismos que fueron utilizados para el desarrollo del instrumento de selección.

Metodología tradicional, permite entender que desde principios de la década del 2018 hasta el 2023, la construcción de software se basó en la aplicación de las mejores prácticas. Es importante destacar que los métodos tradicionales o clásicos de desarrollo de software se fundamentan en el ciclo de vida del desarrollo de software, promoviendo así la construcción centralizada del software (Montero et al., s.f.).

Así, los primeros modelos publicados se acercan a la idea de establecer un estado a través del proceso de desarrollo de software de sistemas de información, son Extracto de Desarrollo de Software, como se ve en la tabla 12.

Tabla 12

Instrumento de selección de mejores prácticas: Fase Análisis

SID	Año	Metodología	Requisitos funcionales	Requisitos no funcionales	Diagrama de caso de uso
S01	2018	Tradicional	1	2	0
S02	2019	Tradicional	1	2	0
S09	2020	Tradicional	1	2	0
S23	2021	Tradicional	1	2	3
S36	2023	Tradicional	1	2	3

Nota: Esta tabla, muestra la selección de los criterios de requisitos funcionales, no funcionales y el diagrama de caso de uso

Como se puede observar en la Tabla 13, en la selección de artículos y publicaciones, se puede asignar de forma numérica a cada mejor práctica identificada, uno de estos criterios para obtener el número total de frecuencias utilizadas para cada criterio.

Tabla 13

Mejores prácticas de la Fase de Análisis 2018 -2023

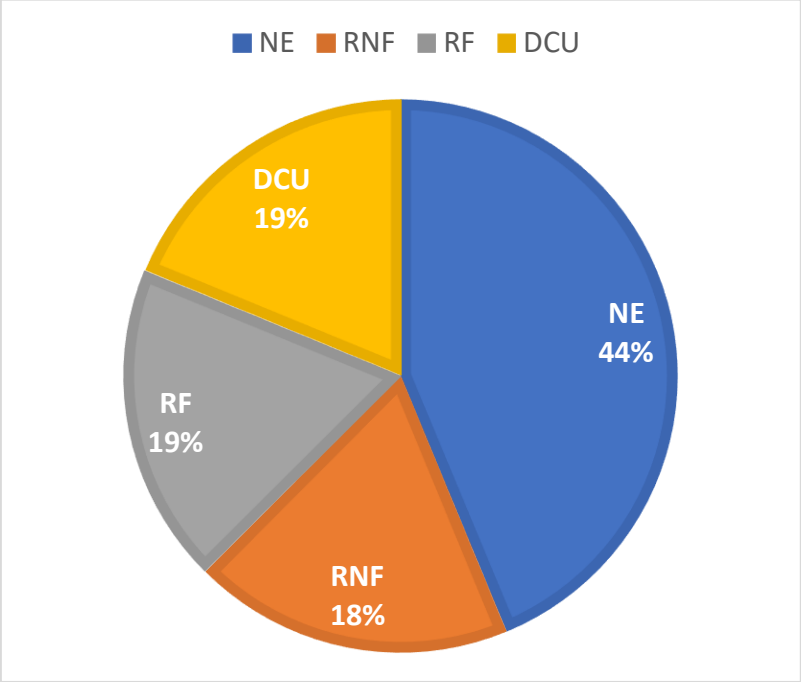
Criterios	Peso	Abreviaturas	Colores
No especifica	0		AZUL
Requisitos funcionales	1	RF	PLOMO
Requisitos no funcionales	2	RNF	TOMATE
Diagrama de Caso de uso	3	DCU	AMARILLO

Nota: Esta tabla, muestra las mejoras prácticas de las metodologías del año 2018-2023

En la Figura 5, se puede observar el resumen de las buenas prácticas de acuerdo con el número de frecuencia de uso

Figura 5

Porcentaje de buenas prácticas



Interpretación:

Después de analizar y categorizar las mejores prácticas del proceso desarrollo de software, se han identificado 5 métodos, compartiendo un 18,74% en el caso de requisitos funcionales y no funcionales y un 18,77 para el uso de diagramas de casos de uso que se

enfocan en los métodos tradicionales. Sin embargo, se encontró que 43,74% no especificaba.

Instrumento de selección para el desarrollo de software: Fase Diseño

Esta fase explora el diseño del flujo de desarrollo. En detalle, comenzamos con la identificación de las publicaciones que se centran en las mejores prácticas de diseño software, para lo cual se creó un instrumento para revisar las 68 publicaciones, el total de artículos propuestos para revisión.

Descripción del instrumento

El instrumento para seleccionar y clasificar las mejores prácticas con el proceso de investigación utilizado para este proceso se estructura de la siguiente manera, con el proceso de desarrollo de software se tiene los siguientes criterios

- Diagrama de secuencia: le permite comprender y describir interacciones. Comportamiento dinámico de los sistemas.
- Diagrama de Clases: Este estándar se ha establecido debido a su capacidad para facilitar el uso del paradigma orientado a objetos. Sirve como una guía para la codificación orientada a objetos en el desarrollo de software.

Sin embargo, según la revisión bibliográfica, su aplicación apareció entre los años 2018 y 2023 para la construcción de software a partir de la aplicación de las mejores prácticas. Cabe señalar que la metodología tradicional o clásica existe desde 1966 (Gabriel, s.f.), Este enfoque ha permitido establecer una base sólida para el desarrollo de diversos tipos de software, perdurando hasta la actualidad.

Como se evidencia en la Tabla 14, a cada criterio se le puede asignar una puntuación numérica para enumerar y clasificar cada mejor práctica identificada. La frecuencia general se obtiene ponderando cada criterio.

Tabla 14

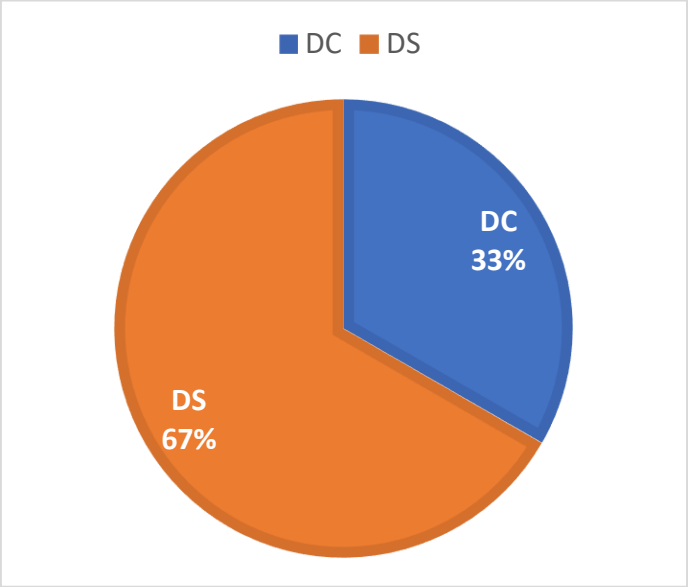
Mejores prácticas de la Fase de Diseño 2018-2023.

Criterios	Peso	Abreviaturas	Colores
No especifica	0		Tomate
Diagrama de clases	1	DC	AZUL
Diagrama de secuencia	2	DS	TOMATE

Además, se instalaron semáforos para una mejor interpretación visual.

Figura 6

Resumen de buenas prácticas



Interpretación:

Después de analizar y categorizar las mejores prácticas del proceso de desarrollo, la revisión de la literatura y los datos del semáforo cumple con el 66,67% en el caso de diagramas de secuencia y un 33,33% para los diagramas de clases.

Instrumento de selección para el desarrollo de software: Fase Codificación.

Esta fase explora la codificación para el proceso de desarrollo de sistemas. En detalle, continuamos identificando publicaciones que promueven las mejores prácticas de desarrollo

de software. Se creó un instrumento para revisar las 68 publicaciones, el total de artículos propuestos para revisión.

Descripción del instrumento

El instrumento para la fase de análisis y clasificación de mejores prácticas está estructurado de la siguiente manera:

- Documentación de código: le permite agregar información para explicar cómo funciona el código para que no solo las computadoras sepan qué hacer, sino que los humanos entiendan qué se está haciendo y por qué.

- Paradigma de Programación Orientada a Objetos: Este estándar permite administrar la complejidad del software, este tipo de paradigma es adecuado para desarrollar y mantener aplicaciones y grandes estructuras de datos.

Como se muestra en la Tabla 15, a cada criterio se le puede asignar una calificación numérica para enumerar y clasificar cada mejor práctica identificada, y la frecuencia general se obtiene ponderando cada criterio. Además, se instalaron semáforos para una mejor interpretación visual.

Tabla 15

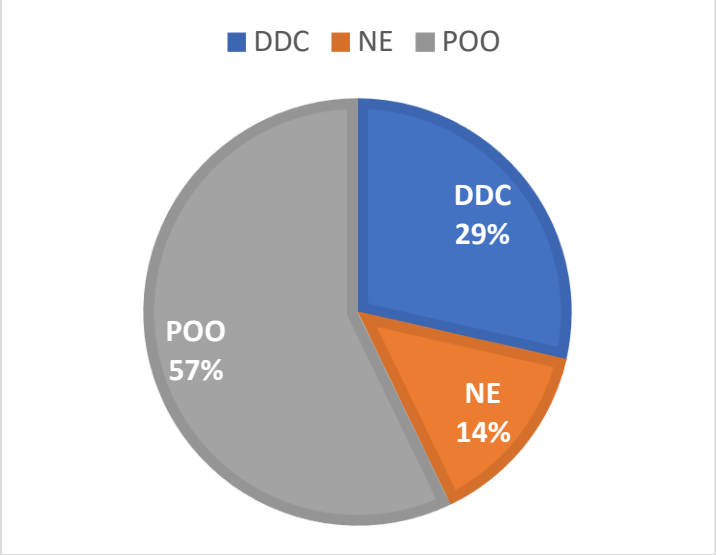
Mejores prácticas de la Fase de Codificación 2022-2023.

Criterios	Peso	Abreviaturas	Colores
No especifica	0		TOMATE
Documentación de código	1	DDC	AZUL
Programación orientada a objetos	2	POO	PLOMO

Nota: En esta Tabla muestra las mejores prácticas de la fase de codificación 2018 al 2023 con sus debidas abreviaturas

Figura 7

Porcentaje de mejores prácticas establecido



Interpretación:

Con el proceso de selección de las mejores prácticas, se determina que en la industria existe un porcentaje mínimo de desconocimiento al momento de aplicar los conceptos de buenas prácticas en el desarrollo de un proyecto, manteniendo una tendencia centrada a cero, además, se establece que en la industria actual se mantiene un mínimo cuando se habla de documentar y se establece un estándar al momento de utilizar programación orientada a objetos.

Instrumento de selección para el desarrollo de software: Fase Pruebas

El uso de pruebas en el desarrollo de software es de suma importancia. Por ello, seguimos buscando publicaciones que promuevan las mejores prácticas centradas en su implementación en compilaciones de software. Hemos creado un instrumento donde puede ver 68 publicaciones, que es el número total de artículos enviados para investigación.

Demostración del instrumento

Para el proceso de investigación utilizado en el proceso de desarrollo de software, el instrumento se estructura de la siguiente manera, y los criterios que se describen a continuación son el resultado de los análisis realizados en las etapas anteriores.

- Pruebas funcionales y no funcionales: permiten verificar que el software está funcionando de acuerdo con las especificaciones de requisitos funcionales.

- Pruebas unitarias o de componentes: Este estándar le permite probar la correcta funcionalidad de una unidad de código.

- Prueba de aceptación: este tipo de prueba la realiza el cliente para verificar la funcionalidad del producto de software integrado.

- Pruebas de integración: permite probar la comunicación e interacción entre los componentes de hardware y software.

Como se muestra en la Tabla 16, para enumerar y clasificar cada mejor práctica identificada, a cada criterio se le puede asignar una calificación numérica, y la frecuencia general se obtiene aplicando pesos a cada criterio. Además, se instalaron semáforos para una mejor interpretación visual.

Tabla 16

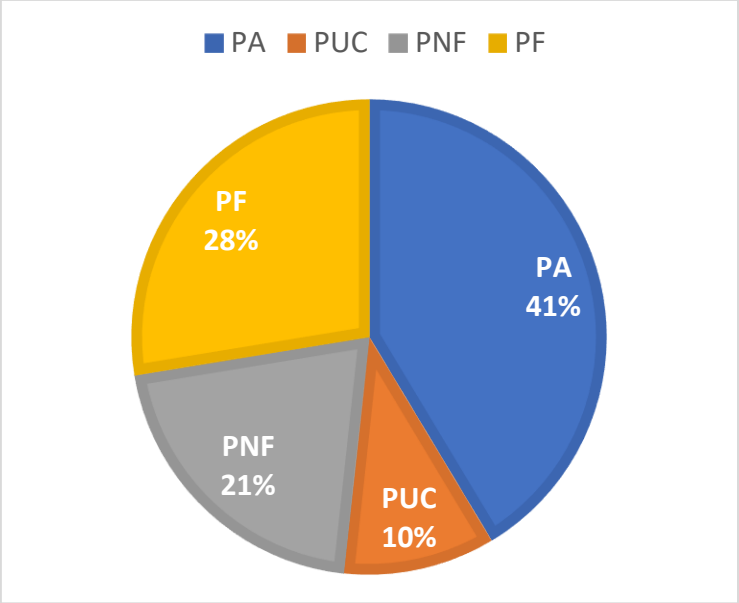
Mejores prácticas de la Fase de Pruebas 2018-2023.

Criterios	Peso	Abreviaturas	Colores
No especifica	0		Celeste
Pruebas funcionales	1	P. F	AMARILLO
Pruebas no funcionales	2	P.N. F	PLOMO
Pruebas unitarias o componentes	3	P.U.C	TOMATE
Pruebas de aceptación	4	PA	AZUL

Nota: La siguiente tabla presenta las mejores prácticas de las metodologías tradicionales identificadas desde el año 2018 hasta el 2023.

Figura 8

Porcentaje de mejores prácticas establecido



Interpretación:

El análisis realizado se puede determinar que, al momento de aplicar buenas prácticas en la etapa de pruebas, la mayoría de los casos presenta un índice de aplicación de entre 10% al 20% en la aplicación de pruebas funcionales y no funcionales, en el caso de pruebas unitarias existe un estándar del 30% de aceptación, en uso y aplicación, por último, siendo las más conocidas al estar en contacto tanto con desarrolladores como QA las pruebas de aceptación con un 40%

Selección de mejores prácticas de metodologías no tradicionales 2018-2023

Esta sección analiza y difunde las mejores prácticas aplicadas en la construcción de software para el volumen anual, de 68 publicaciones científicas desde 2018 hasta 2023. Se pueden observar tendencias que van desde las mejores prácticas hasta enfoques no convencionales

Instrumento de selección para el desarrollo de software: Fase Análisis

Descripción del instrumento

El instrumento de análisis y clasificación de mejores prácticas está estructurado de la siguiente manera:

Roles: Estos criterios son parte del documento de especificación de requisitos y desempeñan un papel significativo en el proceso de desarrollo.

Requisitos funcionales y no funcionales: Forma parte del documento de especificación de requisitos y proporciona información sobre la funcionalidad del software.

Diagrama de casos de uso: Permite representar de manera más concreta y clara los procesos de desarrollo.

Product backlog: En métodos ágiles, este criterio permite conocer la lista de requisitos necesarios.

Historias de usuarios: Facilita que el usuario se comunique en su lenguaje nativo.

El instrumento visualiza el análisis y clasificación de las buenas prácticas por frecuencia de uso en la estructura proporcionada. La buena práctica se determina según la metodología, utilizando en este caso 15 artículos científicos que formaban parte del método tradicional.

Para evaluar la frecuencia con la que las compilaciones de software emplean estas mejores prácticas, se agrupan según su uso. Se asignan colores para identificar y numerar las mejores prácticas, permitiendo verificar la cantidad de estas prácticas y ofreciendo información sobre las publicaciones de investigación de los científicos.

Tabla 17

Semaforización de las mejores prácticas Fase Análisis

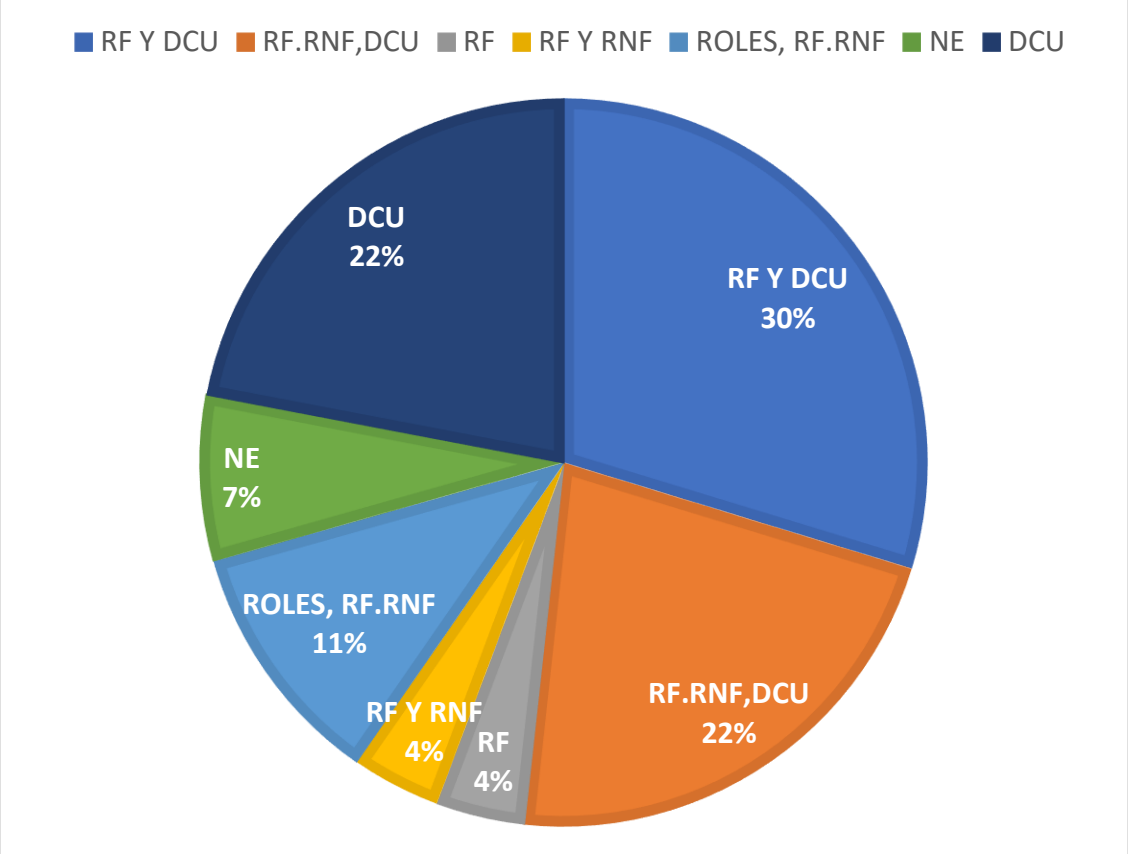
Color	Criterio	Abreviaturas
Azul	Roles	ROLES
	Product Backlong	P. BACK
AZUL	Roles	RF
	Historias de usuario	DCU
CIAN	Requisitos funcionales	RF
	Diagramas de caso de uso	DCU
	Product Backlong	P. BACK
	Historias de usuario	H. USUARIOS
TOMATE	Roles	DCU
	Requisitos funcionales	RF
	Requisitos no funcionales	RNF
VERDE	No Especificado	NO ESPECIFIH.CADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas de la fase de análisis identificadas del 2018 al 2023

En la Figura 9 se presenta un resumen de las mejores prácticas en función de la frecuencia de uso, y para una comprensión visual más clara, se continúa empleando los indicadores visuales previamente establecidos. En este contexto, se detallará el porcentaje de artículos científicos asociados a cada práctica.

Figura 9

Fase de Análisis: Porcentaje de aplicación de las mejores prácticas.



Instrumento de selección para el desarrollo de software: Fase de Diseño.

Descripción del instrumento

El instrumento de análisis y clasificación de mejores prácticas se estructura de la siguiente manera:

- Modelo de Relación de Entidades: Este estándar facilita la representación de entidades, como objetos o cosas, para modelar la base de datos cuyos datos son necesarios durante la fase de desarrollo del software.

- Modelo conceptual: Este estándar permite la representación del software; el modelo ayuda a las personas a comprender, conocer y simular el sistema.

- Diagrama de Clase: Este estándar se creó para permitir el uso de un paradigma orientado a objetos y servir como guía para la codificación del software.

Para determinar con qué frecuencia se usa una mejor práctica en las compilaciones de software, las mejores prácticas se agrupan para este propósito, se crea un color para identificar y enumerar las mejores prácticas, para que pueda controlar la cantidad de mejores prácticas y proporcionar publicaciones de investigación utilizando científicos.

Tabla 18

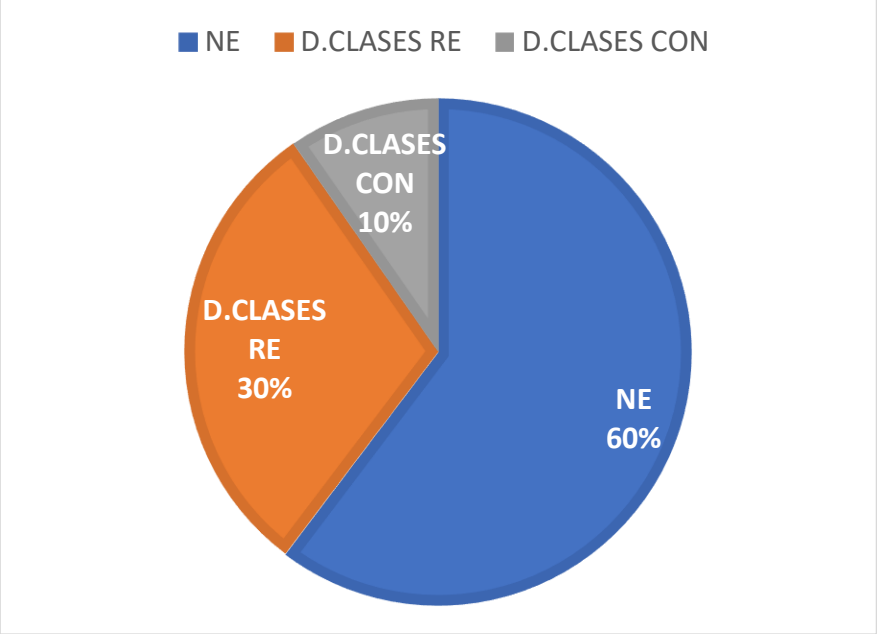
Evaluación de las mejores prácticas en la Fase de Análisis.

Color	Criterio	Abreviaturas
PLOMO	Modelo conceptual	M. CONCEPTUAL
	Diagrama de clases	D. CLASES
TOMATE	Modelo Entidad Relación	M. E. RELACIÓN
	Diagrama de clases	D. CLASES
AZUL	No Especificado	NO ESPECIFIH.CADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en el 2018 al 2023 en la fase de análisis.

Figura 10

Fase Diseño: Porcentaje de aplicación de mejores prácticas.



Instrumento de selección para el desarrollo de software: Fase Codificación

Descripción del instrumento

Para el proceso de investigación utilizado en el desarrollo de software, el instrumento de clasificación y análisis de mejores prácticas se estructuran de la siguiente manera. Los criterios que se describen a continuación son el resultado de los análisis realizados en las etapas anteriores como se ve en la Tabla 19.

- Documentación del código: le permite agregar información para explicar cómo funciona el código para que no solo las computadoras sepan qué hacer, sino que los humanos entiendan qué se está haciendo y por qué.
- Paradigma de Programación Orientada a Objetos: Este estándar permite administrar la complejidad del software, este tipo de paradigma es adecuado para desarrollar y mantener aplicaciones y grandes estructuras de datos.

Tabla 19

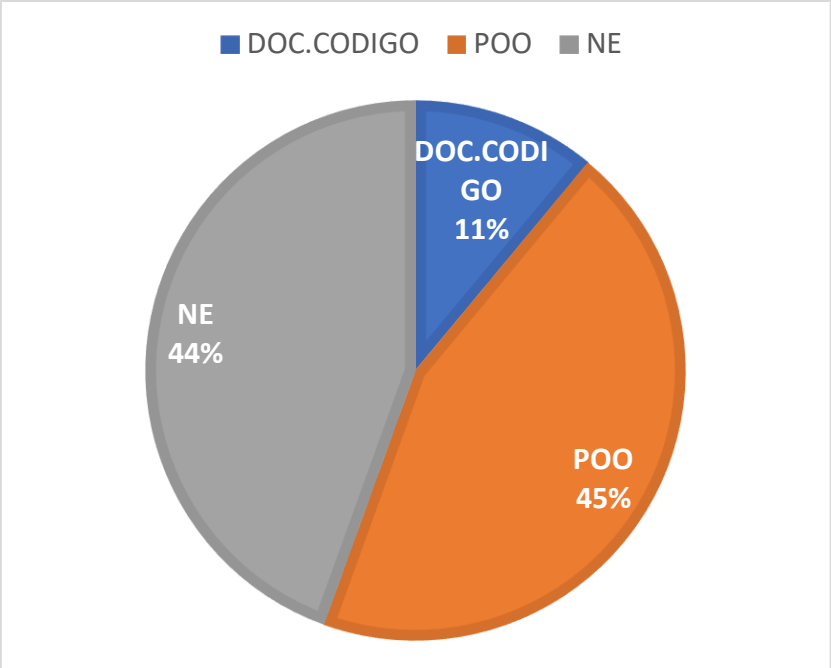
Semaforización de mejores prácticas Fase Codificación.

Color	Criterio	Abreviaturas
PLOMO	Documentación de código	DOC. CODIGO
TOMATE	Programación orientada a objetos	P.O.O.
AMARILLO	No Especificado	NO ESPECIFIH.CADO

Nota: En esta tabla, se presenta la codificación de las mejores prácticas identificadas en las metodologías no tradicionales entre 2011 y 2020, específicamente en la fase de codificación, utilizando un sistema de semáforos.

Figura 11

Fase de Codificación: Porcentaje de Implementación de las Mejores Prácticas.



Instrumento de selección para el desarrollo de software: Fase Pruebas

Descripción del instrumento

Los módulos que conforman el instrumento de análisis y clasificación de mejores prácticas estarán estructurados de la siguiente manera:

- Pruebas funcionales y no funcionales: Esto facilita la verificación de que el software opera conforme a las especificaciones de requisitos funcionales y no funcionales.
- Pruebas unitarias: este estándar le permite probar unidades de código para una funcionalidad correcta.
- Prueba de aceptación: este tipo de prueba la realiza el cliente para verificar la funcionalidad del producto de software integrado.

Semaforización de la frecuencia de uso de las mejores practicas

La semaforización de la frecuencia de uso de las mejores prácticas implica agruparlas mediante el uso de colores específicos como una forma de identificarlas y contarlas. Posteriormente, se lleva a cabo un control de las mismas y del número de publicaciones científicas que contribuyen a este estudio para determinar con qué frecuencia se utilizan. Este enfoque permite una visualización clara y coherente de la prevalencia y el impacto en el desarrollo de software.

Tabla 20

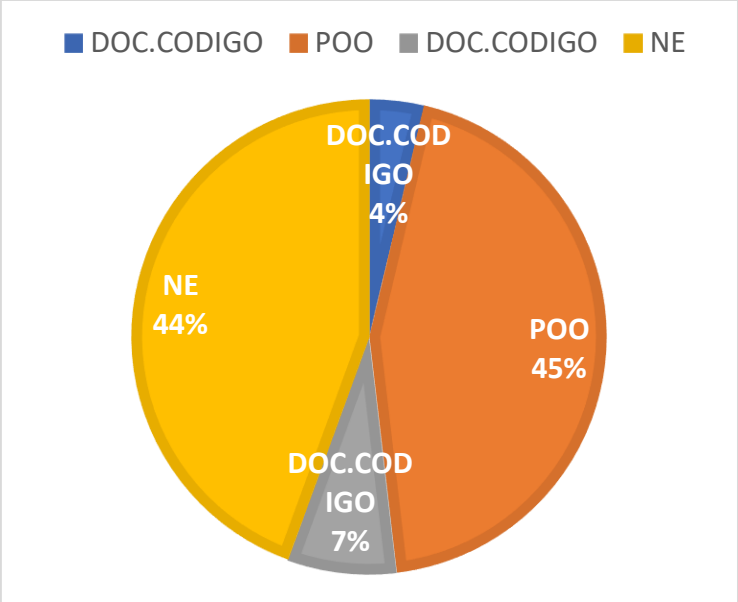
Indicadores visuales de las mejores prácticas en la Fase de Pruebas.

Color	Criterio	Abreviaturas
AZUL	Pruebas funcionales	P.FUN
PLOMO	Pruebas funcionales	P.FUN
	Pruebas unitarias	P.UNI
	Pruebas de integración	P.INT
Color	**Criterio**	**Abreviaturas**
TOMATE	Pruebas unitarias	P.UNI
	Pruebas de aceptación	P. ACEP
	Pruebas de integración	P.INT
AMARILLO	No Especificado	NO ESPECIFICADO

Nota: Observa la tabla que presenta la codificación de colores para resaltar las mejores prácticas identificadas en las metodologías no tradicionales durante la fase de pruebas, desde el año 2018 hasta el 2023.

Figura 12

Evaluación de las mejores prácticas durante la fase de pruebas: Porcentaje de implementación.



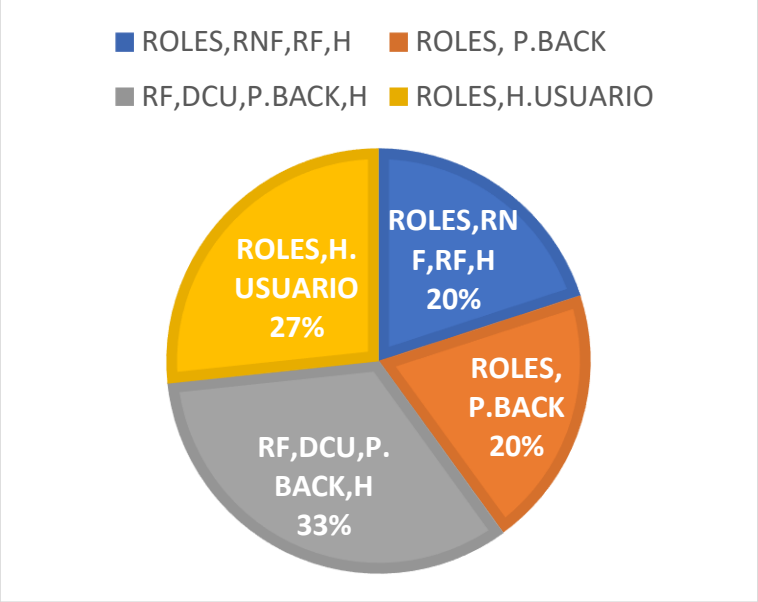
Principales hallazgos de mejores prácticas para la construcción de software

Los estudios científicos examinados representan investigaciones que abordan la implementación de las mejores prácticas en el desarrollo de software. Con el fin de identificar las buenas prácticas más utilizadas en los enfoques tradicionales y no tradicionales, se elaboró un mapa general, que se describe brevemente a continuación:

Como se evidencia en la Figura 30, en el periodo comprendido entre 2018 y 2023, las metodologías tradicionales se distribuyen de la siguiente manera: un 44,44% no especifica, solo un 7,41% documenta el código, un 44,44% utiliza programación orientada a objetos y un 3,70% aplica ambas prácticas.

Figura 13

Resultados clave en relación con las mejores prácticas: Fase de Diseño.

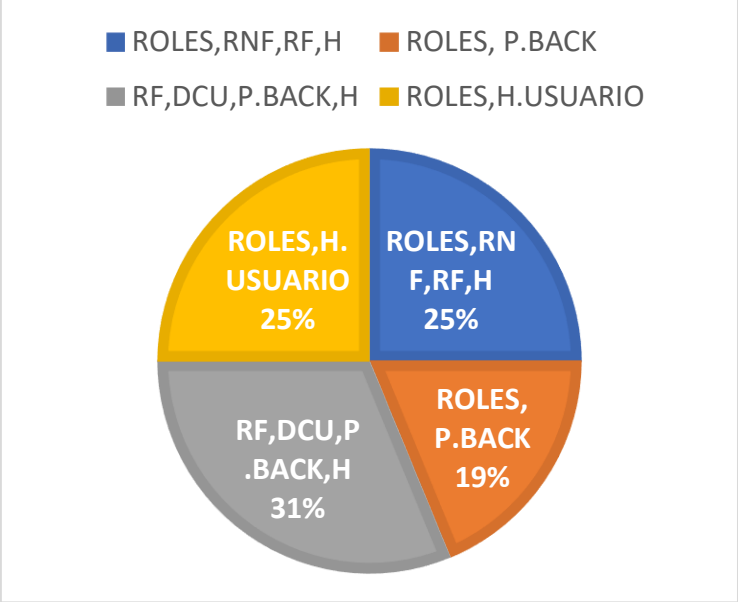


Mejores prácticas en la fase de codificación

Como se puede observar Figura 31, respecto a los años 2018 al 2023 se hallan distribuidas en metodologías tradicionales en donde se ha identificado que el 20% corresponden al uso de Roles que relacionan los requisitos funcionales y no funcionales con las historias de usuario, un 20% además mantiene esta relación entre roles y pruebas unitarias, con un 333,33% que combina las anteriormente mencionadas con las historias de usuario y un 26,67 solo maneja historias de usuario.

Figura 14

Principales descubrimientos acerca de las mejores prácticas: Fase de Codificación

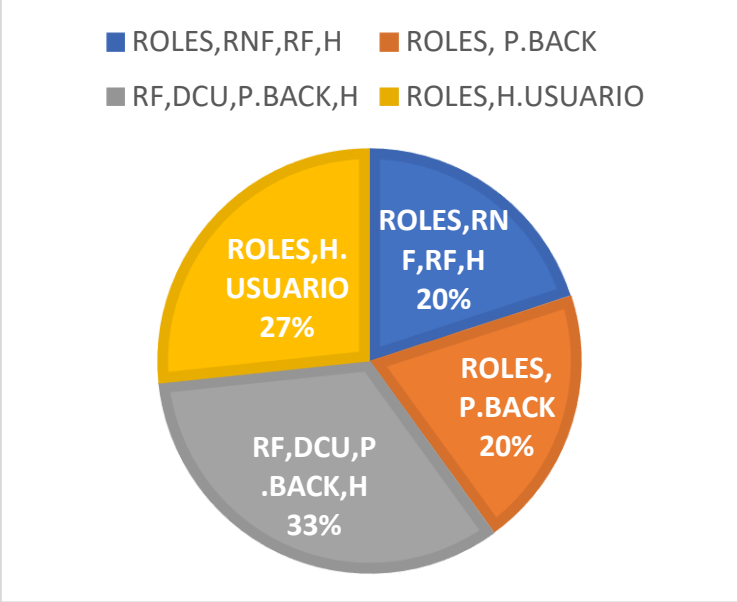


Mejores prácticas en la fase de pruebas

Como se puede apreciar en la Figura 32, durante el periodo de 2018 a 2023, se evidencia el empleo de pruebas funcionales, pruebas no funcionales y pruebas unitarias con una frecuencia del 20% en la aplicación de metodologías tradicionales. Se identifica la aplicación de una única práctica, específicamente pruebas funcionales, con un 20%. En el mismo intervalo temporal y en referencia a las metodologías no tradicionales, se observa que la aplicación de pruebas funcionales y no funcionales, pruebas unitarias y pruebas de aceptación se mantiene en un 33,33%, mientras que el uso de historias de usuario equivale al 26,67%.

Figura 15

Principales hallazgos de las mejores prácticas: Fase de Pruebas.



Propuestas actuales de mejores prácticas para el desarrollo de software

Actualmente, diversos estudios han abordado distintos campos y aplicaciones de las mejores prácticas con el objetivo de asegurar la calidad de los productos de software. Estos enfoques buscan establecer un equilibrio en la creación de software, donde la metodología de desarrollo está diseñada para guiar a los desarrolladores a lo largo de las etapas del proceso de software integrado. Además, estas metodologías ofrecen una respuesta efectiva a los cambios que puedan surgir durante la operación del sistema o del software. En este contexto, se examinan en detalle los métodos principales, como la Propuesta de Desarrollo de Sistemas para la identificación de buenas prácticas, así como los enfoques tradicionales y no tradicionales que respaldan la creación del proceso. La efectividad del modelo de diseño de sistemas integrados se destaca como un aspecto esencial, y se busca aplicar mejores prácticas que sean consideradas comunes y que presenten un valor igual o superior a 5.

Tabla 21

Propuestas de las mejores prácticas para el desarrollo de software

Mejores practicas	Fases					Pruebas a favor	Puntos en contra
	Especificación Requerimientos	Análisis de Requisitos	Diseño	Desarrollo de Software	Pruebas		
Requisitos funcionales	x					Especifican los aspectos funcionales que el software deba realizar	Se enfoca únicamente en el levantamiento de requisitos funcionales y no contempla requisitos no funcionales
Requisitos no funcionales	x					Especifican los comportamientos específicos del software.	Se enfoca únicamente en el levantamiento de requisitos no funcionales
Diagrama de caso de uso		x				Describe las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso.	Los casos de uso nos permiten tener la capacidad de saber explícitamente lo que el cliente necesita
Historia de usuario		x				Documento informal de los requisitos, en caso de ausencia de comunicación, con relación al proceso de Prueba y Aceptación.	Difícil de medir la eficacia de un proyecto con los resultados de una historia
Diagrama de secuencia			x			Describen el comportamiento de un sistema.	Representa estados que puede adquirir una clase.
Diagrama de clases			x			Representa un conjunto de clases que forman parte de un sistema	Los diagramas de clases son estáticos

Mejores practicas	Fases					Pruebas a favor	Puntos en contra
	Especificación Requerimientos	Análisis de Requisitos	Diseño	Desarrollo de Software	Pruebas		
Modelo conceptual			x			Identifica relaciones entre diferentes entidades	No especifica ningún atributo y claves
Documentación de código				X		Permite evitar errores y herramientas de aprendizaje que están a mano para nuevos empleados.	La documentación puede retrasar un proyecto
P. Orientada a objetos				X		Código fácil de mantener	Curva de aprendizaje. Comprensión de conceptos es un desafío
Pruebas funcionales					X	Centra en la experiencia del usuario.	Se limitan a probar qué tan bien una aplicación hace lo que se supone que hace posible perder errores fuera de este alcance
Pruebas de aceptación					X	Se realizan por el cliente para determinar la confianza del sistema a nivel funcional y técnico.	El cliente no comprenda las limitaciones técnicas que puede tener el software.
Pruebas unitarias					x	Permite comprobar un fragmento de código funciona de manera correcta	No identifica todos los defectos del código

Nota: Observamos en esta tabla las sugerencias de las mejores prácticas para el desarrollo de software obtenidas de publicaciones científicas.

Validación del modelo de trabajo para ayudar al desarrollo de software.

El desarrollo de este proyecto presenta el uso de una metodología de trabajo en la empresa de ACR DIGITALD S.A.S las cuales fueron evaluados por un instrumento de recopilación de información denominada encuesta misma que se realizaron a los mueve trabajadores de la compañía para conocer su perspectiva de la aplicación de la metodología de trabajo.

Validación del Sistema

La eficacia de la aplicación de un proceso metodológico en el desarrollo se demostró mediante la implementación experimental y la consecuente aplicación de una encuesta, mostrando los siguientes resultados:

Pregunta 1: Creo que me gustaría utilizar este método de trabajo con frecuencia

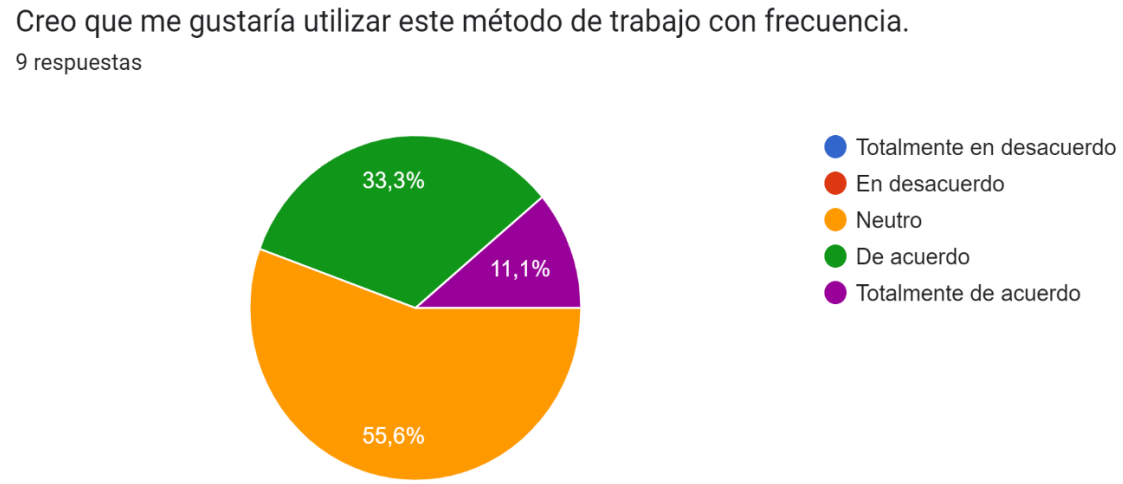
Tabla 22

Pregunta 1

Creo que me gustaría utilizar este método de trabajo con frecuencia		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	0	0%
En desacuerdo	0	0%
Neutro	5	55.6%
De acuerdo	3	33.3%
Totalmente de acuerdo	1	11.1%
Total	**9**	**100%**

Figura 16

Pregunta 1



Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 5 de ellas perteneciente al 55.6% mencionaron que su opción es de forma neutra ante la pregunta en cuestión mientras que 3 personas dando el 33.3% señalaron que están de acuerdo en utilizar este método y 1 personas dando el 11.1% está totalmente de acuerdo en usar el método de trabajo en frecuencia.

Pregunta 2: Encontré la metodología de trabajo innecesariamente complejo

Tabla 23

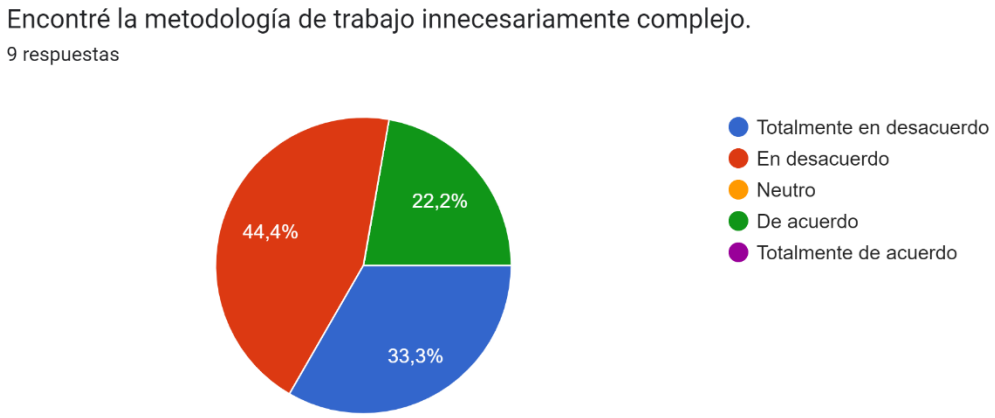
Pregunta 2.

Encontré la metodología de trabajo innecesariamente complejo		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	3	33.3%
En desacuerdo	4	44.4%

Encontré la metodología de trabajo innecesariamente complejo		
Variable	**Frecuencia**	**Porcentaje**
Neutro	0	0%
De acuerdo	2	22.2%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 17

Pregunta 2



Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 3 de ellas perteneciente al 33.3% mencionaron que están totalmente en desacuerdo ante la interrogante planteada mientras que 4 personas dando el 44.4% señalaron que están en desacuerdo y 2 personas dando el 22.2% están de acuerdo en que la utilización de la metodología es innecesariamente compleja

Pregunta 3: Pensé que la metodología de trabajo era fácil de manejar.

Tabla 24

Pregunta 3

Pensé que la metodología de trabajo era fácil de manejar.		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	0	0%
En desacuerdo	1	11.1%
Neutro	4	44.4%
De acuerdo	4	44.4%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 18

Pregunta 3

Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 1 de ellas perteneciente al 11.1% mencionaron que están en desacuerdo ante la interrogante planteada mientras que 4 personas dando el 44.4% señalaron la opción neutra y 4 personas dando el 44.4% están de acuerdo en que la metodología de trabajo es fácil de manejar

Pregunta 4: Creo que necesitaría el apoyo de un técnico para poder utilizar la metodología de trabajo

Tabla 25

Pregunta 4

Creo que necesitaría el apoyo de un técnico para poder utilizar la metodología de trabajo		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	6	66.7%
En desacuerdo	2	22.2%
Neutro	0	0%
De acuerdo	1	11.1%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 19

Pregunta 4

Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 6 de ellas perteneciente al 66.7% mencionaron que están

totalmente en desacuerdo ante la pregunta planteada mientras que 2 personas dando el 22.2% señalaron que están en desacuerdo y 1 personas dando el 11.1% está de acuerdo en que se necesitaría el apoyo de un técnico para usar la metodología de trabajo.

Pregunta 5: Encontré que las diversas funciones de la metodología de trabajo estaban bien integradas.

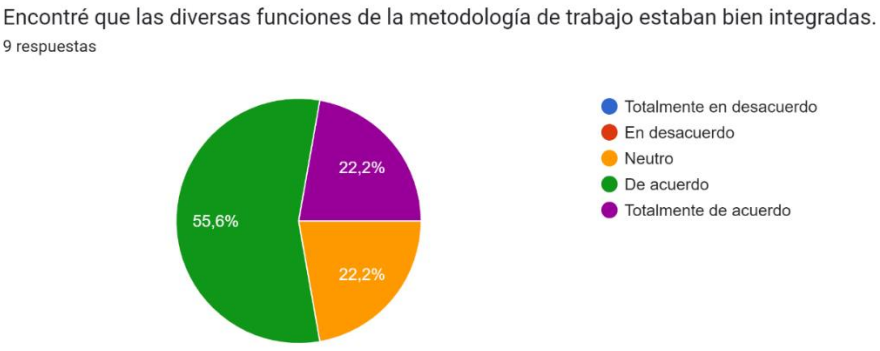
Tabla 26

Pregunta 5

Encontré que las diversas funciones de la metodología de trabajo estaban bien integradas		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	0	0%
En desacuerdo	0	0%
De acuerdo	5	55.6%
Totalmente de acuerdo	2	22.2%
Total	**9**	**100%**

Figura 20

Pregunta 5



Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 2 de ellas perteneciente al 22.2% señalaron la opción neutra ante la pregunta planteada mientras que 5 personas dando el 55.6% señalaron que están de acuerdo y 2 personas dando el 22.2% están totalmente de acuerdo en que las diversas funciones de la metodología de trabajo están bien integradas.

Pregunta 6: Pensé que había demasiada inconsistencia en la aplicación de la metodología de trabajo.

Tabla 27

Pregunta 6

Pensé que había demasiada inconsistencia en la aplicación de la metodología de trabajo.		
Variable	Frecuencia	Porcentaje
Totalmente en desacuerdo	2	22.2%
En desacuerdo	4	44.4%
Neutro	2	22.2%
De acuerdo	1	11.1%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 21

Pregunta 6

Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 2 de ellas perteneciente al 22.2% mencionaron que están totalmente en desacuerdo ante la interrogante planteada mientras que 4 personas dando el 44.4% señalaron que están en desacuerdo, 2 personas dando el 22.2% señalaron la opción neutra y 1 persona dando el 11.1% está de acuerdo en que había demasiada inconsistencia en la aplicación de la metodología de trabajo.

Pregunta 7: Me imagino que la mayoría de la gente aprendería se adaptaría a la metodología de trabajo muy rápidamente.

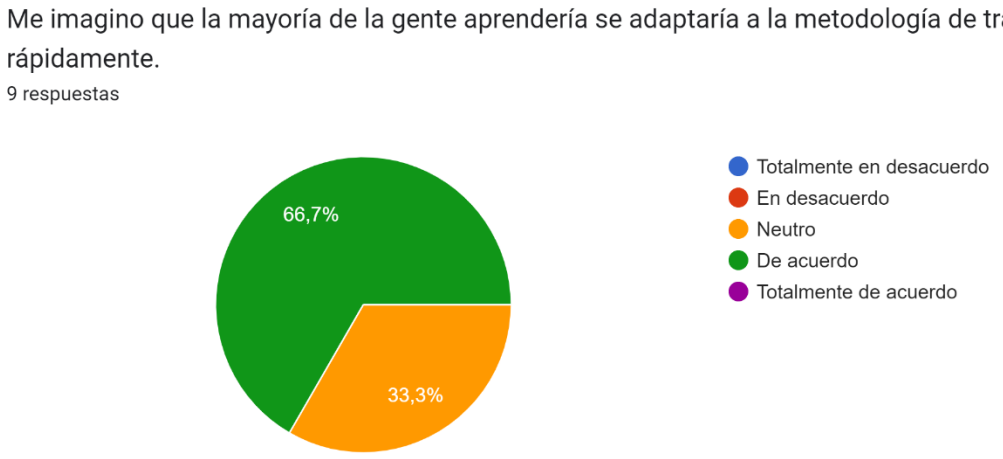
Tabla 28

Pregunta 7

Me imagino que la mayoría de la gente aprendería se adaptaría a la metodología de trabajo muy rápidamente.		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	0	0%
En desacuerdo	0	0%
Neutro	3	33.3%
De acuerdo	6	66.7%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 22

Pregunta 7



Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 3 de ellas perteneciente al 33.3% señalaron la opción neutra ante la interrogante planteada mientras que 6 personas dando el 66.7% señalaron

que están de acuerdo en que la mayoría de las personas aprenderían y se adaptarían a la metodología de trabajo rápidamente.

Pregunta 8: Encontré la metodología de trabajo muy complicado de usar.

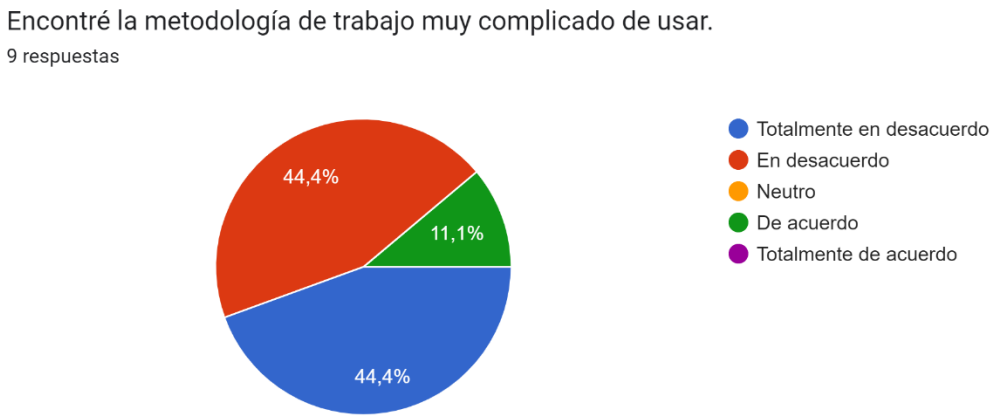
Tabla 29

Pregunta 8

Encontré la metodología de trabajo muy complicado de usar.		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	4	44.4%
En desacuerdo	4	44.4%
Neutro	0	0%
De acuerdo	1	11.1%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 23

Pregunta 8



Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 4 de ellas perteneciente al 44.4% señalaron que están

totalmente en desacuerdo ante la pregunta planteada mientras que 4 personas dando el 44.4% señalaron que están en desacuerdo y 1 personas dando el 11.1% está de acuerdo en que la metodología de trabajo es complicada de usar.

Pregunta 9: Me sentí muy seguro usando la metodología de trabajo

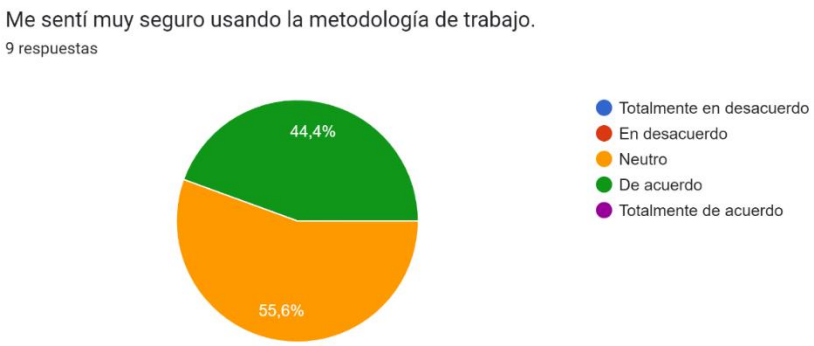
Tabla 30

Pregunta 9

Me sentí muy seguro usando la metodología de trabajo		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	0	0%
En desacuerdo	0	0%
Neutro	5	55.6%
De acuerdo	4	44.4%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 24

Pregunta 9



Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 5 de ellas perteneciente al 55.6% señalaron la opción

neutra ante la interrogante planteada mientras que 4 personas dando el 44.4% señalaron que están de acuerdo en que se encuentran seguros usando la metodología de trabajo.

Pregunta 10: Necesitaba aprender muchas cosas antes de empezar con la metodología de trabajo.

Tabla 31

Pregunta 10

Necesitaba aprender muchas cosas antes de empezar con la metodología de trabajo.		
Variable	**Frecuencia**	**Porcentaje**
Totalmente en desacuerdo	4	44.4%
En desacuerdo	4	44.4%
Neutro	0	0%
De acuerdo	1	11.1%
Totalmente de acuerdo	0	0%
Total	**9**	**100%**

Figura 25

Pregunta 10

Análisis

La encuesta realizada a las 9 personas arrojo los siguientes resultados dividiéndolas según la respuesta dada por las mismas, 4 de ellas perteneciente al 44.4% señalaron que están

totalmente en desacuerdo ante la pregunta planteada mientras que 4 personas dando el 44.4% señalaron que están en desacuerdo y 1 personas dando el 11.1% está de acuerdo en que se necesita aprender algunas cosas antes de empezar con la metodología de trabajo.

Capítulo V

Ejecución del modelo de trabajo basado en las mejores prácticas de las metodologías tradicionales y no tradicionales.

Introducción del Capítulo

En este capítulo, se describirá cómo se aplicará el modelo de trabajo basado en las metodologías tradicionales y no tradicionales en ACR DIGITALD S.A.S., con el objetivo de implementar las mejores prácticas de la industria del desarrollo de software. Se explicarán los pasos y consideraciones necesarios para adaptar dichas metodologías al contexto específico de la empresa, destacando los beneficios que se esperan obtener.

Análisis del entorno de ACR DIGITAL S.A.S.

Antes de implementar un modelo de trabajo, es crucial comprender el entorno y las características de la empresa. En esta sección, se analizarán los procesos actuales de desarrollo de software en ACR DIGITAL S.A.S., se identificarán las fortalezas y debilidades y se evaluará la capacidad de adaptación a nuevos enfoques.

Historia

ACR Digital Marketing inició sus operaciones en agosto de 2016, representando una valiosa oportunidad para el desarrollo profesional y la generación de empleo en Ambato. Desde sus inicios, ACR ha focalizado sus servicios en el ámbito del Marketing Digital, abarcando campañas, gestión de identidad digital, mantenimiento de sitios web, y ampliando su oferta con el desarrollo de aplicaciones móviles y la reciente incorporación de un centro de llamadas.

Después de cinco años de trayectoria, ACR ha experimentado desafíos y éxitos, destacándose actualmente como una de las pocas empresas en Ambato que ha logrado establecerse en mercados internacionales, especialmente en Miami, Estados Unidos. Sus principales clientes, entre los que se encuentran Florida Education Institute, Pasión Miami y

American Bartending School, han expresado su satisfacción con los servicios personalizados y seguros proporcionados por ACR.

Flujo de Procesos Actuales

ACR Digital Marketing no cuenta con procesos metodológicos establecidos, por ende, el trabajo de implementación se realizará desde la base, tomando en cuenta factores como:

- Flujo de trabajo en el desarrollo de Marketing (SEM)
- Flujo de trabajo en el desarrollo de aplicaciones Web
- Manejo de Herramientas SEO
- Manejo de Consultoría y Social Media

Diseño del modelo de trabajo

Utilizando la revisión académica de la literatura, se configurará un modelo operativo que fusionará aspectos de las metodologías convencionales y las no convencionales, integrando las mejores prácticas identificadas en la industria del desarrollo de software.

Figura 26

Etapas en el desarrollo de software

A partir de dicho análisis, se han identificado y seleccionado cinco etapas fundamentales aplicadas en el proceso de desarrollo de software, cada una elegida por su

relevancia y comprobada aplicación, tal como se representa en la figura 33. Estas etapas se detallan a continuación.

Especificación Requisitos

Propósito

El objetivo de esta etapa es recopilar los requisitos del cliente y del usuario, definiendo claramente las funciones que el software llevará a cabo.

Descripción

Es esencial mantener una comunicación efectiva con los usuarios durante la creación de cualquier software. En esta etapa, se busca capturar de manera precisa lo que el software o sistema debe lograr. Aquí es donde se define los Requisitos funcionales y no funcionales para el desarrollo software, por lo tanto, estas prácticas formaran parte del documento de Especificación de requisitos

Mejores practicas

Requisitos funcionales

Requisititos no funcionales

Análisis Requisitos

Propósito

La extracción de los requisitos del producto de software es crucial en esta etapa. Se requiere aplicar habilidades y experiencia en los procesos de ingeniería de software, así como utilizar técnicas o herramientas que faciliten la visualización de la funcionalidad del software.

Descripción

En el desarrollo de software, es esencial comprender qué funcionalidades debe tener el sistema. El análisis se centra en describir las características que el software debe poseer. Por lo tanto, es crucial analizar los requisitos, ya que permite identificar posibles errores a tiempo. Se ha demostrado que corregir errores en la etapa de análisis es más económico que hacerlo en las fases finales del proyecto. En esta etapa, se han identificado como mejores prácticas de análisis el uso de Diagramas de Casos de Uso e Historias de Usuarios.

Mejores practicas

Historias de usuario

Casos de usos

Diseño

Propósito

La fase de Diseño del producto tiene como objetivo facilitar la realización de las tareas relacionadas con el desarrollo del producto. En esta etapa, se lleva a cabo el diseño del software necesario.

Descripción

Para el diseño de los sistemas web, se debe tomar en cuenta el uso de arquitectura relacionadas a servicios web, a fin de determinar el correcto uso de artefactos que permitan la correcta documentación del proceso arquitectónico y flujogramas del sitio a trabajar.

Mejores practicas

En esta fase, se incluye la creación de la arquitectura que opera mediante las siguientes perspectivas:

Diagrama de clases

Diagrama de secuencia

Modelo conceptual

Desarrollo de software

Propósito

El objetivo de esta etapa es construir o desarrollar el software de un sistema. Esta tarea es llevada a cabo por un equipo de programadores, cuya cantidad y tiempo de trabajo dependen de los acuerdos establecidos en la planificación del proyecto.

Descripción

En esta fase, el equipo se dedica principalmente a la construcción de los componentes del software, incluyendo la documentación y el código. Se hacen uso de los artefactos definidos en la fase de diseño, así como se establece el paradigma de programación, lenguajes de programación, métodos y técnicas necesarios para el proceso de desarrollo de software.

Mejores practicas

Documento codificación

Lenguaje POO

Pruebas

Propósito

Esta fase tiene como objetivo llevar a cabo pruebas sobre la solución propuesta, centrándose en el uso y manipulación realista del producto. El equipo de pruebas y desarrollo se dedica a priorizar y corregir errores, preparando la solución para su implementación.

Descripción

La fase de pruebas desempeña un papel crucial en el proceso de desarrollo de software al detectar los errores de software lo más pronto posible. Se verifica que el software ejecute correctamente las tareas definidas en la especificación de requisitos. Es posible realizar pruebas por separado en cada módulo del software y luego llevar a cabo pruebas integrales. Este proceso asegura la ausencia de errores funcionales y la cumplimentación de cada requisito planificado en la etapa de planificación.

Mejores practicas

Pruebas funcionales

Pruebas unitarias

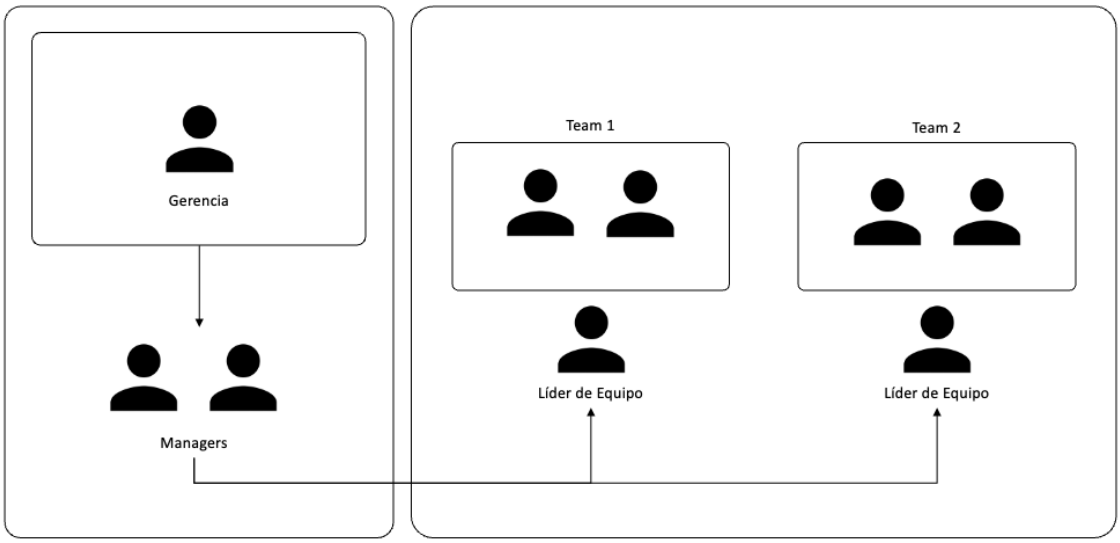
Pruebas de aceptación

Resultados experimentales

Para evaluar la aplicabilidad de las Mejores Prácticas de Software, se ha considerado en realizar la integración de estas en uno de los proyectos internos de la empresa con el nombre de “El sabor de la Tía”, el presente sistema es un gestor del personal, el mismo que funciona como un Webservice. El proyecto hace énfasis en las áreas de administración, gestión y planificación, dicho lo siguiente el primer paso a realizar es organizar a los entes involucrados, para lo cual se esquematizo el flujo actual de trabajo, ver figura 27.

Figura 27

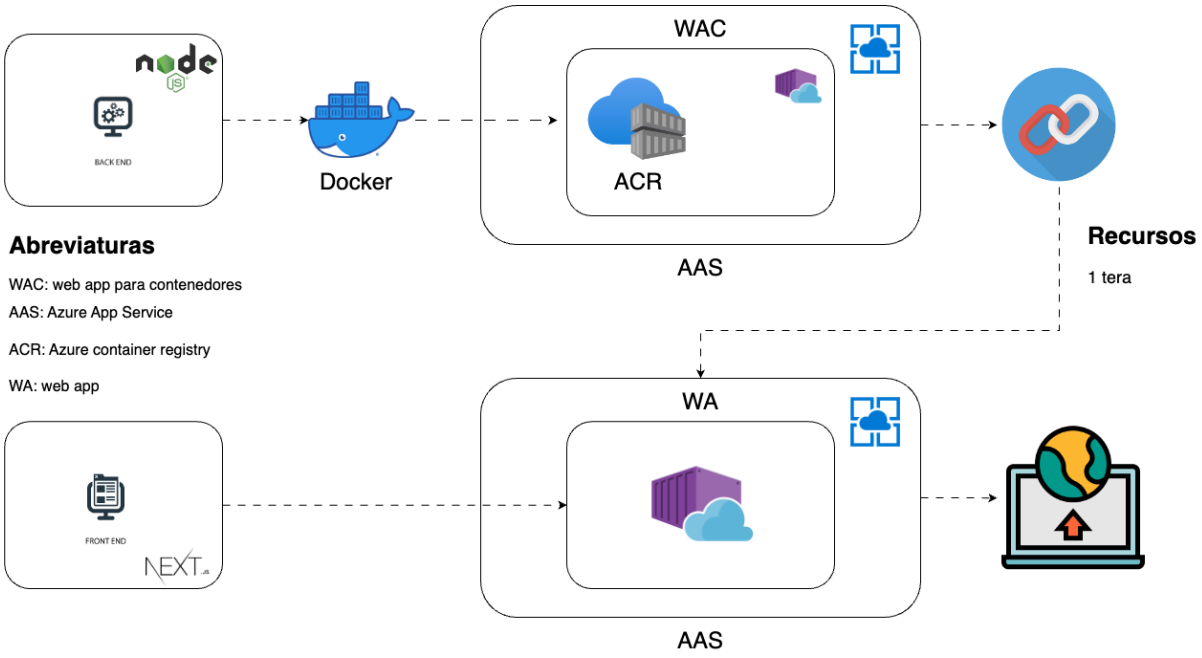
Flujo de comunicación



Con el flujo definido se busca implementar la arquitectura de la aplicación, para lo cual se hará uso de contenedores, permitiendo el manejo ágil y el despliegue continuo de la misma como se observa en la figura 28.

Figura 28

Arquitectura de la aplicación



Especificación de requisitos

El enfoque principal del proyecto es implementar un instrumento de gestión. El equipo responsable del desarrollo de software para el sistema comenzó por definir las características del mismo. Para lograrlo, se llevó a cabo la fase de Especificación de Requisitos, donde se detallaron las especificaciones del sistema. A continuación, se describe la fase de especificación de requisitos.

Análisis de requisitos y Desarrollo de software

Para poder desarrollar el presente sistema se escogió el lenguaje de programación js para el back usando el framework de node y para el front se hará uso de React, debido a que se busca el máximo rendimiento del sistema, tomando en cuenta los flujos de órdenes y usuarios que debe manejar, A continuación, se establecen las historias de usuario, expansión de las mismas y desarrollo de cada una de ellas.

Tabla 32

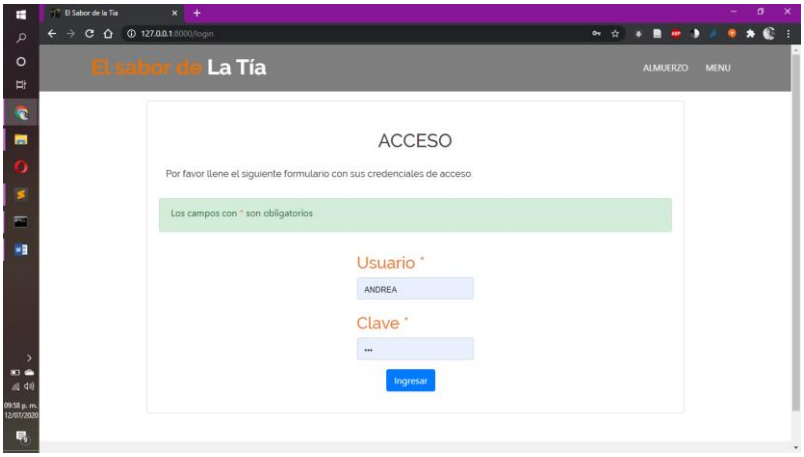
Historia de Usuario número 1

Número de requisito	**HU-0001**
Nombre de requisito	Ingresar Credenciales
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	El sistema mostrará un formulario para ingresar el usuario y contraseña con el que el usuario podrá logearse y se mostrará el perfil dependiendo al rol de usuario
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

- Ingreso al sistema oculto debido a que los usuarios catalogados como clientes no tendrán acceso al mismo
- Cada rol de cada usuario debe estar condicionado en vistas y su respectivo menú de opciones
  - ✓ Administrador: Todo el sistema
  - ✓ Cajero: Clientes y Orden de Venta
  - ✓ Cocinero: Almuerzo y Menú

Figura 29

Pantalla de Login e Ingreso al sistema



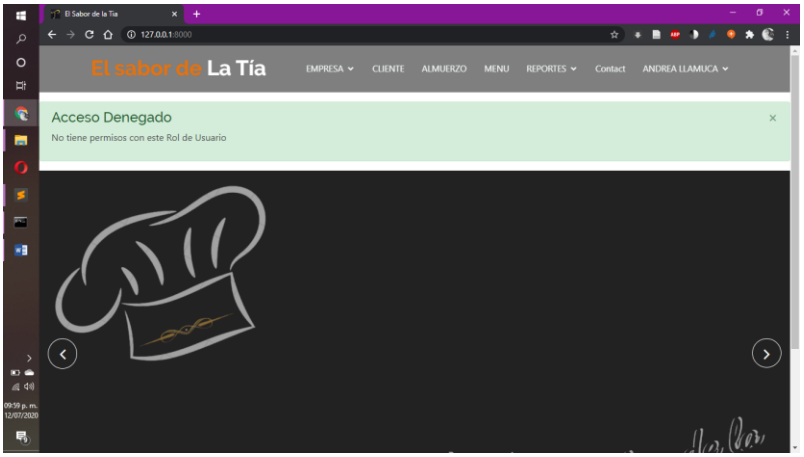


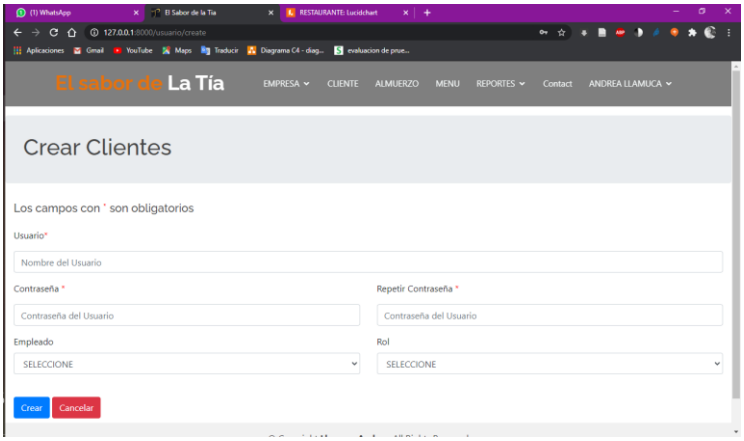
Tabla 33

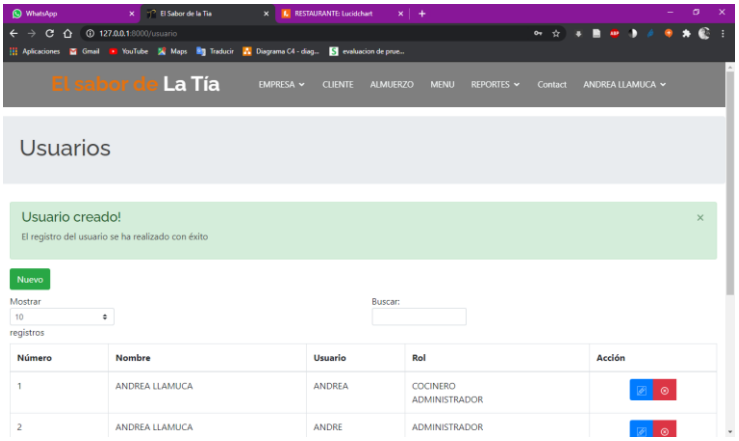
Historia de Usuario número 2

Número de requisito	**HU-0002**
Nombre de requisito	Gestión de usuarios
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	El sistema permitirá crear y eliminar usuarios mediante los datos usuario y contraseña; y se permitirá listar en la pestaña de usuarios
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Figura 30

Pantalla de Login e Ingreso al sistema





Nota: La presente vista permite al administrados crear usuarios con diferentes restricciones como no repetir el nombre de usuario y la contraseña ser iguales en los dos campos

Tabla 34

Historia de Usuario número 3

Número de requisito	**HU-0003**
Nombre de requisito	Gestionar Empleados
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	El sistema permitirá crear, modificar y eliminar a los empleados mediante los datos nombre, cédula, dirección, teléfono, email, grado de estudios, experiencia, foto, área de trabajo, comentario, referencias personales y se permitirá listar en la pestaña de empleados
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Figura 31

Proceso de adicionar usuarios

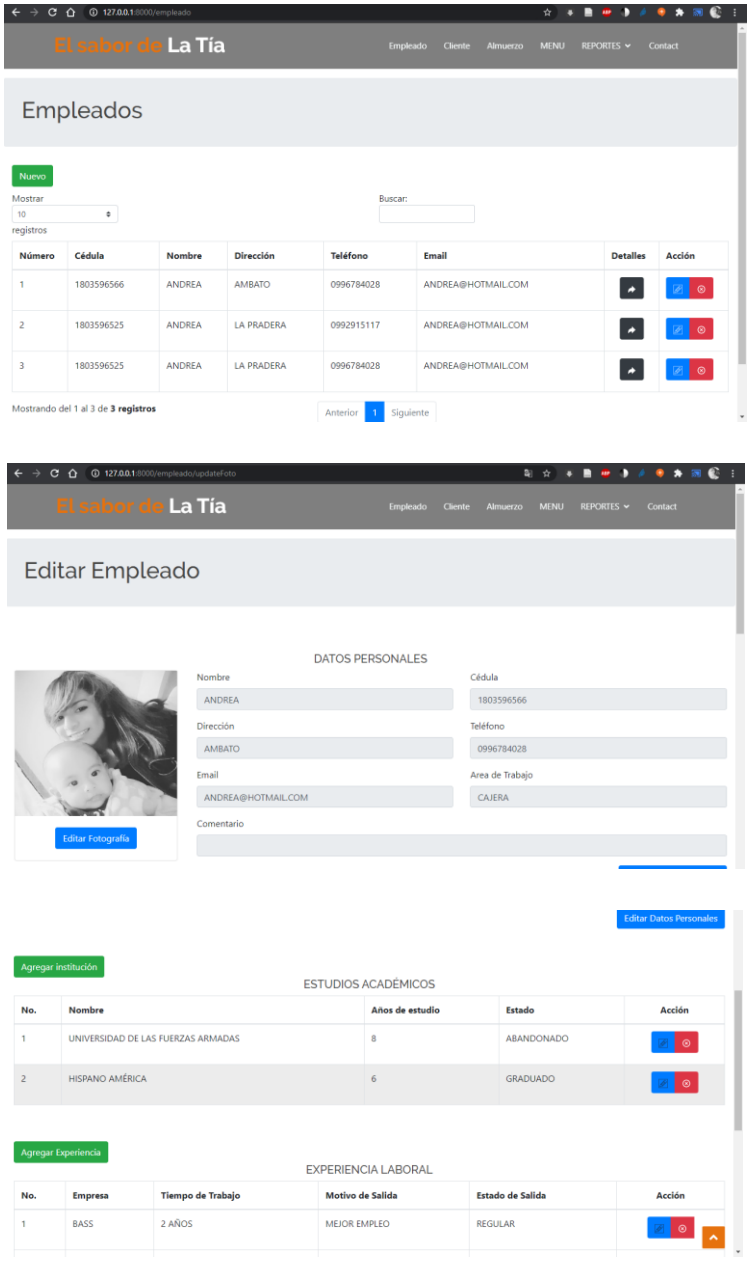


Tabla 35

Historia de Usuario número 4

Número de requisito	**HU-0004**
Nombre de requisito	Gestionar el menú que posee todos los días el restaurante
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	El sistema permitirá crear, modificar y eliminar platillos mediante los datos nombre, costo, hora y días de preparación y se permitirá listar en la pestaña de Menú
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Figura 32

Proceso de ingreso de órdenes

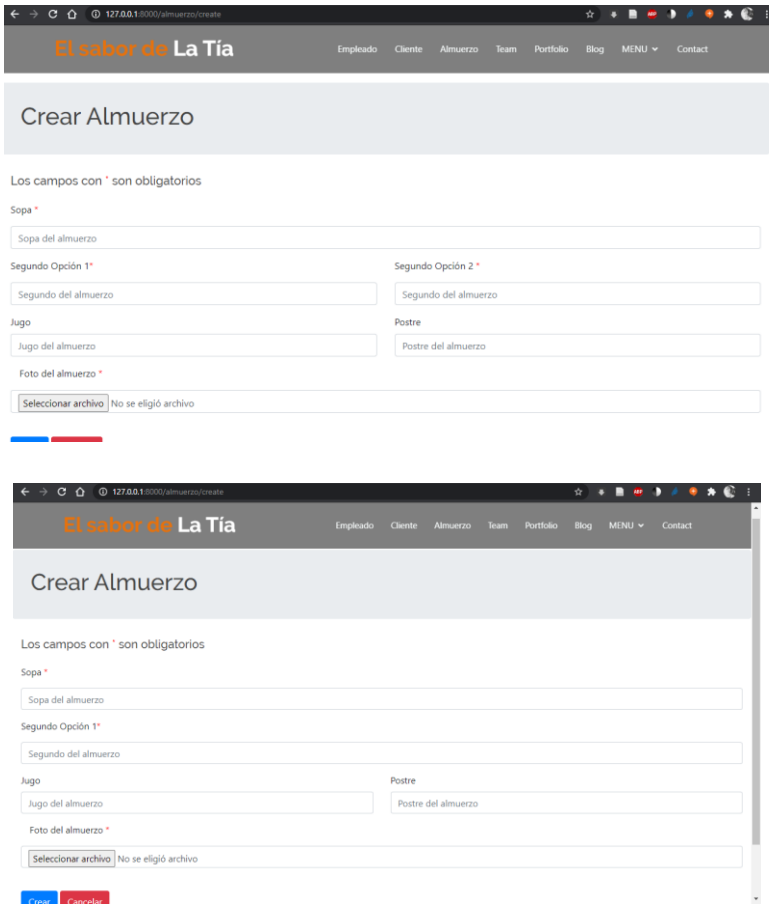


Figura 33

Pantalla menú

Tabla 36

Historia de Usuario número 5

Número de requisito	**HU-0005**
Nombre de requisito	Registro de las órdenes de ventas
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	El sistema permitirá listar y pagar órdenes de venta que se encuentran en la lista mediante los datos de los productos despachados y permitir listar en un historial de órdenes
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Figura 34

Proceso registro de órdenes

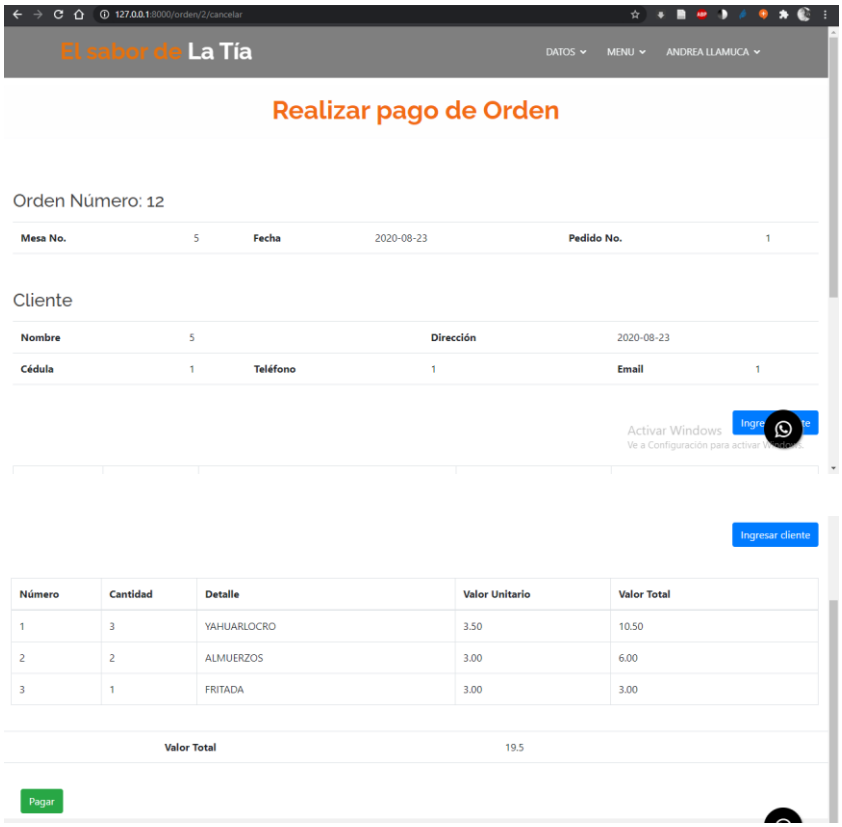


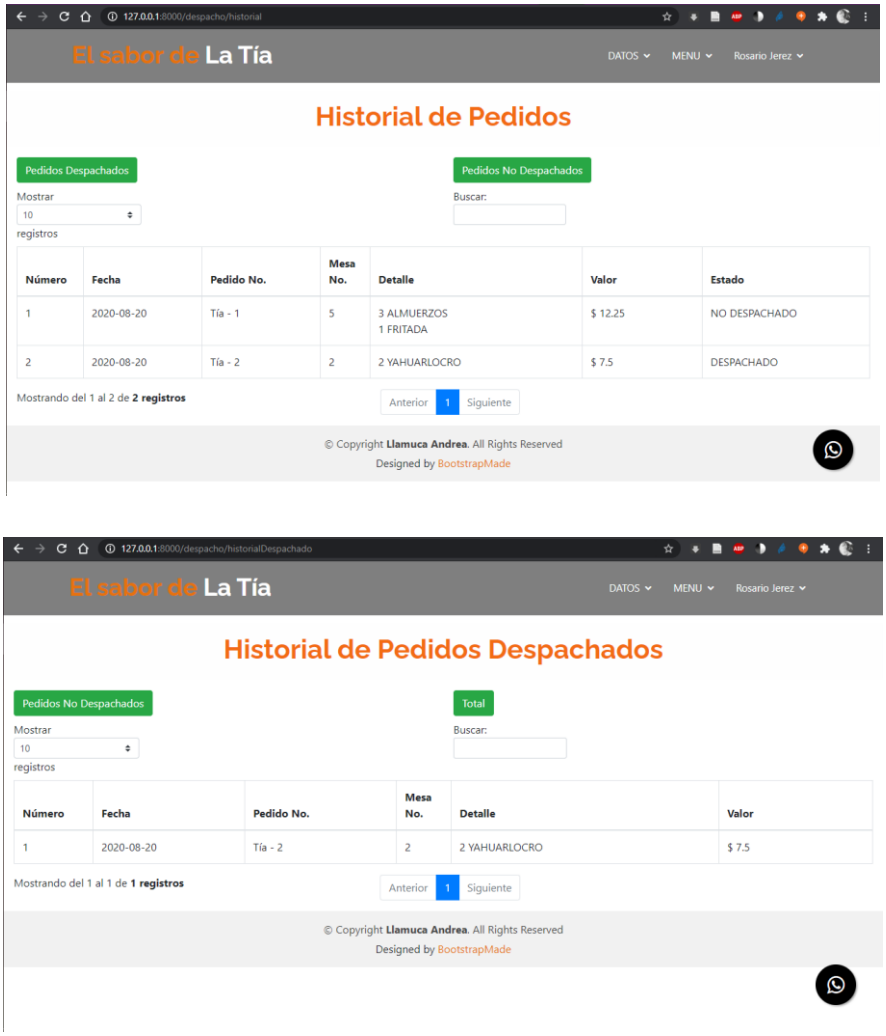
Tabla 37

Historia de Usuario número 6

Número de requisito	**HU-0006**
Nombre de requisito	Despachar Productos
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	El sistema permitirá visualizar información de los pedidos que el mesero realiza en las mesas y poder despacharlo por el cocinero cambiado el estado
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Figura 35

Proceso despachos



Con el desarrollo del sistema finalizado procedemos a evaluar la factibilidad del modelo de trabajo finalizado, obteniendo resultados beneficiosos para la institución y en el proceso de desarrollo de software, los mismos se muestran en la tabla 55.

Tabla 38

Modelo de trabajo

Mejores Practicas	Etapas				
	Especificación de requerimientos	**Análisis de Requisitos**	**Diseño**	**Desarrollo de Software**	**Pruebas**
Requisitos no funcionales	X				
Diagrama de casos de uso	X				
Historia de Usuario		X			
Diagrama de clases		X			
Modelo conceptual			X		
Documentación de código			X		
P. Orientada a. objetos				X	
Pruebas Funcionales				X	
Pruebas de aceptación				X	
Pruebas unitarias					x

Nota: Esta representación gráfica exhibe el modelo de trabajo establecido con las fases, artefactos y entregables necesarios para la construcción del sistema.

Integración de herramientas y tecnologías

El uso de herramientas y tecnologías adecuadas es fundamental para el éxito de un modelo de trabajo en el desarrollo de software. En esta sección, se discutirán las herramientas y tecnologías recomendadas para respaldar la implementación del modelo propuesto. Esto puede incluir sistemas de control de versiones, herramientas de seguimiento de proyectos, herramientas de colaboración y automatización, entre otros.

El Tablero Kanban es una herramienta visual empleada en Scrum para visualizar el flujo de trabajo y el estado de las tareas. Se estructura en columnas que representan las diferentes etapas del proceso, y se utilizan tarjetas para representar las tareas individuales. Permite una gestión eficiente y transparente de las tareas en el equipo.

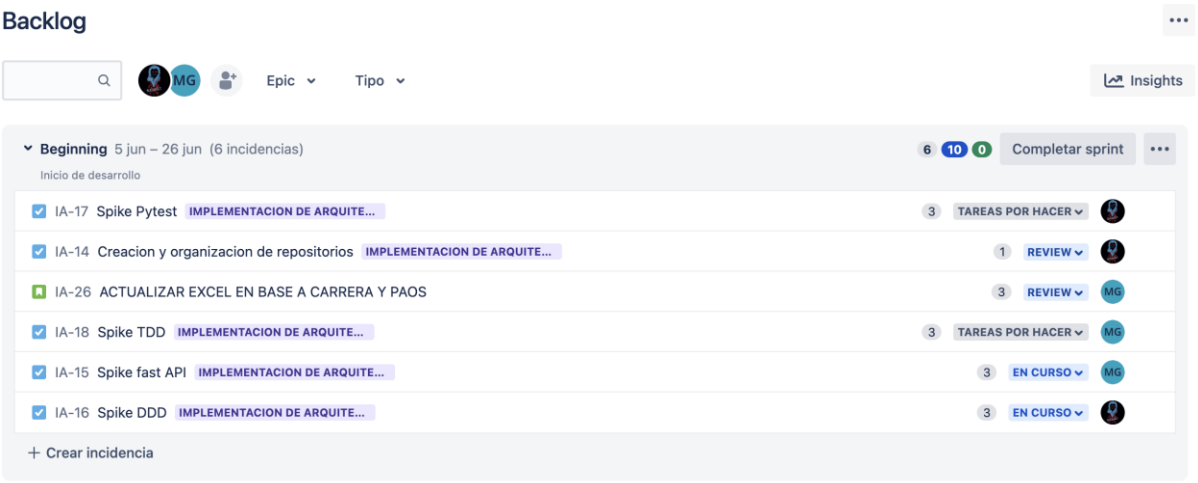
Figura 36

Tablero de Trabajo

Burndown chart: Es una herramienta gráfica que muestra la cantidad de trabajo pendiente en un sprint a lo largo del tiempo. Permite visualizar el progreso del equipo y si están en camino de completar todas las tareas. Ayuda a identificar posibles desviaciones y tomar medidas correctivas.

Figura 37

Trabajo Pendiente



Reuniones diarias (Daily Scrum): Son reuniones cortas y diarias donde el equipo de Scrum comparte el progreso, los desafíos y las próximas tareas. Se realiza en pie y cada miembro responde a tres preguntas: ¿Qué hice ayer?, ¿Qué haré hoy? y ¿Hay algún impedimento?

Sprint Backlog: Se refiere a una lista de tareas seleccionadas del Backlog de Producto que se deben abordar durante un sprint particular. El Sprint Backlog establece las tareas que el

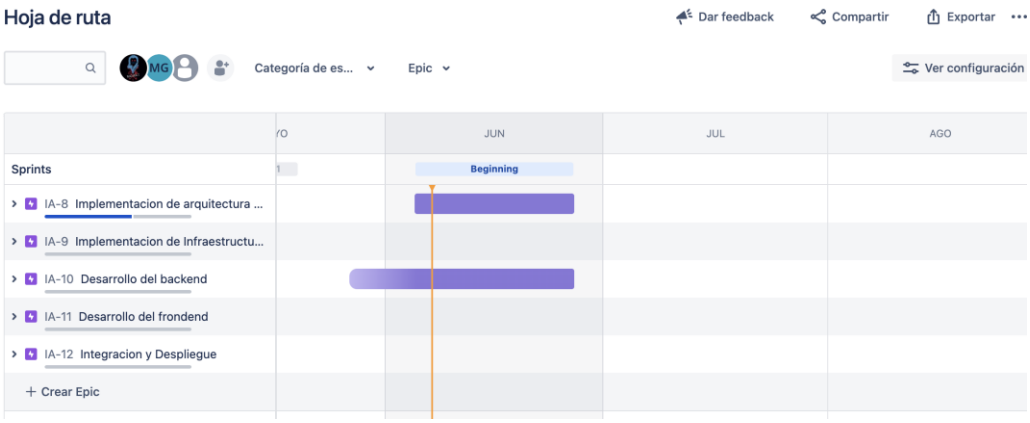
equipo se compromete a completar durante ese sprint y se ajusta a medida que se agregan o eliminan tareas.

Product Backlog: Consiste en una lista que comprende todas las funcionalidades, características y mejoras que se aspira incorporar en el producto. El product backlog es dinámico y se actualiza de forma continua a medida que se obtiene retroalimentación y se realizan cambios en los requisitos.

Épicas (Epics): Las épicas son historias de usuario a gran escala, que representan una funcionalidad o conjunto de características amplias y completas dentro de un producto. Son desglosadas en historias de usuario más pequeñas y manejables para su implementación en los sprints.

Figura 38

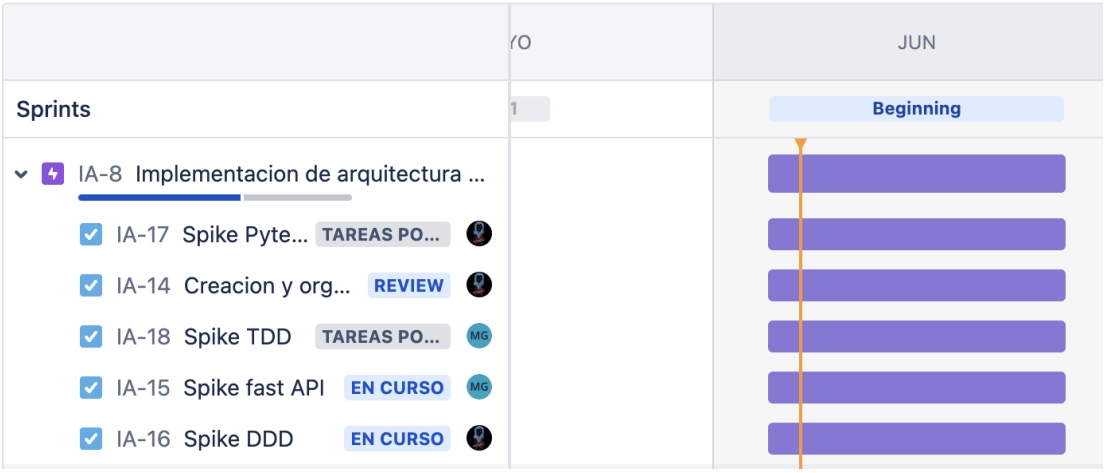
Épicas



Casos de Uso: Son descripciones detalladas de las interacciones entre un sistema y los usuarios o actores participantes. Su función principal es proporcionar una comprensión detallada de cómo se empleará el sistema, destacando las acciones específicas que se anticipan en diversos escenarios. Los casos de uso resultan valiosos para la identificación de requisitos y la definición de funcionalidades del sistema.

Figura 39

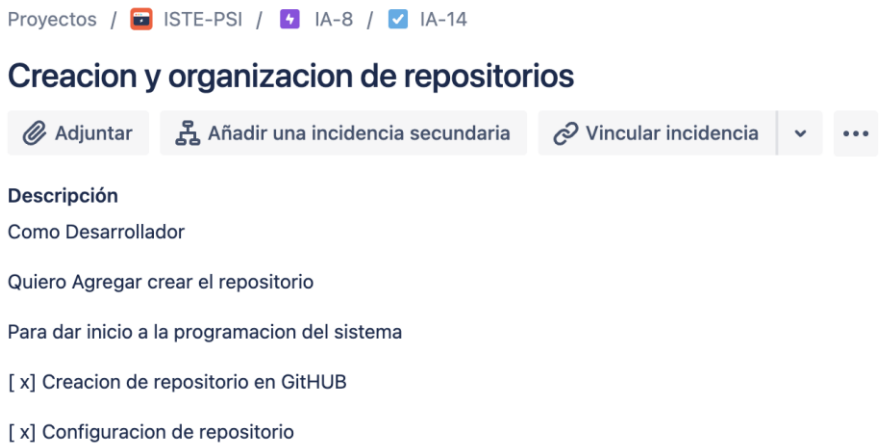
Casos de Uso



Historias de usuario: Las historias de usuario son breves descripciones centradas en el usuario de una funcionalidad específica del producto. Siguen un formato simple y conciso: "Como [rol del usuario], quiero [acción] para [objetivo o beneficio]". Las historias de usuario desempeñan un papel crucial al comprender y priorizar las necesidades de los usuarios, además de servir como base para la planificación y el desarrollo iterativo del software.

Figura 40

Historia de Usuario



Capítulo VI

Conclusiones y recomendaciones

Conclusiones

- El desarrollo de esta tesis, basado en el análisis bibliográfico académico permitió obtener un modelo para soporte de desarrollo de software, mediante la implementación de las mejores prácticas, en la industria de desarrollo de software para la empresa ACR DIGITALS S.A.S.
- La integración de diferentes metodologías tanto tradicionales y no tradicionales puede proporcionar un enfoque más completo y adaptable para afrontar las necesidades específicas en cada proyecto. Los equipos de desarrollo pueden enfrentar los desafíos cambiantes del proyecto y entregar un producto final de alta calidad.
- Con base en la investigación realizada, es práctico describir cada fase en el desarrollo de software, se ha podido identificar una serie de técnicas comunes y esenciales en el proceso. En la etapa de planificación se destacan elementos como las historias de usuario, los casos de uso, los diagramas de clases y de secuencia, así como el modelo conceptual.
- Los estudios revisados de diferentes técnicas y procedimientos comunes obtenidos de desarrollo de software, son fundamentales para mejorar el mismo. Estas técnicas, ofrecen un conjunto de prácticas para optimizar el proceso de desarrollo de software en diversas áreas.
- La comunicación efectiva, la colaboración, la formación adecuada y las revisiones periódicas son elementos clave para el éxito de la aplicación de las mejores prácticas de las metodologías tradicionales y no tradicionales.
- Los estudios revisados ofrecen valiosas técnicas en pruebas funcionales, pruebas de aceptación y pruebas unitarias durante la fase de pruebas de software, las cuales desempeñan una función fundamental en la obtención de productos de alta calidad. Estas técnicas, optimizan el proceso de desarrollo de software y garantizar la detección temprana

de errores y la validación efectiva de la funcionalidad del software, lo que contribuye significativamente a la mejora continua y al cumplimiento de los estándares de calidad en cada etapa del ciclo de desarrollo.

- El modelo desarrollado no solo se alinea con los principios del método ágil clásico, sino que también mejora la disciplina del proceso al incorporar opciones de gestión. Esto facilita la creación de documentos y artefactos, apoyando a los desarrolladores a lo largo del proceso de desarrollo de software.

Recomendaciones

- Analizar cuidadosamente las necesidades y características del proyecto antes de seleccionar una metodología. No existe un enfoque único que sea adecuado para todos los proyectos.
- Considerar la posibilidad de utilizar un enfoque híbrido que combine elementos de diferentes metodologías. Esto permite adaptar el enfoque a las necesidades específicas del proyecto y aprovechar lo mejor de ambos enfoques.
- Fomentar la comunicación y la colaboración efectivas dentro del equipo. Esto es esencial para el éxito de cualquier metodología, ya que facilita la coordinación, el intercambio de ideas y la resolución de problemas.
- Proporcionar una formación adecuada a los miembros del equipo sobre la metodología seleccionada. Esto asegura que todos estén alineados y tengan una comprensión clara de los principios y prácticas a seguir.
- Realizar revisiones periódicas del progreso del proyecto y la eficacia de la metodología.

Bibliografía

Amaro Calderón, S., & Valverde Rebaza, J. (2007). *Metodologías ágiles.* Perú.

Bustos, G. (2002). Integración Informal De Modelos En Uml. *Ingenerare*.

Cadavid, A., Martínez, J., & Vélez, J. (2013). Revisión de metodologías ágiles para el desarrollo de software.

Canós , J., Letelier, P., & Penadés, M. (2003). Metodologías ágiles en el desarrollo de software. Valencia: Universidad Politécnica de Valencia.

Cataldi, Z. (2000). *Una metodología para el diseño, desarrollo y evaluación de software educativo .* Universidad Nacional de la Plata.

Cendejas Valdéz, J., Vega Lebrún, C., Careta Isordia, A., Gutiérrez Sánchez, O., & Ferreira Medina, H. (2014). *Diseño del modelo integral colaborativo para el desarrollo ágil de software en las empresas de la zona centro-occidente en México.*

Delgado Olivera, L., & Díaz Alonso, L. (2021). Modelos de Desarrollo de Software. *Revista Cubana de Ciencias Informáticas*, 37-51.

Figueroa, R., Solís, C., & Cabrera, A. (2008). Metodologías tradicionales vs. metodologías ágiles. Universidad Técnica Particular de Loja.

Garcés , L., & Egas, L. (2013). Evolución de las Metodologías de desarrollo de la Ingeniería de software en el proceso la Ingeniería de Sistemas Software. *Revista Científica y Tecnológica UPSE*, 1-3.

Gómez, J., & Fuentes, M. (2012). Taxonomía de los modelos y metodologías de desarrollo de software más utilizados.

Letelier, P., & Penadés, M. (2006). *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*

Maida, E., & Pacienzia, J. (2015). *Metodologías de desarrollo de software.*

Mariño, S., & Alfonzo, P. (2014). Implementación de SCRUM en el diseño del proyecto del Trabajo Final de Aplicación. *Scientia et technica*, 413-418.

Martínez, R. (2015). *El proceso de desarrollo de software.* IT Campus Academy.

Montero, B., Cevallos, H., & Cuesta , J. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espirales revista multidisciplinaria de investigación*.

Ojeda, J., & Fuentes, M. (2012). Taxonomía de los modelos y metodologías de desarrollo de software más utilizados.

Pressman, R. (2010). *Ingeniería de Software.* MCGRAW HILL EDDUCATION.

Rivas, C., Corona, V., Gutiérrez, J., & Hernández, L. (2015). Metodologías actuales de. Vol2, No.5.

Sommerville, I. (2011). *Ingeniería de Software (9na ed.).* Pearson Educación.

Tinoco Gómez, O., Rosales López, P., & Salas Bacalla, J. (2014). *Criterios de selección de metodologías de desarrollo de software.* Industria Data.

Villacis, S. (2013). *El uso de las NTIC´S en el proceso de enseñanza-aprendizaje de los estudiantes del Octavo, Noveno y Décimo Año de Educación Básica del Centro Educativo "Jerusalén"*. Obtenido de Repositorio Académico de la Universidad Técnica de Ambato: http://repositorio.uta.edu.ec/bitstream/123456789/4676/1/ti_2010_10.pdf

Zerón Acastenco, S. (2014). *Conceptos Básicos de metodología.* DIVULGARE Boletín Científico de la Escuela Superior de Actopan.

Zumba Gamboa, J., & León Arreaga, C. (2018). Evolution of the methodologies and models used in software development. *Universidad Internacional del Ecuador INNOVA Research Journal*, 20-33.

Anexos